Time Series Analysis and Forecasting of the US Housing Starts using Econometric and Machine Learning Models

by Sudiksha Joshi

under the Direction of

Professor Michael Robinson

A Thesis

Submitted to the Faculty of Mount Holyoke College

in partial Fulfillment of the Requirements for the Degree of

Bachelor of Arts with Honors

Economics Department

Mount Holyoke College

South Hadley, MA 01075

Abstract

In this thesis, I have performed time series analysis and forecasted the monthly value of housing starts for the year 2019 using several econometric and machine learning algorithms. In the rapidly emerging field of artificial intelligence, data scientists are heavily improvising and using machine learning models to predict any variable. I have applied a few popular techniques from machine learning to predict housing starts for the year 2019. Some of these methods are - artificial neural networks, ridge regression, K-Nearest Neighbors, and support vector regression, and created an ensemble model. The ensemble model stacks the predictions from various individual models, and gives a weighted average of all predictions. In my analysis, the ensemble model has performed the best among all the models as the prediction errors are the lowest. The econometric models have higher error rates than the machine learning models. In my analysis, I have elucidated each model mathematically and made plots to compare forecasts from each model. Lastly, I have stated the limitations of their usage and ways to enhance the current model.

Acknowledgements

I would like to thank Professor Michael Robinson for being my thesis advisor. Furthermore, I express my gratitude to Professor Evan Ray who taught me Applied Regression Methods. Without his patience and help in debugging coding errors in R for a class project, I would not have been able to implement a few of the complex machine learning models applied in the thesis. Special thumbs up to my friends, classmates, and also economics majors − Marihah and Abena, who worked with me in my class group project. Some of the techniques learned in the project enlightened me with novel ideas that were cornerstone in building the second half of my analysis. Last but not the least, I applaud the effort that Dawn Larder, the Coordinator of the Economics Department, put in helping me format the document.

Table of Contents

List of Tables

List of Figures

1. Introduction[1]

In this empirical paper, I have forecasted the housing starts in the United States. Housing starts are the number of new residential construction projects that have begun in a month. The common techniques of forecasting include econometric time series modeling and machine learning (ML). A subset of artificial intelligence, machine learning explores how to construct and analyze algorithms that can learn from data. Machines learn from patterns observed in lagged and exogenous variables (if given) and can predict the endogenous variable. While economists in decision-making can make causal inferences to understand the relationship between variables from econometric models and test hypothesis, finding insights and patterns are not relevant for ML models. Recently, econometricians are increasingly incorporating ML models such as neural networks and decision trees to enhance the model's predictive performance. The goal in ML is to obtain the lowest generalization error among the different systems built. One of commonalities between ML and econometric models is that they both aim to enhance accuracy of forecasts by minimizing some loss function such as sum of squared errors using different approaches −ML methods are, however, more computationally expensive.

I have chosen housing starts as it is a leading indicator in the real estate or mortgage market . This forward looking variable estimates a good gauge for future levels of real estate supply, and creates a ripple effect in the overall economy. Its is primarily of interest as the inception and collapse of the housing bubble in 2007-08 were the turning points in the subsequent developments that embroiled the American and the Eurozone economies in a deep-seated financial meltdown. Buying new houses also increases the demand of complementary durable goods such as furniture, refrigerators, etc. Thus, new residential construction boosts employment in construction, raw materials, banking and other manufacturing sectors. Mortgage rates directly affect housing activity as higher interest rates raise the housing expenses. This lowers the number of qualified borrowers, declining home sales, and housing starts. Contrarily, lower interest rates make houses affordable, spurring housing starts and home sales.

This thesis is divided into the following sections. Chapter 2 reviews the methodologies adopted in the statistical and economic literature useful in forecasting (2.1 and 2.2, respectively). Chapter 3 breaks down the complete analysis into two parts. Firstly, chapter 3.2 examines the econometric models in depth (from

[1] The dataset used in this thesis is different from that used in the class project as it has an extra column of variable, and more number of observations. Furthermore, the models used in this thesis are very different from those in the project, but derived from them.

3.2.1 to 3.2.11). I have used a few basic and advanced econometric tools to conduct time series analysis by using ARIMA, ARIMAX, GARCH and VARX models. In this process, I tested for stationarity, cointegration, conditional heteroskedasticity, and error correction. Thereafter, chapter 3.3 investigates the machine learning algorithms and the ensemble models comprising of Artificial Neural Networks, Support Vector Regression, Ridge Regression and K-Nearest Neighbors models. I have measured the forecast accuracy through mean absolute percent errors and percent bias in the test sets. The ensemble model stacks predictions from these methods to predict the number of housing starts in the test set. It applies different weights to predictions from each of the aforementioned machine learning models, resulting in the weighted average predictions (from 3.3.1 to 3.3.7). Chapter 4 compares the results from each model to gain insight into relative model performance, and displays forecasts from the econometric and ML models for the year 2019. Chapter 5 highlights the stumbling blocks in conducting the time series analysis, potential ways to overcome them, and future research in machine learning forecasting. Chapter 6 concludes the analysis by stating that the ensemble model performs the best among both ML and econometric models as it has the lowest prediction error in the test set. The Appendix in chapter 7 contains the relevant plots, tables and models. Finally, the Bibliography in chapter 8 consists of all the references.

2. Literature Review

In this section, I will discuss the relevant empirical research done to model and forecast the number of housing starts in the US. I will explain the regression and time series models that authors have applied, the dependent and independent variables used, observations and conclusions on how their models and empirical results answer their research question. Also, I will discuss a few econometric and machine learning concepts pertaining to time series. These are helpful in understanding the drawbacks and efficacies of the various methods that the authors have employed in deciding which forecasting tools to use. Primarily, the forecasting tools used are – Vector Autoregression (Thom et al.1985), Vector Error Correction Models (Printzis et al. 2015), Structural Equation Models (West et al. 2000) and advanced machine learning tools such as Artificial Neural Networks (Khalafallah et al. 2008).

2.1. Discussion of the Forecasting Techniques Using Machine Learning Algorithms

In one of the recent studies, Makridakis et al (2018) has compared the performance of eight machine learning models and eight statistical models, deliberated problems arising from measuring forecast accuracy and the computational complexity of the methods. Among the ML methods used were Multi Layer Perceptron (MLP), Bayesian Neural Network (BNN), CART Regression Trees, Radial Basis Function (RBF) etc. A few of the statistical forecasting methods were Naive, Box Jenkins, Holt-Winter etc. Secondly, they made a forecasting model in the first 18 observations, generated forecasts in subsequent 18 observations and evaluated the accuracy by comparing the actual values with the observed values. Computational Complexity (CC) is the total time spent to train a model and extrapolate information from it. Thirdly, the paper discusses the outcomes and explains why the statistical methods produced higher forecasting accuracy than the ML models, and how to enhance the accuracy.

Applying similar techniques, Pavlyshenko (2019) has predicted sales by machine-learning ensemble models using a small set of historical data. These are especially useful for firms who have just launched their store or product(s) and want to predict the volume of sales in the short-run. The models comprising neural networks, LASSO, ridge, ARIMA, random forests and trees are stacked together, resulting in lower error rates in the cross-validation and out-of-sample sets (not used on model training). Neural networks, trees and LASSO have positive weights, while ARIMA and random forests have zero weights. On the

out-of-sample set, one can calculate stacking errors. In the next level, predictions on the cross-validated become regressors for the linear model with Lasso regularization. Overall, stacking methods have enhanced the performance of predictive models in forecasting sales.

Johnson et al. (2016) has forecasted financial time series using machine learning models. Generally, advanced forecasting methods predict changes in price in financial markets with high degree of accuracy, leading to profits in some of the trades based on the predictions. This paper has tried to resolve the contradiction that financial economists have pointed out − market inefficiencies prevent investors from predicting price, and thus trade profitably. So, they made forecasting simulations from thirty four financial indices spanned over six years. These simulations provide evidence that the best machine learning models outperform the best econometric models with respect to accuracy. The paper also scrutinizes the factors that influence predictive accuracy of ML models. The results suggests that model-based trading and predictability are affected by the maturity of the market, forecasting technique used, forecasting horizon, and methods to assess model.

Mullainathan and Spiess (2017) have discussed the basics of machine learning, its applications, how it is used, different kinds of models and their performance, and compared them with OLS models. They also predicted housing value from a dataset comprising of 10,000 randomly chosen owner-occupied units, collected in the 201 metropolitan sample of American Housing Survey. Besides the values of each units, the sample has 150 variables containing information about the units, their location, base area, number of rooms, census region in the US, etc. Comparing various prediction methods, they examined how well each method predicted log unit value on a test set (also called hold-out set) that had 41,808 units from the same sample. Sometimes models such as random forests overfit in the training set, overstating the performance in the in-sample by lowering the error rates. However, the same models may perform poorly in the test set. Nonetheless, random forests perform significantly better than ordinary least square models albeit moderate sample sizes with few covariates. They also created an ensemble model composed of OLS, regression trees tuned by depth, LASSO and random forests. The ensemble and the OLS have the narrowest and widest 95 percent confidence interval, respectively. The paper has discussed different kinds of shrinkage or regularization parameters that add bias and reduce the coefficients to approximately zero in order to reduce the overall variance. In neural networks, the regularizers are the number of hidden layers and neurons in each layer, while in parametric equations, they are LASSO, and ridge regressions.

Focusing on just one ML model to predict housing prices, Khalafallah et al. (2008) has used neural network based models to predict the performance of the housing market. The paper has elaborated on how Artificial Neural Network models[2] can assist home developers and real estate investors in forecasting changes in house price in the short run. This model has several features: artificial neural networks can construct generalized information about the housing industry's current and past performance; it can predict the ratio between asking price and average home sales. The main indicators used in constructing ANN models are the mean interest rate, time- year and month, percentage change in volume of sales compared to that in the previous year, inventory volume, monthly inventory supply, percentage change in the median price of house and average number of days a house is available in the market. The eight input neurons in the ANN structure represent these variables. Its output is the ratio between the house's selling and asking price. The author tested several network structures in unforeseen cases to choose the best ANN structure that would forecast accurately with minimal training.

Khalafallah et al. selected this design as neural networks are very robust in estimating almost any output or input. They are trained, cross-validated and tested many neural structures by differing the number of neurons in each hidden layer, changing the number of hidden layers, applying the transfer function and learning method. The drawback of this model is that it cannot forecast the reaction of the housing market in the long duration. Lastly, the test set and validation set model predicted the error to be in the range of –2% and +2%.

2.2. Discussion of the Econometric Forecasting Techniques on Housing Starts

Wallace et al. (1969) used. a two-equation disaggregated model of residential construction industry. They assumed that both single and multiple family housing starts were related to potential demand, availability, and cost of funds, construction costs, and other conditions of money and housing markets. Using quarterly data from 1960 through the second quarter of 1968, they performed multiple regression and correlation analyses to test the two housing models. The results from the equations were consistent with expectations. The paper shows that housing starts respond negatively mortgage yields and positively to demographic characteristics. Also, single-family starts respond negatively to vacancy rates, implicit price deflator, and

[2] **Neural networks** are data-driven machine learning algorithms. Like smoothing algorithms, they learn patterns from data. Like regression models, they are designed to capture the relationship between input and output variables using cross-sectional data. Neural nets are also used to forecast time series (numerical or binary). It mimics the functionalities of the human brain to solve the problems. It link the input information with output information through a network of neurons. They neurons learn from other neurons, process information using activation functions and generate output.

the Federal Reserve discount rate. Multiple-family starts respond negatively to yields on long-term bonds and rent-price variable, and negatively to the Treasury bill rate.

Single-family housing starts were highly correlated with Mortgage Yield (measures costs of funds), number of Households Headed by an Individual 25-to 34 Years of Age (measures potential demand), Federal Reserve Discount Rate (measures the availability of funds), Implicit Price Deflator for Residential Construction (measures construction costs), and Vacancy Rates (measures the balance of demand and supply in the housing sector).

Housing starts of multiple-family units were highly correlated with Mortgage Yields (a measures costs of funds), number of Persons Reaching Age 21 (a measures additional demand for multiple-family housing), yield on three-month treasury bills (a measures the availability of funds), ratio of rent component of consumer price index to implicit price deflator for residential construction (measures profit margin of investments), and long-term bond yields (measures expected conditions in the bond and money markets). The models indicate that changes in housing starts lag changes in many of the causal variables. The lags imply the time builders need to plan and enforce tasks once the monetary policy stipulated to raise or diminish the number of housing starts.

Ira G. Kawaller and Timothy W. Koch (1981) forecasted the housing starts using aggregated monetary targets to understand how monetary policy impact housing starts. Thus, the flow of funds to financial institutions primarily determine housing starts. This is a supply side or a "supply of funds side" approach. Alternatively, demand-side factors reflect changes in the mortgage rate. Mortgage loans depend on how funds flow to financial intermediaries that grant mortgage loans. The mortgage loans impact the value and pace of housing starts. In this approach, Koch et al. (1981) assume yield on mortgages move in a way such as demand for mortgage credit is equal to its supply.

They ran various ordinary least squares regressions for different types of housings starts. The dependent variables—number of housing starts (in thousands) was regressed on the aforementioned variables. They forecasted using quarterly data from the third quarter of 1962, through the second quarter of 1970. First, they examined the structure of the residential housing market − the number of housing starts. This variable is decomposed into several categories such as conventionally financed single-family owned dwelling units (HC), federally underwritten single-family dwelling units (HFED), Multiple-family dwelling units (HM), and Mobile homes (HMHUS). Then, they forecasted both subsectors of single-family units and for different types of housing starts till the end of 1971.

Koch et al. included a dummy variable to capture the effect of extreme weather that hurt the housing industry in the first three months of 1978 and 1979. They projected the flow-of-funds data (FF*) based on the regression equation estimated quarterly from 1970 through 1979. So, there is a linear relationship between the flow of funds (FF) and CM2. CM2 is the difference between the aggregate M2 of the third month between the current and former quarter. Housing value will be greater if more funds flow. To forecast housing starts, they used the average new home sales price in 1980. The model reduced forecast for housing starts when inflation is higher. This is because consumers can finance fewer units with the same amount of dollars.

Unlike the previous two similar methods followed by Wallace et al and Koch et al., Brady et al. forecasted short-run movements in different types of residential housing (single and multiple family housing starts). The financial variables related to data series are loan-to-value ratios, interest rates, amortization periods on mortgages, costs of residential construction, and savings volume flowing into the financial intermediaries.

According to their analysis, demand variables such as per capita disposable income and relative price of housing services, shift in the long-run only. Their analysis closely relies on supply variables to explain the short-term fluctuations in the housing markets. From the supply side, the interest rate on mortgages, monetary stringency, and rate ceilings kept on savings deposits with financial intermediaries determine the savings inputs to financial intermediaries as they purchase most of the mortgages. Along with the savings inputs, the federally set ceilings on interest rate and loan-to-value ratio imply the volume of mortgage instruments held in the asset portfolio of both public and private sector. Coupled with their participation in the mortgage market are the demand factors which together establish the terms of acquiring mortgages. Finally, the costs of residential construction, and government intervention in the market link and close the supply side factors of housing activity.

Thom et al. (1985) provided evidence on the relationship between mortgage availability and housing starts. Firstly, they employed a four variable vector autoregression (VAR)[3] to compute impulse response functions.[4] The results show that both mortgage availability and interest rates significantly affect housing starts. Secondly, they estimate vector autoregression to decompose starts using 1979(12) as the base

[3] **Vector autoregressive (VAR*)*** models are used in multivariate time series. Here each variable is a linear function of previous lags of itself and that of other variables.

[4] Sometimes, we would like to know how one variable responds to an impulse from another variable in a higher dimensional system comprising of multiple variables. So, economists compute **impulse response relationship** between two variables or find out how the variable reacts or alters behavior due to the impulse.

period. According to the decomposition, the availability effects have weakened as competitive financial markets have evolved and been deregulated. So, the relationship between housing starts and mortgage availability has diminished. Thirdly, they made an unrestricted model that entails observations on mortgage interest rate, mortgage availability, private sector starts, and mortgage availability. They estimate the mean interest rate on long-term bonds the mortgage interest rate as a vector autoregression (VAR).

They use the moving average representation of the VAR to judge how important are interest rates in determining housing starts since financial deregulation in 1980. In the model to showcase the relationship, the variables they used are average interest rates on long-term government bonds, interest rates on new home mortgages, mortgage availability and the number of privately-owned housing starts. Government bonds rates are the proxy for capital market rates. The IRFs suggest that housing starts are persistently responsive to shocks in mortgage availability and interest rates. The responses of housing starts to an availability shock are not very significant and of a smaller magnitude than the responses to interest rate shocks.

West et al. (2000) measured the forecasting accuracy of regional single-family housing starts. They have predicted residential construction in regional housing markets. To analyze the accuracy of the forecast, the data consists of quarterly forecasts (from previously published econometric papers) of Florida and its largest six metropolitan economies. The sample consists of simulation from periods 1985:1 - 1996:2 and three phases of the business cycle: expansion, recession, and recovery. Thereafter, the paper compare the forecasts of the single-family unit starts with univariate time series and random walk alternatives and the results suggest that multi-family housing starts perform better than single-family starts for the same housing market in Florida.

Regional structural equation models (RSEMs)[5] are constructed to analyze expected changes in the supply of new housing units. Furthermore, they used random walk[6] and univariate ARIMA models to assess the accuracy of regional econometric models that forecast residential construction. They made the data on

[5] **Structural equation models** are models used to construct multivariate statistical analysis i.e. a combination of multiple regression analysis and factor analysis. It measures structural associations between latent constructs and variables. In a single analysis, it computes interrelated and multiple dependence. Here two types of variables are exogenous and two are endogenous. The dependent variables are endogenous variables are equivalent to the explanatory variables.

[6] A time series follows a **random walk** pattern if the first difference of observation i.e difference between observation at time t and time (t-1) are random.

single-family housing starts stationary by taking the first difference. The autocorrelation functions highlight autoregressive and moving average parameters. Finally, they calculated the Modified Theil inequality coefficients, which is the ratio of RMSEs of the structural model to those associated with a random walk and univariate ARIMA housing starts equations. The modified Theil inequality coefficients are reliable indicators as it forecast models with higher accuracy. If a modified Theil inequality coefficient is greater than one, it means that ARIMA, univariate or a random walk model yields smaller absolute forecast errors than the structural model. Contrarily, if the inequality coefficient is less than one, it means that the prediction errors from RSEMs are less than those of univariate, ARIMA or a random walk model. The paper concluded that the accuracy of structural model that forecasts single family construction activity is low relative to those of univariate ARIMA and random walk models.

Puri et al. (1988) constructed a multi-equation econometric model of the US housing market and contrast its viability with the forecasts generated from ARIMA models. The econometric model has credit and cost availability variables and stock-flow structure focused on the housing supply equation. They made two types of forecasts up to four time periods ahead for both time series and parametric models.

The two-time series models shown in the paper are the univariate ARIMA model and a bivariate leading indicator model. In the latter, they constructed a time series with private dwelling units as a leading indicator series. Of the two, the ARIMA model performs better as it is difficult to forecast indicator series and there is a short lag between housing starts series and indicator series. Hence, they dropped the indicator model from the analysis. Also, ARIMA forecasts have minimal bias and a negligible proportion of error due to bias. They used actual and forecast values of the exogenous variables to conduct simulations. The exogenous variables are CPI, public housing dwellings, weighted population, credit availability by mortgage agencies, T-bill rate (3 months bond), rate on 3-5 year treasury bond rate, effective interest/dividend rate paid by insured S&L associations. The paper concluded that time series forecasts were less superior to "unaided one-period ahead econometric forecasts." However, as econometric models are biased, predictions made by ARIMA models are better in the long run.

In the literature review above, I have examined the scholarly empirical research papers wherein economists have used various methods to forecast housing starts or similar variables of the real estate market. I observed that statisticians and economists have not undertaken much research in this field using the advanced machine learning tools as it is a relatively emerging field. So, I will study and use a few of those methods, forecast housing starts, and compare my results with those from the econometric methods that I have described in the literature review.

3. Data Analysis

3.1. Description of the Data

First, I will detail the the characteristics of the dataset before giving a synopsis on the econometric tools used for time series analysis. I have used monthly data of all variables from Federal Reserve Bank of St. Louis Economic Research (FRED) from June 1976 to December 2018. Totalling 511 observations, the dependent variable is the number of housing starts. Below is the list of all the variables.

1. **hous_st :** The number of privately owned housing starts for each month (in 1000s of units).
2. **CPI:** The consumer price index measures the weighted average of prices of a fixed basket of consumer goods and services. It assesses changes in cost of living standards with respect to inflation. A higher CPI implies higher inflation, reducing people's disposable income/savings, and vice versa.
3. **mortgR:** A 30 year mortgage rate that measures the interest rate charged when financing a new home.
4. **fed_fundsR:** The federal funds rate is the interest rate that banks charge other other banks for overnight loan requests to meet the federal reserve requirement.
5. **income:** The real disposable personal income per person on average (in billions of USD).
6. **pvt_house_comp:** The number of new privately owned houses completed (in 1000s of units).
7. **sec_conL:** The number of securitized total consumer loans outstanding (in billions of USD). Through securitization, financial institutions distribute various assets such as residential mortgages, and auto loans to third party investors to generate a sustainable cash flow.[7]
8. **real_estL:** The number of real estate loans from all commercial banks in billions of USD.
9. **yield_sp:** The difference between a 10 year treasury bond and a 2 year treasury bond.[8]

[7] Through **securitization,** financial institutions pool diverse array of contractual debt such as commercial, residential mortgages, credit card debt obligations, auto loans, and other non-debt assets. By selling these assets (called securities) to third party investors, they generate cash flows. These securities can be pooled to generate collateralized debt obligations (CDOs), bonds, credit default swaps (CDS), mortgage backed securities (MBS), and asset backed securities (ABS).

[8] In general, **yield spread** is the difference between the return or yield between any not debt instruments, not just treasury bonds. For instance, if a 10 year treasury bond is trading at 6 percent, and a 3 months treasury bill

10. **unempR:** The civilian unemployment rate.
11. **house_supply:** Monthly Supply of Houses in the United States, Months' Supply

I have divided the original dataset into the training and test sets, wherein the first 80 percent of observations belong to the training set, and the last 20 percent belong in the test set. After fitting the models in the training set, I measured their performance using two metrics in both the sets: mean absolute percent error and percent bias. The mean absolute percentage error (MAPE) is the absolute percent difference between the predicted and observed values and is useful when we compare forecasts of series in different scales. Here, we measure the error in terms of percentage.

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{|y_i|}$$

The percent bias calculates the the average amount by which the observed values deviates from the predicted values. Its value is close to 0 if the model is unbiased. From the bias-variance tradeoff, complex models have lower values of bias and higher variance, and vice-versa.

$$PB = \frac{1}{n} \sum_{i=1}^{n} \frac{y_i - \hat{y}_i}{y_i} \times 100\%$$

3.2: Summary of the Econometric Analysis

Before making any models, I had to ensure that all the variables in the time series are stationary. So, I have made the ACF and PACF plots to check for autocorrelation in all the series. To corroborate results from plots, I have conducted hypothesis tests that verifies whether the series are stationary. As all the variables have unit-root (are non-stationary), I differenced each of them to make them stationary. Thereafter, I created an ARIMA model using housing starts as the variable, and an ARIMAX variables which incorporates both exogenous stationary variables and endogenous variables (housing starts). I have also analyzed the diagnostic plots to check if the residuals are serially correlated via ACF, PACF, and Ljung Box Tests, and examined if the histograms of the residuals are skewed. The squared residuals of ARIMA model shows conditional heteroskedasticity, so I have further analyzed the variance of the series using ARCH and GARCH models. Once the assumptions are fulfilled, and forecast accuracy measured via error rates, I forecasted monthly values of housing starts using the ARIMAX, ARIMA and ARMA-GARCH

is trading at 3 percent, then the yield spread is 2 percent. It primarily depends on factors such as credit ratings, riskiness, and time to maturity.

model for the year 2019. The downside of an ARIMA model is that it only projects the future value based in previous values, devoid of external economic variables that can sufficiently influence the movement of the variable. Therefore, I made a VARX, which is a dynamic model. It involves variables that most impact housing starts model – mortgage rates, private houses completed, real estate loans, etc. Previously, I tested for cointegration in the variables, and built a vector error correction model, which elucidates how deviations from the long-run are corrected. Moreover, I constructed impulse response functions to depict how sensitive the variables are to shocks from another variable, and how long it takes to revert to its mean. Again, after ensuring that the errors are serially uncorrelated and checking the model diagnostics, I forecasted value of housing starts using the VARX model for one year. From chapter 3.2.1, I have described each method in detail.

3.2.1.Autocorrelation

Autocorrelation measures the linear relationship between lagged variables in a time series data. The ACF plot shows different autocorrelation coefficients. For example, r_1 measures the relationship between y_t and y_{t-1}. r_2 measures the relationship between y_t and y_{t-2}. and so on.

The PACF plot measure the relationship between y_t and y_{t-k} after removing the effects of lags $1, 2, ...(k-1)$. So, the first partial autocorrelation is identical to the first autocorrelation, because there is nothing between them to remove. In simple terms, the PACF removes the lags that cause autocorrelation.

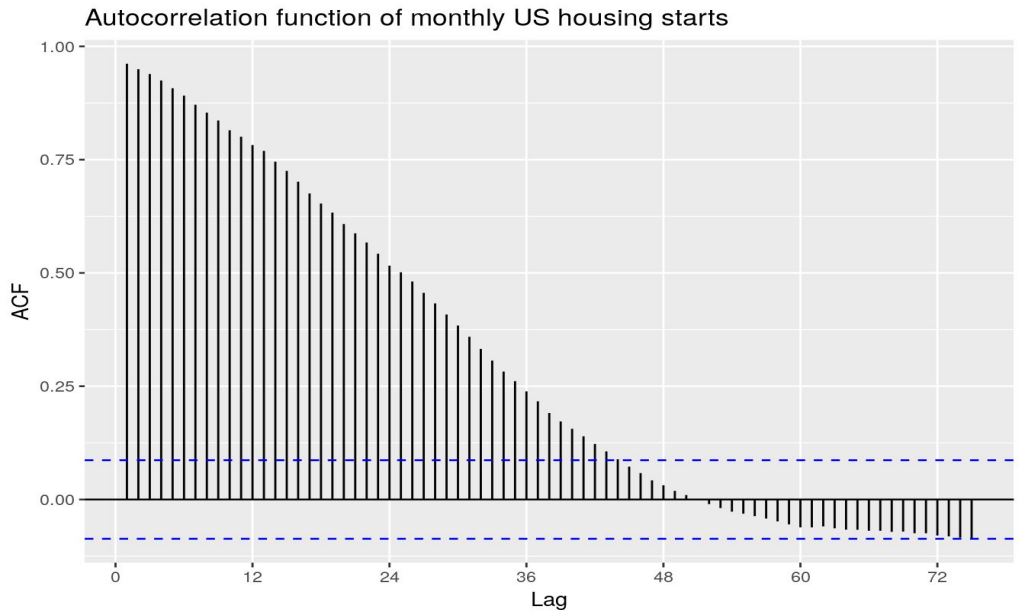


Figure 1: Plot of the Autocorrelation Function (ACF) of the US Housing Starts shows serial correlation as the ACF dies down

When a plot has trends, then the spikes in the ACF plot gradually decrease as lags increase. Because the housing starts series has autocorrelation, it is not white noise. The blue lines in the ACF and PACF plots indicate significance and the spikes in the plot that exceed the significance lines above and below imply that the current level of housing starts is significantly autocorrelated with its lagged values. From figure 3, we will include 3 lags. The number of significant correlations (spikes greater than than the significance lines) indicate the term p of the AR model.

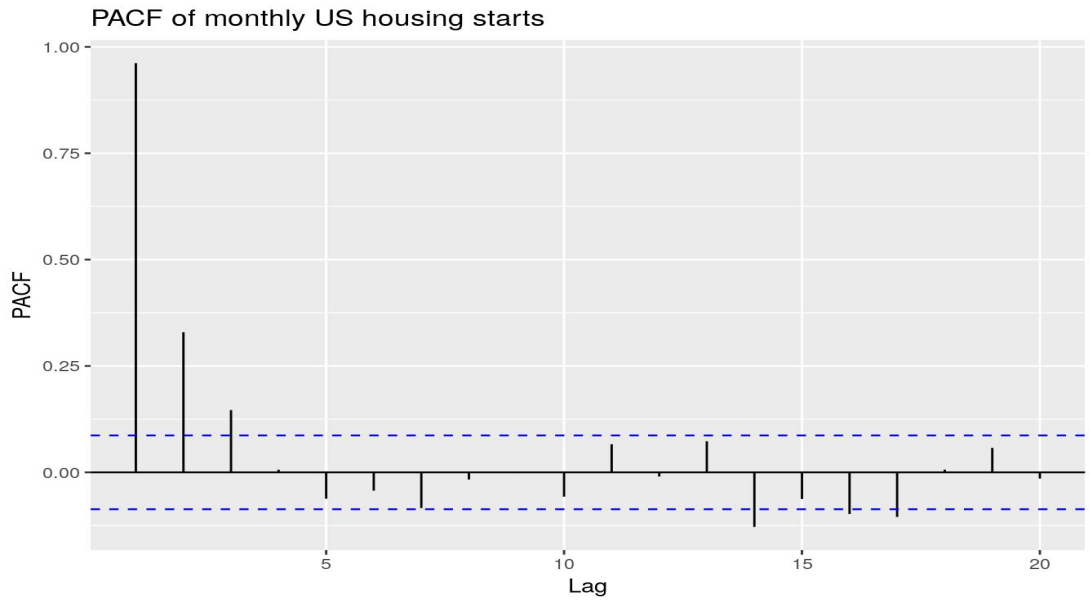


Figure 2: Partial Autocorrelation Function (ACF) of the US Housing Starts as PACF cuts off after 3 lags

A time series is stationary whose properties are independent of the time in which we observe the data. So, a series with trends or seasonality is non-stationary as trends and seasonality change the values of parameters at different points in time. Alternatively, a white noise series is stationary as it looks the same at any point in time. Generally, stationary time series have no predictable patterns in the long run. Such time plots will be approximately horizontal, have constant variance and mean, albeit they may be cyclical. The distribution of the time series should be constant between time t and time $t+n$.

A non-stationary time series can be made stationary by differencing consecutive observations. Logarithmic transformations and differencing can stabilize the variance and the mean of the time series, respectively. Furthermore, differencing eliminates seasonality and trends. The ACF plots are helpful to identify stationary time series. For such data, ACF plummets to zero fast. On the contrary, the ACF of a unit root or non-stationary series decreases relatively gradually, as also depicted by the spikes of the housing starts data. The value of r_1 is often large and positive for non-stationary data.

ARIMA models explain or capture serial correlation present within a time series. We test whether the first h autocorrelations are significantly different from we expect in a white noise process. The Ljung-Box test checks for autocorrelation where:

H_0 : the time series data points at each lag are i.i.d i.e. there is no autocorrelation,

H_A : the data points at each lag are not i.i.d. and are serially correlated.

The ACF of the differenced housing starts doesn't look like that of a white noise series. First differencing removed the trend in the residuals, resulting in uncorrelated errors. There are autocorrelations lying outside the 95% limits, and the Ljung-Box Q* statistic has a very small p-value of 3.892×10^{-11}. This suggests that the monthly values of the US housing starts are not random, but are correlated with those of the previous months. After differencing all the variables, I have made ARIMA models to forecast.

3.2.2 ARIMA Models

In ARIMA, the autoregressive (AR) is an extension of the concept of random walk used for non-stationary economic and financial data. The observations in random walk models can change unpredictably in any direction. Thus, the forecasts are equal to the last observation as the values are equally likely to move up or down. Let a time series be $\{\varepsilon_t : t = 1, ..., n\}$. If the elements of the series, ε_i, are independent and identically distributed (i.i.d.), with zero mean μ, variance σ^2, and no serial correlation $(cor(\varepsilon_i, \varepsilon_j) \neq 0, \ \forall \ i \neq j)$, then the time series is a discrete white noise (DWN). In particular, if the values w_i are drawn from a standard normal distribution (i.e. $\varepsilon_t \sim N(0, \sigma^2)$), then the series is known as Gaussian White Noise. In a random walk, each term, x_t depends entirely on the previous term, x_{t-1} and a stochastic white noise term, ε_t: $x_t = x_{t-1} + \varepsilon_t$

From the random walk models, we derive the AR model which is linearly dependent on the previous terms. A time series model, $\{x_t\}$ is an autoregressive model of order $p -$ $AR(p)$, if: $x_t = \alpha_1 x_{t-1} + ... + \alpha_p x_{t-p} + \varepsilon_t$, where x_t is a white noise. In $AR(1)$ models, the forecasts converge to the mean at a faster rate as the value of α_1 lowers. In $AR(2)$ models, if $\alpha_1{}^2 + 4\alpha_2 < 0$ is true, then the model displays a pseudo-periodic behavior, implying that the forecasts appear as stochastic cycles. If the condition fails, then the model does not follow a stochastic cycle and behaves similar to an $AR(1)$ model. [9]

The Moving Average (MA) in ARIMA is a linear combination of the past white noise terms. Intuitively, this means that the random white noise "shocks" are directly seen at each current value of the model. This is in contrast to an $AR(p)$ model, where the white noise "shocks" are only seen indirectly, via regression onto previous terms of the series. A time series model, x_t is a moving average model of order q, $MA(q)$, if:

$x_t = \varepsilon_t + \beta_t \varepsilon_{t-1} + ... + \beta_q \varepsilon_{t-q}$, where ε_t is a white noise.

[9] The above condition condition arose from the general solution to the homogeneous form of the linear, autonomous, second-order difference equation.

The error terms in an ARIMA model explain "short-term" influence of the past, and lagged terms explain "long-term" influence. The order of integration is another concept closely associated to stationarity. The order tells the number of times we should difference the series to make it stationary. An $I(0)$ series has order 0 if it does not require any differencing, and is already stationary. A series is order 1 or $I(1)$ if it is non-stationary initially, but the first difference makes it stationary. An $I(0)$ series frequently crosses the mean, whereas $I(1)$ and $I(2)$ series can stary or wander farther from their mean value and rarely comes across the mean. ARIMA repeatedly differences d times to make a stationary series.

A time series model, x_t is an autoregressive integrated moving average model of order p, d and q : $ARIMA(p, d, q)$ if:

$x'_t = \alpha_1 x'_{t-1} + \alpha_2 x'_{t-2} + ... + \varepsilon_t + \beta_1 \varepsilon_{t-1} + \beta_2 \varepsilon_{t-2} + ... + \beta_q \varepsilon_{t-q}$, such that:

x'_t is the differenced series where the differences could be more than 1,

ε_t is a white noise process with $E(\varepsilon_t) = 0$ and variance σ^2 .

Whilst the AR model uses its own previous behavior as inputs to capture market effects such as mean reversion in the housing price, the MA model shows the “shock” information to a series, such as the impact of a boom in housing construction or a steep drop in real estate loan rates. Combining autoregression with differencing and a moving average model yields a non-seasonal Autoregressive Integrated Moving Average (ARIMA) model. A series with $ARIMA(0, 0, 0)$ is a white noise series. Intuitively, ARIMA denotes the number of previous time steps the current value of our variable depends on. For example, at time *t*, the variable $hous\ st_t$ linearly depends on $hous\ st_{t-1}$ and $hous\ st_{t-2}$.

Figure 1 in the Appendix shows the time series of each economic variable. Graphically, all the predictors are non-stationary. The statistical hypothesis unit root tests, such as the Augmented Dickey Fuller (ADF) test checks whether the series requires differencing. The null hypothesis is that the series is unit root or non-stationarity. Applying a difference operator to a non-stationary or a random walk series x_t gives a stationary or a white noise ε_t series: $\Delta x_t = x_t - x_{t-1} = \varepsilon_t$. More precisely, in time series, the difference operator is known as a backshift operator, B where $Bx_t = x_t - x_{t-1}$. The null hypothesis is that $\alpha = 1$, while the alternative hypothesis is that $\alpha < 1$. In this test,

H_O : the series is not stationary,

H_A : the series is stationary.

We look for evidence that the null hypothesis is false. Consequently, small p-values (e.g., > 0.05) suggest that differencing is required. In the housing starts series, the p-value of 0.3791 is much bigger than the 5% critical value, so we fail to reject the null hypothesis. That is, the data is not stationary. Occasionally, the differenced data will not appear to be stationary and it may be necessary to difference the data a second time to obtain a stationary series. Thus, differencing the series twice (2nd difference), and applying the test again yields a small p-value of 0.01. Now the differenced housing starts series is stationary. Likewise, after differencing all the variables, the p-values from the Augmented Dickey-Fuller test are less than 0.01. Surprisingly, the CPI series is stationary after differencing upto 3 times. From table 1 below, the variables − income, securitized consumer loans, and real estate loans are twice differenced. The backshift operator, B, represents the second order differencing as :

$$B^2 x_t = B(B\, x_t) = B(x_t - x_{t-1}) = Bx_t - Bx_{t-1} = (x_t - x_{t-1}) - (x_{t-1} - x_{t-2}) = x_t - 2x_{t-1} + x_{t-2}$$

Variable	P-value before differencing	Number of differences
Housing Starts	0.3790981	1
Income	0.6345780	2
Federal Funds Rate	0.1145223	1
Yield Spread	0.2567	1
Securitized consumer loans	0.9705642	2
Unemployment rate	0.0859251	1
CPI	0.2545744	3
Private houses completed	0.7298269	1
Mortgage rate	0.0702159	1
Real estate loans	0.6012911	2
Housing supply	0.3389323	1

Table 1: The number of differences required for each variable to be stationary

I have used the stationary variables in modelling ARIMAX models in the next chapter.

3.2.3.ARIMAX Model

The standard ARIMA models forecast solely based on the past values of the housing starts, and does not have covariates. The model assumes that the future values are linearly dependent on the past values and previous stochastic shocks. Similar to ARIMA, a multivariate regression model is the ARIMAX model wherein the covariates − mortgage rate, first difference of private houses completed, and second difference of income, securitized consumer loans and real estate loans are present on the right hand side of the model. The $ARIMAX(2,1,3)$ model follows an $ARMA(2,3)$, with a first order difference or backshift:

$B\ hous\ st_t = hous\ st_t - hous\ st_{t-1}$. This gives a model with covariates at time t and their coefficients:

Regression with $ARIMAX\ (2,1,3)$:

$$B\ hous\ st_t = -48.1082\ mortgR_t - 0.0054\ income.d2_t + 1.2251\ sec\ conL.d2_t - 7.1930\ CPI.d3_t + 0.1023\ pvt\ house\ comp.d1_t + 0.0096\ real\ estL.d2_t + \varepsilon_t,$$ where:

$$\varepsilon_t = 1.5159\ B\ hous\ st_{t-1} - 0.8454\ B\ hous\ st_{t-2} - 1.9201\varepsilon_{t-1} + 1.4622\ \varepsilon_{t-2} - 0.3158\ \varepsilon_{t-3}$$

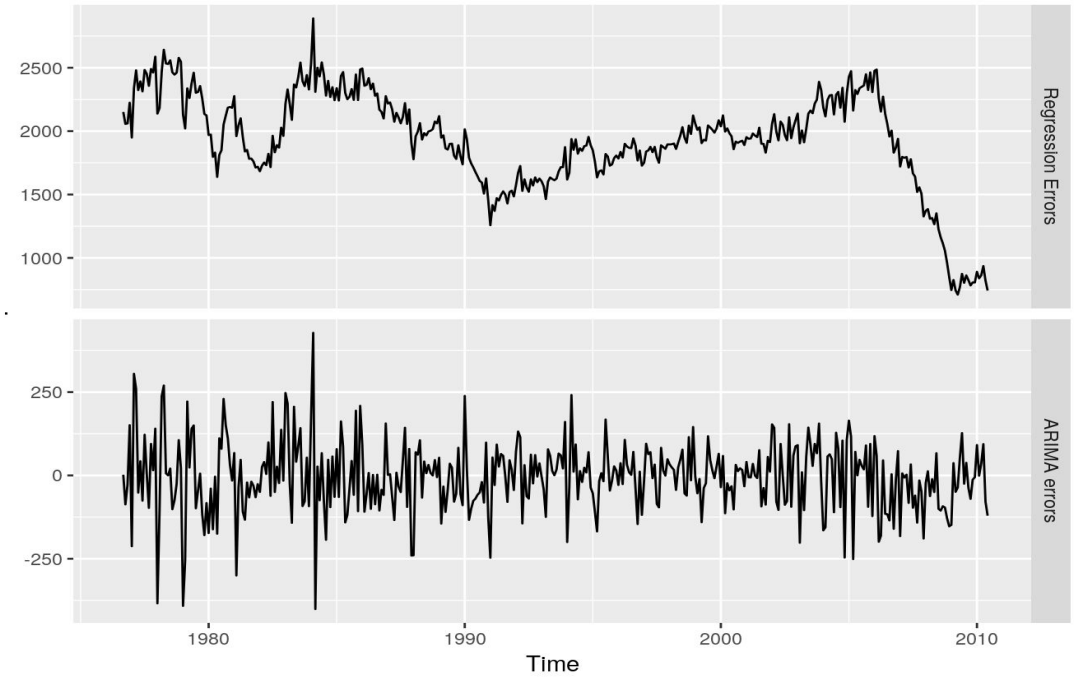


Figure 3 : The plots show errors from regression and ARIMA models. The ARIMA errors resemble white noise.

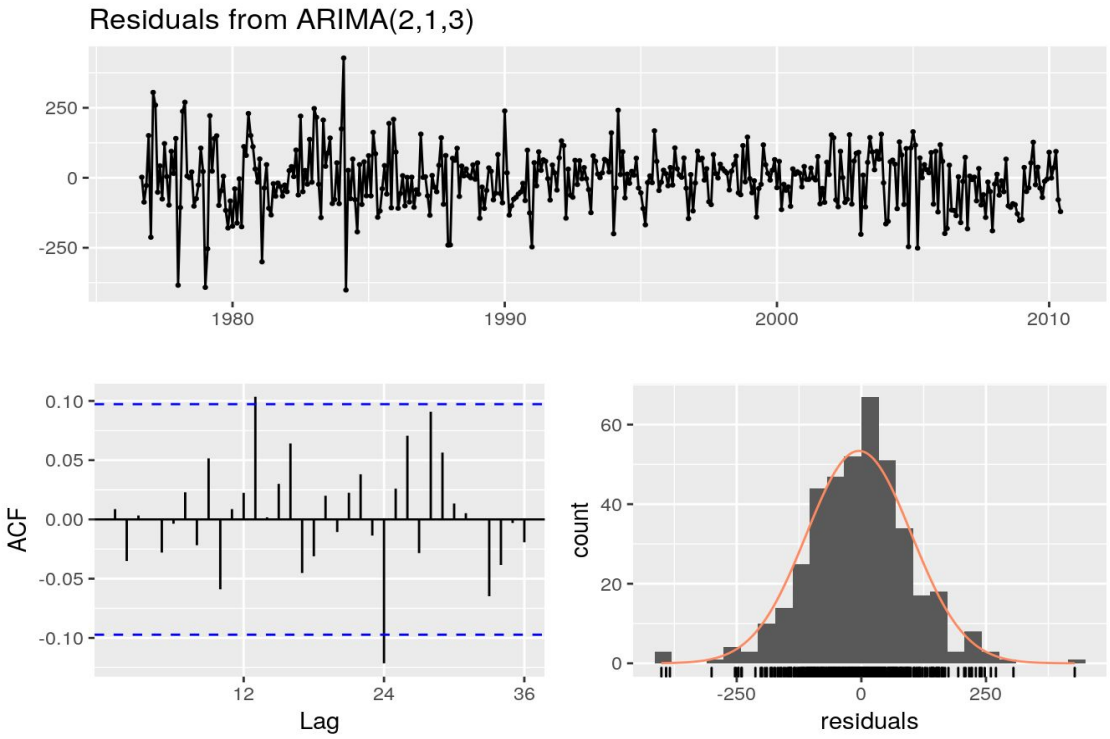


Figure 4: The diagnostic plots show the ACF and histogram of residuals from ARIMA(2,1,3)

From the Ljung-Box test on the data comprising the residuals from regression with $ARIMAX(2,1,3)$, the p-value of $0.1069 > 0.05$, implying that the model is not serially correlated. The time plot and histogram of the residuals shows that the variance in the residuals are almost constant. The MAPE is 0.357 and the percent bias is 0.3439.

Unlike the ARIMAX model which contains exogenous variables, the $ARIMA(3,1,2)$ predicts the endogenous variable − housing starts using only its lags and the model is:[10]

$$B\ hous\ st_t = 1.0488\ B\ hous\ st_{t-1} - 0.4336\ B\ hous\ st_{t-2} - 0.1437\ B\ hous\ st_{t-3} - 1.427\ \varepsilon_{t-1} + 0.8473\ \varepsilon_{t-2} + \varepsilon_t,$$

where ε_t is white noise. The mean absolute percentage error (MAPE) is 0.3922 and the percent bias is 0.3885. Both the error rates are higher than those of $ARIMAX(2,1,3)$.

ARIMA models are athoretic, and therefore cannot be interpreted the same as those of regression coefficients. As there are three lagged terms of housing starts in the model, its current value depends of on

[10] In R, I had looped over several combinations of p,d and q and stored the fitted model of ARIMA(p,d,q). Given any order of ARIMA, if the current AIC value is less than the previously generated AIC, then the current AIC is the final AIC and that order is chosen. After terminating the loop, the order obtained was ARIMA(3,1,2).

the value in the all previous four months combined. Figure 5 below shows the plot of residuals of the model to check if the series is discrete white noise (DWN):[11]

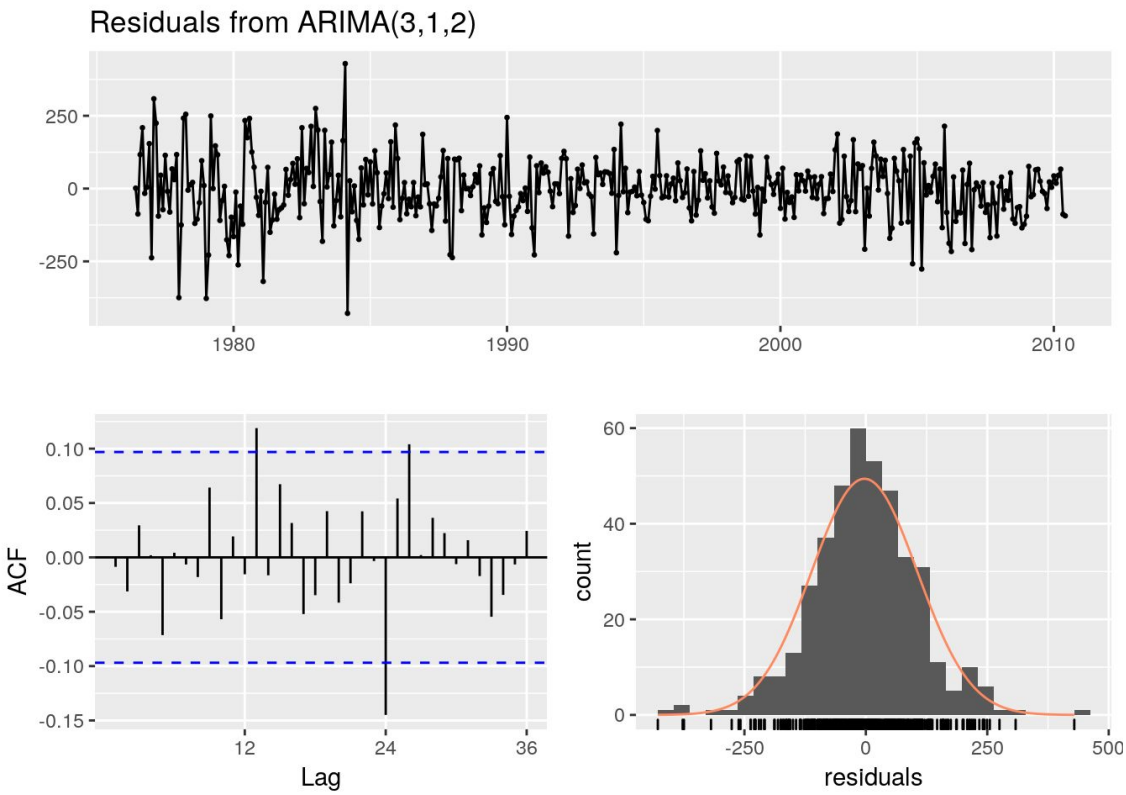


Figure 5: The ACF plot of the residuals of ARIMA(3,1,2) shows that the errors are not serially correlated.

The correlogram seems to follow discrete white noise. The p-value of $0.07732 > 0.05$, fails to reject the null hypothesis that the residuals of $ARIMA(3,1,2)$ is not serially correlated i.e. the series is white noise. Next, I will explain how to mathematically obtain forecasts from an ARIMA model. In order to forecast, we have to rewrite the $ARIMA(p,d,q)$ using the backshift notation as:

$$(1-\alpha_1 B - ..... - \alpha_p B^p)(1-B^p)\, x_t = (1+\beta_1 B + ... + \beta_q B^q)\, \varepsilon_t$$

$$\Downarrow \qquad \Downarrow \qquad \Downarrow$$

$AR(p)$ $\quad$ d differences $\quad$ $MA(q)$

[11] A good forecasting method will yield uncorrelated residuals. If there are correlations between residuals, then there is information left in the residuals which should be used in computing forecasts. The residuals should also have zero mean, otherwise the forecasts are biased. Additionally, residuals should also have constant variance and be normally distributed.

The $ARIMA(3,1,2)$ is:

$$(1 - 1.0488\,B + 0.4336\,B^2 + 0.1437\,B^3)\,(1 - B)\,hous\,st_t = (1 - 1.472\,B + 0.8473\,B^2)\,\varepsilon_t$$

I have expanded the left hand side, keeping the right hand side constant:

$$[1 - (1 + 1.0488)\,B + (1.0488 + 0.4366)\,B^2 + (-0.4336 + 0.143)\,B^3 - 0.143\,B^4]\,hous\,st$$

Now, I have applied the backshift operator on both sides:

$$hous\,st_t - (1 + 1.0488)\,hous\,st_{t-1} + (1.0488 + 0.4366)\,hous\,st_{t-2} + (-0.4336 + 0.143)\,hous\,st_{t-3} - 0.143\;hous\,st_{t-4}$$

$$= \varepsilon_t - 1.472\,\varepsilon_{t-1} + 0.8473\,\varepsilon_{t-2}$$

In the next step, I have moved all the terms, except $hous\,st_t$ to the right hand side:[12]

$$hous\,st_t = 2.0488\,hous\,st_{t-1} - 1.4854\,hous\,st_{t-2} + 0.2906\,hous\,st_{t-3} + 0.143\;hous\,st_{t-4} + \varepsilon_t - 1.472\,\varepsilon_{t-1} + 0.8473\,\varepsilon_{t-2}$$

To get one step ahead forecast for the next month: $hous\,st_{T+1}$, I have replaced t with $T + 1$:

$$hous\,st_{T+1/T} = 2.0488\,hous\,st_T - 1.4854\,hous\,st_{T-1} + 0.2906\,hous\,st_{T-2} + 0.143\;hous\,st_{T-3} - 1.472\,\varepsilon_T$$

To get two step ahead forecast for the following second month: $hous\,st_{T+2}$, I have replaced t with $T + 2$:

$$hous\,st_{T+2/T} = 2.0488\,hous\,st_{T+1/T} - 1.4854\,hous\,st_T + 0.2906\,hous\,st_{T-1} + 0.143\;hous\,st_{T-2}$$

This process continues for all future monthly values. In order to better understand the dynamics of the system, I have depicted the forecasts in the results (Table 9) and graphed them (Figure 4 in the Appendix). Another approach would be to observe the time paths or impulse response functions related to the system (discussed in 3.2.10).

3.2.4. ARCH and GARCH models

The two main drawbacks of ARIMA models are − firstly, they do not consider volatility clustering i.e. they are not conditionally heteroskedastic. Secondly, because ARIMA linearly models the data, the forecast width is constant as the model does not incorporate new information or recent changes. Hence, we need to use the Autoregressive Conditional Heteroskedastic (ARCH) model and Generalised Autoregressive Conditional Heteroskedastic (GARCH) model. OLS models assume that the expected value of the squared error terms are constant over time i.e. errors are homoskedastic. (G)ARCH models

[12] Albeit the above equation looks like an ARIMA(4,0,2) model , it is not ARIMA(4,0,2) as it does not satisfy the stationarity conditions. It is still an ARIMA(3,1,2).

emphasize on this assumption. Data suffers from heteroskedasticity when variances of error terms are larger and smaller in other range of data points. Rather than correcting the problem, (G)ARCH model heteroscedastic variances.

Different forms of volatility such as sell-offs during a financial crises, can cause serially correlated variances which produce heteroscedasticity. A univariate $GARCH(1,1)$ helps in modeling volatility, its clustering and forecasting. In financial time series, we cannot assume that the variance or the volatility is constant as some periods are more volatile than others. When the volatility of a time series changes over time, this technical behavior of volatility clustering is called conditional heteroskedasticity. Further, if the conditional variance is conditional on information of the variable itself (housing starts), its past shocks and previous values of conditional variance, then the data is conditionally heteroskedastic.

The ACF and PACF of residuals can confirm that if the residuals are not white noise, they can be predicted. Residuals of strict white noise series are i.i.d normally distributed with zero mean. Moreover, the PACF and ACF of squared residuals in figure 6 have no significant lags. Finally, we cannot predict a strict white noise series, either linearly or nonlinearly. The squared residuals of $ARIMA(3,1,2)$ model show a cluster of volatility or conditional heteroskedasticity as shown from the ACF plots.

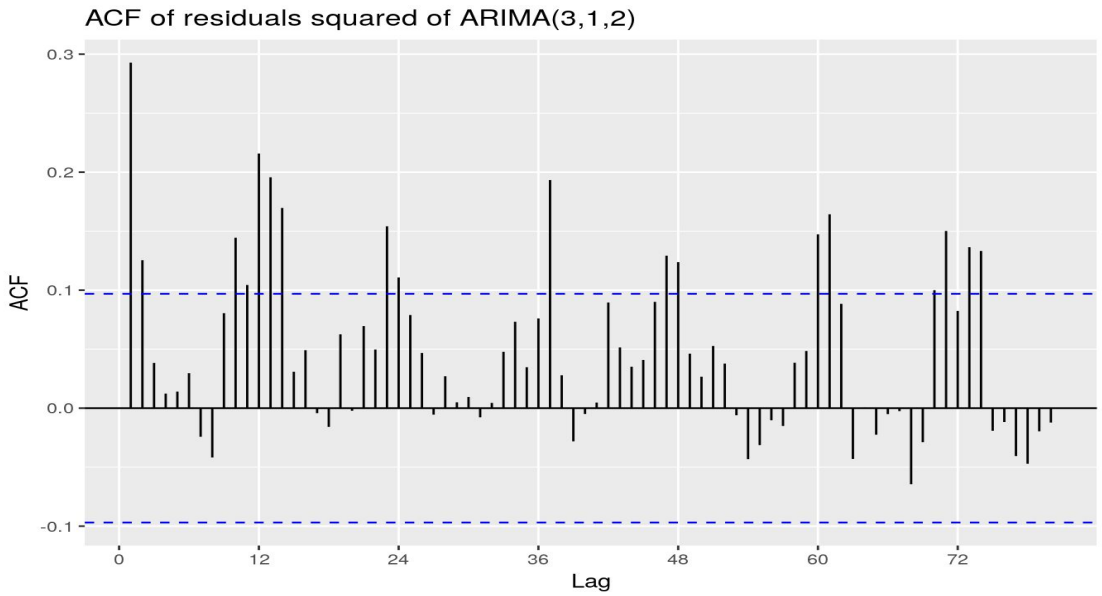


Figure 6: The ACF plot of the squared residuals of ARIMA(3,1,2) indicate that volatility is clustered together in different time periods.

The squared residuals from $ARIMA(3,1,2)$ are autocorrelated as they do not randomly occur in time, implying that the time series exhibits conditional heteroskedasticity. Also, the conditional heteroskedastic series is non-stationary. This is also called volatility clustering as periods of high variance tend to group up together. The Ljung-Box test on the squared residuals yields a p-value of $2.623 \times 10^{8} < 0.05$, which rejects the null hypothesis of no serial correlation. To further prove the presence of conditional heteroskedasticity, I have performed the Engle's ARCH-LM test with q lags. This checks for the presence of ARCH effects at lags 1 to q. The general procedure to model an ARCH process is as follows:

Let ε_t is the error term of a model which is normally distributed with zero mean and conditional variance of the error term is $\sigma^2{}_t = Var(\varepsilon_t / \varepsilon_{t-1}, \varepsilon_{t-2}, ..., \varepsilon_{t-q})$. In other words,

$$\varepsilon_t \sim N(0, \alpha_0 + \sum_{i=1}^{q} \alpha_i \varepsilon^2{}_{t-i})$$

The $ARCH(q)$ model for the conditional variance is:

$$\sigma^2{}_t = \alpha_0 + \alpha_1 \varepsilon^2{}_{t-1} + ... + \alpha_q \varepsilon^2{}_{t-q} = \alpha_0 + \sum_{i=1}^{q} \alpha_i \varepsilon^2{}_{t-i}, \quad \text{where} \quad \alpha_0 > 0 \text{ and } \alpha_i \geq 0, \ \forall \ i = 1, ..., q \text{ to avoid}$$

negative variances.

Subsequently, we estimate the best fit $AR(q)$ model, obtain the squared errors and regress them with q lagged values of the errors and a constant shown in the above equation. The null hypothesis is that no ARCH effects are present, and the alternate hypothesis is that at least one of α_i must be significant. We test for ARCH effects jointly at different values of lags: $H_0 = \alpha_1 = \alpha_2 = ... = \alpha_q = 0$. As the p-values with varying values of $q = 2, 4, 8, 12,$ are very small, we reject the null hypothesis to conclude that ARCH effects are present.

GARCH is an extension of ARCH where $\sigma^2{}_t$ not only depends on the lags of the squared error terms but also on its own lags. In contrast to ARCH which involves only the most recent return, GARCH boosts the forecasting accuracy by assigning lesser weights β_i to past returns corresponding to more distant volatilities. The model is:

$$\sigma^2{}_t = \alpha_0 + \alpha_1 \varepsilon^2{}_{t-1} + ... + \alpha_q \varepsilon^2{}_{t-q} + \beta_1 \sigma^2{}_{t-1} + ... + \beta_q \sigma^2{}_{t-q}$$

The method of maximum likelihood estimates most (G)ARCH models, such as measuring relative loss or profit from trading stocks in a day. α and β measure the ARCH and GARCH effect, respectively. The GARCH models assume that the conditional mean of the time series is zero, which in reality is not always true. We can supplement the conditional variance structure of GARCH by a conditional mean that is

modeled by an ARMA model. By observing the time series, we can identify the ARMA order. Likewise, the squared residuals from the fitted ARMA model helps to identify the GARCH order. Then, we maximize the log-likelihood function for ARMA + GARCH model to obtain the maximum likelihood estimation for the model.

I have fit GARCH model(s), starting with a $GARCH(1,1)$ model with Gaussian innovations.[13] $GARCH(1,1)$ considers a single autoregressive and a moving average lag, i.e. $ARMA(1,1)$. The model predicts the variance by taking the weighted average of the long term historical variance, the predicted variance at time t, and the predicted variance of the squared residuals at time $(t-1)$. An $ARMA(1,1)-GARCH(1,1)$ model is :

$$y_t = \mu + \theta_1 y_{t-1} + \gamma_1 \varepsilon_{t-1} + \varepsilon_t$$

$$\varepsilon_t = \sigma_t z_t, \ where\ z_t \sim N(0,1)\ i.i.d., \ and\ \sigma^2{}_t = \alpha_0 + \alpha_1 \varepsilon^2{}_{t-1} + \beta_1 \sigma^2{}_{t-1}, \ \alpha > 0, \ \beta \geq 0.$$

α, β *and* z are constants that the model estimates using maximum likelihood and ε_{t-1} is the squared residual at time $(t-1)$. θ_n and γ_n measure $AR(n)$ and $MA(n)$, respectively. In the GARCH model, the persistence explains the rate at which volatility decays after applying a shock. α_1 measures the extent to which a volatility shock today feeds through into next period's volatility and $\alpha_1 + \beta_1$ measures the rate at which this effect dies over time. Large values of β_1 means that large changes in the volatility will affect future volatilities for a long period of time since the decay is slower. [14]

Ideally, $\alpha_1 + \beta_1 < 1$

If $\alpha_1 + \beta_1 > 1$, the series will become unstable and the volatility forecasts are explosive.

If $\alpha_1 + \beta_1 = 1$, the model has an exponential decay and persistence of volatility occurs.

The $ARMA(2,2)-GARCH(1,1)$ model for housing starts is:[15]

---

[13] The gaussian distribution is the same as normal distribution.

[14] Source: https://newonlinecourses.science.psu.edu/stat510/node/85/

[15] The order of ARMA - GARCH should be simultaneously determined, not separately. When the ARMA - GARCH models estimate the process well, estimates will be inconsistent if we consider the conditional mean (ARMA) model only and neglect the conditional variance, (GARCH) model as this implicitly assumes that the conditional variance is constant, and vice-versa. Only under certain conditions, such as when MA(0), we can estimate the order of ARMA independent from GARCH and also generate consistent estimates.

Unfortunately, it is difficult to jointly estimate the orders of ARMA - GARCH. The reasonable approach is to experiment with a few candidates models in the training set, pick the model with the lowest prediction rate, and use the selected model to predict in the test set. Ensuring that the prediction rate is lowest does not necessarily

$hous\,st_t = 0.6152\,hous\,st_{t-1} + 0.3685\,hous\,st_{t-2} - 0.4984\,\varepsilon_{t-1} + 0.1791\,\varepsilon_{t-2}$

$\sigma^2{}_t = 2027.8896 + 0.1277\ \varepsilon^2{}_{t-1} + 0.4095\,\sigma^2{}_{t-1}$, where $\varepsilon_t \sim iidN(0,1)$.

$AR(1)$ and $AR(2)$ are statistically insignificant, implying that no autocorrelation exists. $\beta_1 = 0.4094$ is also insignificant, which means there is no persistent volatility clustering. To check for normality, we have to observe the distribution of the residuals.

H_0 = white noise innovation process ε_t is Gaussian.

H_A = white noise innovation process ε_t is not Gaussian.

The standardized residual tests in table 2 such as Jarque-Bera Test and the Shapiro-Wilk tests suggest that the data is normally distributed. From the Ljung-Box Test, the distribution of residuals and squared residuals is Gaussian as the p-value is greater than the 5 percent significance level at lags 10, 15 and 20. Similarly, the model has no $ARCH$ effects as the p-value of the $ARCH-LM$ is very high. The table below shows the different residual tests and the p-values:

Residual Test		**Statistic**	**p-value**
Jarque-Bera Test	Residual	χ^2 (chi square)	0.6927309
Shapiro-Wilk Test	Residual	W	0.9463657
Ljung-Box Test	Residual	$Q(10)$	0.1138027
Ljung-Box Test	Residual	$Q(15)$	0.05391022
Ljung-Box Test	Residual	$Q(20)$	0.06944194
Ljung-Box Test	Residual Squared	$Q(10)$	0.2186401
Ljung-Box Test	Residual Squared	$Q(15)$	0.1054082
Ljung-Box Test	Residual Squared	$Q(20)$	0.1093043
Arch-LM Test	Residual	TR^2	0.1709148

guarantee that the model is suitable. We also have to check if the residuals and the squared residuals have ARCH effects, autocorrelation and normal distribution. For instance, ARMA(2,1) - GARCH(1,1) resulted in the lowest MAPE of 0.30, but the residuals failed the diagnostic tests, so I eliminated it.

Table 2: Standardized residual tests and the p-values. All the p-values are significant at 5 percent.

The qq-plot of the standardised residuals, suggests that the fitted standardised skew-t conditional distribution is good. Using the above model, I have forecasted housing starts for the year 2019 which is shown in chapter 4: Results.

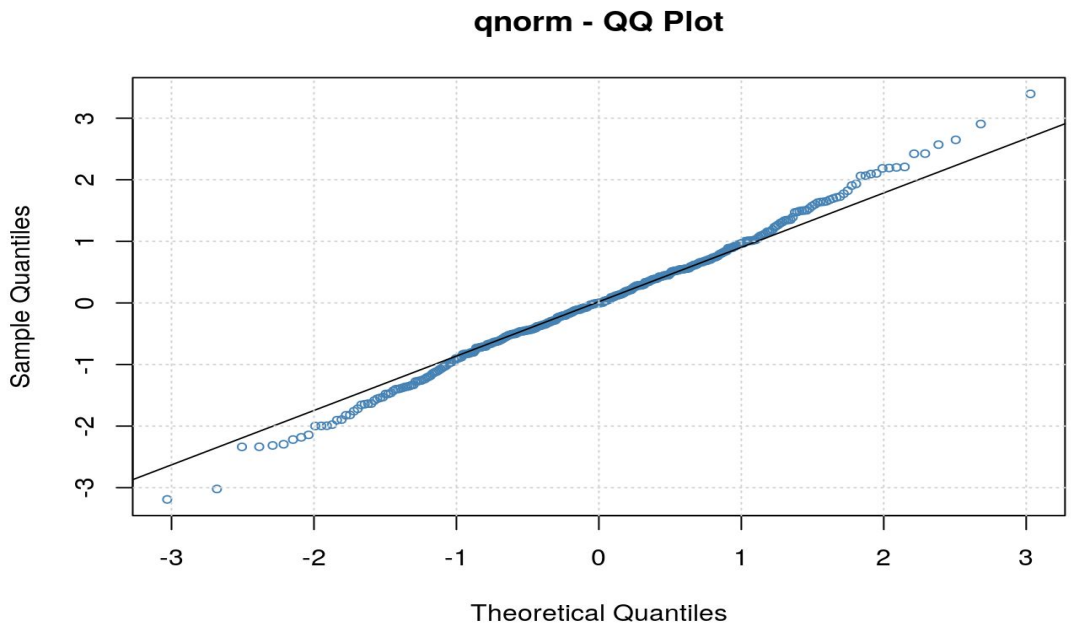


Figure 7: The plot shows the quantiles of the residuals. Since most of the points lie along the line, the distribution has the same shape as that of the theoretical distribution.

Until now, I had used some of the simpler forecasting techniques that primarily involved just the response variable: housing starts. I have explored how we can expand ARIMA models by adding exogenous variables and their lagged values to forecast housing starts. Now, I will discuss some of the advanced methods that necessitates me to check first if the explanatory variables have common stochastic trends, or are cointegrated.

3.2.5 Cointegration

When the trends and patterns of two series are similar, then they are cointegrated. The cointegration test measures whether the residuals from a regression are stationary. Stationary residuals are cointegrated. It is also a Dickey-Fuller stationarity test on residuals where the null hypothesis is that the series are not cointegrated. For a stationary test, we should reject the null hypothesis of no cointegration. The concept of

cointegrated time series arises from the idea that housing prices, securities' prices, interest rates and other economic indicators return to their long-term average levels after significant movements in short terms (mean reversion). Besides the imbalance in the demand and supply of houses, prices revert to their means as housing prices are highly correlated with inflation. Further, inflation rates are highly correlated with wages or real disposable income.

Given two series x_t and y_t , we search for parameters $\alpha, \beta,$ and ϱ such that $y_t = \alpha + \beta\, x_t + r_t = \rho\, r_{t-1} + \varepsilon_t,$

where $r_t =$ residual and, $\varepsilon_t =$ series of independently and identically distributed (i.i.d) innovations[16] with mean = 0

If $|\rho| < 1,$ then x_t and y_t are cointegrated (i.e., r_t does not contain a unit root). If $|\rho| = 1,$ then the residual series r_t has a unit root and follows a random walk. Below I have constructed an Engle Granger cointegration model using the variables private house completed and housing starts as explanatory and response variables, respectively. Both series have unit root as they are $I(1)$. [17]If they cointegrate, then the coefficients α and β will define an equilibrium.

$$hous\ st_t = 66.8147 + 0.9736\ pvt\ house\ comp_t + r_t,\quad r_t = 0.7482\ r_{t-1} + \varepsilon_t$$

Figure 2 in the Appendix shows the plots of the equation, residuals and innovations. Here, the series seems cointegrated but the residuals are not $AR(1)$. From the above regression, I tested if the residuals had unit root. As the p-values (shown below) are very small for all the tests, we reject the null hypothesis that the residuals are unit root, implying the series: housing starts and private houses completed are cointegrated. In fact, no other variable is cointegrated with housing starts.

Test	**p-value**
Augmented Dickey Fuller	0.00583

[16] In time series, the term *innovations* is used the same way as the term *errors* is used in a cross-sectional analysis. So, we can interchangeably use them. Errors are called innovations in time series because the errors contain new information in the system. The same errors are not considered new in a cross-sectional analysis as the observations are not chronologically arranged in a sequence. For instance, the tenth observation in a cross sectional data is neither newer nor older than the ninth observation. However, in a time series, the tenth observation comes after ninth, hence, the error or innovation has new information.

[17] If one of the series is stationary i.e. $I(0)$ and the other one is $I(1)$, the series cannot be cointegrated. This is because cointegrated series have common stochastic trends or same long run behavior. The linear combination of the cointegrated series cancels out the common stochastic trends, creating a stationary linear relationship. The OLS estimates of cointegrated variables will be very consistent.

Phillips Perron	0.0001
Schmidt and Phillips Rho	0.0001
Johansen Trace Test	0.0001

Table 3: Unit root test for stationarity of residuals from cointegration

3.2.6. Error Correction Model (ECM)

A drawback of Error-Granger statistic regression is that it has a small bias, and we cannot infer if the estimated parameters in the regression are significant as the distribution depends upon unknown parameters. To avoid this problem, I have built an error correction model that investigates the long-run relationship between the two series. The variables without cointegration $I(0)$ have a short term relationship as opposed to those variables with cointegration $I(1)$, as the latter have a long term relationship. These theoretically-driven models estimate both short and long term effects of one time series on another. For instance, if the first lagged value in a series deviates from its long run equilibrium, how does this error influence the short run dynamics? So, the ECMs measure the rate at which the variable in question returns to its equilibrium after other variables change. The error correction is $\varepsilon_t = y_t - \beta\, x_t$, where β is the cointegrating coefficient and ε_t is the error from regressing y_t on x_t. We define ECM as: $\Delta y_t = \alpha\, \varepsilon_{t-1} + \gamma\, \Delta x_t + u_t$.

This equation states that the ε_{t-1} and Δx_t can explain change in y_t. ε_{t-1} is the disequilibrium term or the equilibrium error that occured in the previous period. The model is in the state of disequilibrium if $\varepsilon_{t-1} \neq 0$, and vice-versa. When $x_t = 0,$ and $\varepsilon_{t-1} > 0,$ y_{t-1} is high above its equilibrium, so Δy_t should be negative to restore the equilibrium value. Intuitively, this signifies that the error correction coefficient α should be negative for the ECM equation to be stable dynamically. Therefore, if y_{t-1} is above equilibrium, then it will decline in the next period, which will correct the error or disequilibrium. In this case, the ECM is:

$$\Delta\, hous\, st_t = -0.25682\, \varepsilon_{t-1} + 0.24530\, \Delta\, pvt\, house\, comp_t + u_t$$

From the cointegration part and the error correction term, $\beta = 0.9736$ is the long run parameter, while $\alpha = -0.25682$ and $\gamma = 0.24530$ are the short-run parameters. So far, I have only used one explanatory

variable, but to find the number of cointegrating relations when there are more than two $I(1)$ series, we use the Johansen Test and a Vector Autoregressive Model (VAR).

3.2.7. The Johansen Test for Cointegration

This test measures the number of cointegrating relations present out of n integrated time series. Cointegrated series have at least one common trend among the variables. Then we test if the linear combination of underlying series forms a stationary series. If the series $x_1, ..., x_n$ at time t are individually integrated, $I(1)$, then the linear combination, $y = \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$ at time t is stationary: $I(0)$, and the variables are cointegrated. If the linear combination is not $I(0)$, then the variables together are not cointegrated, and we don't have to construct a vector error correction model. The Johansen test is composed of two tests: trace test and maximum eigenvalue test.

The trace test examines that there are r number of cointegrating vectors such that $r < n$.

$H_0 : r < n$

$H_A : r = n$

The test sequentially proceeds for $r = 1, 2, 3, ..., n$, and the first value of r_i that the trace test fails to reject the null hypothesis, is the estimate of r. In essence, the trace test measures if there is at least one linear combination of explanatory variables that makes the process stationary. The "maximum eigenvalue " test is similar to the trace test with a slight difference in the alternative hypothesis. It ensures that the number of linear combinations does not equal to the number of explanatory variables $(r \neq n)$. If $r = n$, the all the explanatory variables are stationary and not cointegrated. The test also proceeds sequentially and the hypotheses are:

$H_0 : r < n$

$H_A : r = r + 1$

The Johansen test checks if $r = 0$ *or* 1. $r = n - 1$, where n is the number of time series under test.

$H_0 = 0$ means that no cointegration is present. When rank $r > 0$, there is a cointegrating relationship between at least two time series. The eigenvalue decomposition outputs a set of eigenvectors. The

components of the largest eigenvector is used in formulating the coefficients of the linear combination of time series. This creates stationarity.

In this case, I have tested cointegration between variables housing starts, private houses completed, housing supply and real estate loans. The largest eigenvalue generated by the test is 0.1506. Next, the output shows the trace test statistic for the four hypotheses of $r = 0$ *to* $r \leq 3$. From $r = 0$ to $r \leq 2$, the test statistic exceeds the 0.05 significance level. For instance, when $r = 0$, $133.69 > 48.28$. Similarly, in the second test we test the null hypothesis for $r \leq 1$ against the alternative hypothesis of $r > 1$. As $50.59 > 31.52$, we reject $r \leq 1$, i.e. the null hypothesis of no cointegration. However, when $r \leq 2$, we fail to reject the null as $12.87 < 17.95$. Thus, the matrix' rank is 2 and the series will become stationary after using a linear combination of two time series.

r = rank	**Test statistic**	**10 % level**	**5 % level**	**1 % level**
$r \leq 3$	0.65	6.50	8.18	11.65
$r \leq 2$	12.87	15.66	17.95	23.52
$r \leq 1$	50.59	28.71	31.52	37.22
$r = 0$	133.69	45.23	48.28	55.43

Table 4: Values of trace statistics at significance levels for the hypotheses

The linear combination by using components of eigenvectors associated with the largest eigenvalue of 0.1506 results in the following stationary series:

$$linear\ series\ =\ 1.00\ house\ st\ -0.9398363\ pvt\ house\ comp\ .$$

Moreover, the p-value in the Dickey-Fuller test is $0.01 < 0.05$. So, we reject the null hypothesis of unit root and conclude that the series formed from the linear combination is stationary. [18]

[18] Source: https://www.quantstart.com/articles/Johansen-Test-for-Cointegrating-Time-Series-Analysis-in-R

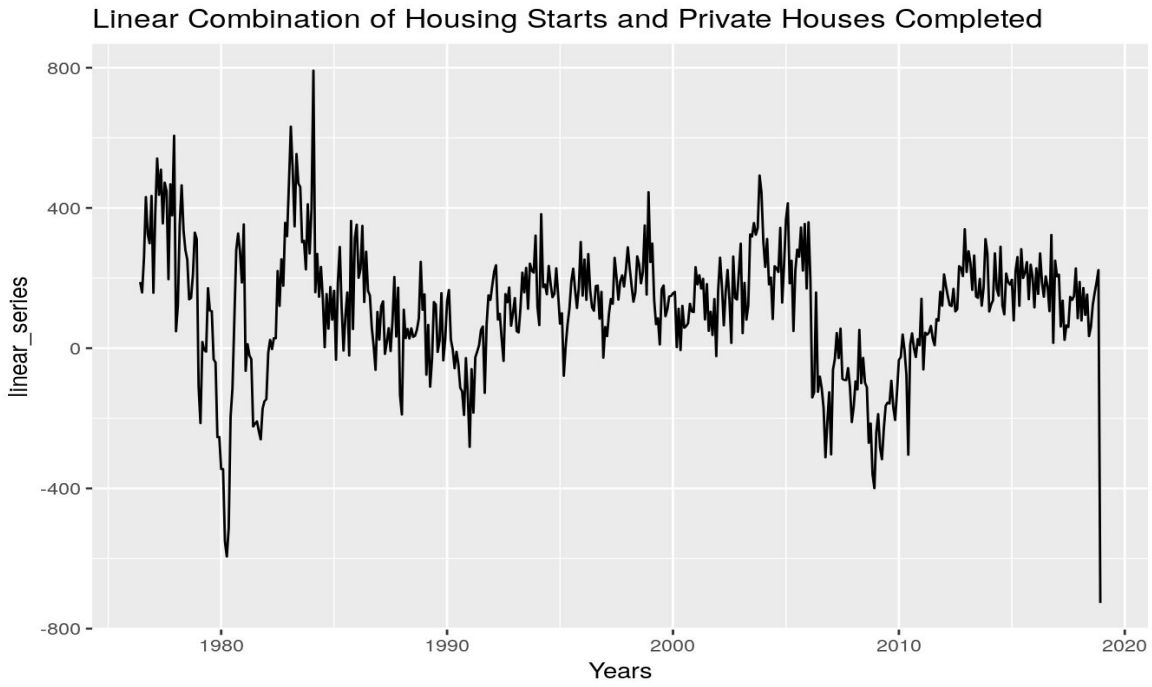


Figure 8: The plot shows the linear combination of two series as there are two cointegrating relations.

Having obtained a linear combination of cointegrated series, I have made a Vector Error Correction Model to model the cointegrated time series.

3.2.8. Vector Error Correction Model (VECM)

Unlike the error correction model which is a single equation, a VECM is a multiple equation model used when series are non-stationary and cointegrated. The concept of error correction helps to understand how deviations from the long-run are "corrected". We also make VECM when variables have at least one cointegrating relations.

Analogous to coefficients in a regression model, the coefficients of ECT in a VECM quantifies the effect of the error correction term on a specific dependent variable. The sign shows whether error is corrected or inflated. In the former case, the variable approaches equilibrium; while in the latter case, the error is inflated i.e. it further deviates from the equilibrium. If the coefficients of certain variables are indistinguishable from zero, then those variables are considered "dominant" as they do not adjust towards equilibrium, but they "drive" the system of variables. Other variables that do adjust are considered "dominated", and if no variables adjust towards equilibrium, then they are not cointegrated.

Mathematically, suppose there are two variables in a system with no lagged difference terms. It has one cointegrating equation which is: $y_{2,1} = \beta y_{1,t}$

Its VEC model is:

$$\Delta y_{1,t} = \alpha_1(y_{2,t-1} - \beta y_{1,t-1}) + \varepsilon_{1,t}$$
$$\Delta y_{2,t} = \alpha_2(y_{2,t-1} - \beta y_{1,t-1}) + \varepsilon_{2,t}$$

In the above models, the variable in the right hand side is the error correction term, which is zero in the long run equilibrium. When y_1 and y_2 drift from their stable position or the long run equilibrium, the error correction term will not be zero and both y_1 and y_2 will adjust to restore the equilibrium status. The coefficient of α_i measures the rate of "adjustment of the i^{th} endogenous variable towards equilibrium."[19] Using the same variables used to check for cointegration, I made a VECM model with 3 lags and 2 cointegrating relations (check Table 1 in the Appendix for the full model). The value of lag which results in the minimum information criteria (in this case, AIC) is chosen as the order of the model. Table 5 depicts only the error correction terms (ECT1 and ECT2) and the p-values of the the four variables. As the p-values are extremely small, the coefficient estimates are significant at the 5 percent critical value.

Variables	**ECT 1**	**ECT 2**	**p-value**
Housing Starts	-0.0267838950	-0.0363147859	5.283760e-01 3.335254e-01
Private Houses Completed	0.3198985285	0.3288063833	8.922016e-19 3.075843e-24
Mortgage Rate	0.0002606689	-0.0001891956	1.870030e-02 5.323680e-02
Housing Supply	0.0003273892	-0.0002339375	1.274370e-01 2.175834e-01

Table 5: Coefficient estimates and p-values of the two error correction terms of the four variables

[19] Source: http://www.eviews.com/help/helpintro.html#page/content/VAR-Vector_Error_Correction_(VEC)_Models.html

The error correction terms (ECT) describe how the time-series adjust to disequilibrium and ECTs are between 0 and 1. The ECTs for housing starts are negative. This implies that when $hous\,st_{t-1}$ is above its long-run level, then there will be a negative change in $\Delta\, hous\, st_t$, which would pull housing starts back towards its long-run relationship with mortgage rate, housing supply and private houses completed. The estimated coefficient $ECT(-1)$ of $hous\,st$ is -0.0268, suggesting that 2.68 percent of disequilibrium is corrected between between one month. The coefficients of ECT1 and ECT2 of housing supply and mortgage rate are almost negligible, denoting that there are dominant variables and do not adjust towards the equilibrium.

While VECM tests for long term relationship, a VARX captures short-run relationship among the variables employed (example, where there is a shock). We can transform a VECM into a Vector Autoregression Model (VAR) to forecast values. VECM and the corresponding VARX model equivalently represent the same model. Yet, as it is computationally easier to forecast directly from a VARX model, I did not transform the vector error correction model, rather I made a separate VARX.

3.2.9. Vector Autoregression with Exogenous Variables (VARX)

A vector autoregression model generalizes a univariate autoregressive model to forecast a vector of time series. A VAR-X extends a VAR model by including exogenous variables in the system. The system has one equation per variable. The right hand side of each equation comprises of a constant, the variable's autoregression, its distributed lags, and the lags of other variables in the system. For, instance, if we would like to model three different time series $x_1, x_2, \, and \, x_3$, the vector autoregressive model of order 1, or $VARX(1)$ with three variables is the following:

$$x_{t,1} = \alpha_1 + \beta_{11}\, x_{t-1,1} + \beta_{12}\, x_{t-1,2} + \beta_{13}\, x_{t-1,3} + \varepsilon_{t,1}$$
$$x_{t,2} = \alpha_2 + \beta_{11}\, x_{t-1,1} + \beta_{22}\, x_{t-1,2} + \beta_{23}\, x_{t-1,3} + \varepsilon_{t,2}$$
$$x_{t,3} = \alpha_3 + \beta_{31}\, x_{t-1,1} + \beta_{32}\, x_{t-1,2} + \beta_{33}\, x_{t-1,3} + \varepsilon_{t,3}$$

In the first equation, x_1 is a function of its past lag x_{t-1} and the first lag of x_2 and x_3. In general, if a system has k variables, each equation in a $VARX(p)$ model will include p lags of y and p lags of x, and an error term. We treat all variables symmetrically in VARX i.e we model them in such a way that these endogenous variables equally impact each other. For a stationary series, we can directly fit a VARX to the data and forecast. This is called "VARX in levels". Otherwise, we first commonly difference the

non-stationary data, and then fit the model. The resulting model is called "VARX in differences." Using leveled variables (which are stationary) in VARX models can result in spurious regression. But, differenced variables will remedy the problem. [20]In both instances, the concept of least squares estimates the model.[21]

The VARX model can be used when the variables under study are $I(1)$ but not necessarily cointegrated. I estimated a $VARX(3)$ model (Table 2 in the Appendix) with with four predictors – housing starts, housing supply, mortgage rate and private houses completed. The appropriate p number of lags is selected using the usual goodness of fit criteria and then checked if the residuals correspond to the model assumptions – absence of serial correlation, homoscedastic and normally distributed residuals. The lag parameter, $p = 3$ minimizes the information criterion: AIC. After computing the Breusch-Godfrey test for serially correlated errors, where the hypothesis are:

$H_0 =$ no serial correlation, $H_1 =$ serial correlation is present.

The large p-value of $0.0786 > 0.05$ suggests that the residuals for this model are not serially correlated, so the model is appropriate. [22] I have forecasted the values of housing starts using $VARX(3)$ in the results chapter. But "a good forecast does more than provide a current 'best estimate'—it identifies the key variables and states the nature of their impact. As new data come in, such a forecast gives a basis for continual reappraisal of the situation." [23] So, I have performed structural analyses such as impulse response functions and granger causality test that summarize the properties of a $VARX(p)$.

3.2.10. Impulse Response Function

As the coefficients of a VARX model are very hard to interpret, we interpret the impulse response function which is based on momentum and persistence. The momentum effect occurs when the variable moves in

[20] Sims, Stock and Watson (1990) have estimated and conducted hypothesis testing on time series models, in which some variable are non-stationary. They made a vector autoregression at levels even when variables have unit root.

[21] Source: https://newonlinecourses.science.psu.edu/stat510/node/79/

[22] Difference between VARX and ARIMAX: VARX is devoid of MA terms and uses autoregressive lags to approximate MA terms. It's solution is a less parsimonious solution than when we directly include MA terms in the ARIMAX model. OLS or GLS can estimate VARX quickly, whereas maximum likelihood method estimates ARIMAX model, which is usually slow.

[23] by George Schultz, A Note on Forecasting, 1963

the same direction but moves away from its historical mean for some time. Hence, it temporarily offsets the force of regression (convergence) towards the mean. As opposed to momentum, a variable that is persistent will stay in the current state before converging to its historical mean. IRFs are useful in answering questions such as how will a variable, which is at its historical mean, respond in the future, to a temporary unit shock in a single period? Do the forecasts have cycles, and how quickly do they converge to the mean?

In IRF, we shock one variable, say federal funds rate, and propagate it through the fitted VARX model for a number of periods. We can trace this through the VARX model and see if it impacts the other variables in a statistically significant way. The vertical axis represents the y variable. The solid black line estimates the amount y that is expected to change if there is a one unit impulse on the x variable after a time period.

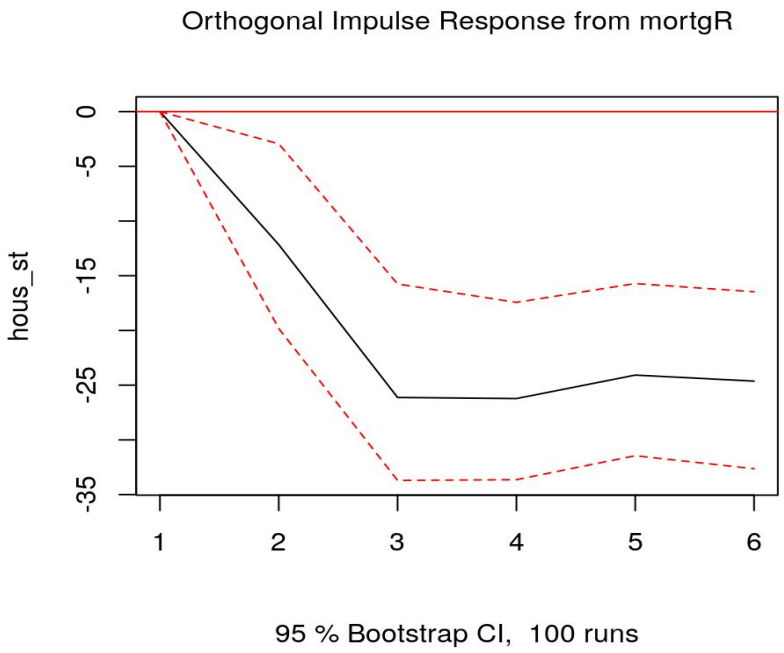


Figure 9: Impulse response from mortgage rate on housing starts

When there is a shock in housing starts by mortgage rate, the value of housing starts traverses negatively at a fixed rate away from the mean, creating a momentum effect. After three periods, it hovers around a constant rate, creating a persistantance effect, and doesn't return to its historical mean. The dotted lines show the 95 percent interval estimates of these effects. The VARX function prints the values corresponding to the impulse response graphs.

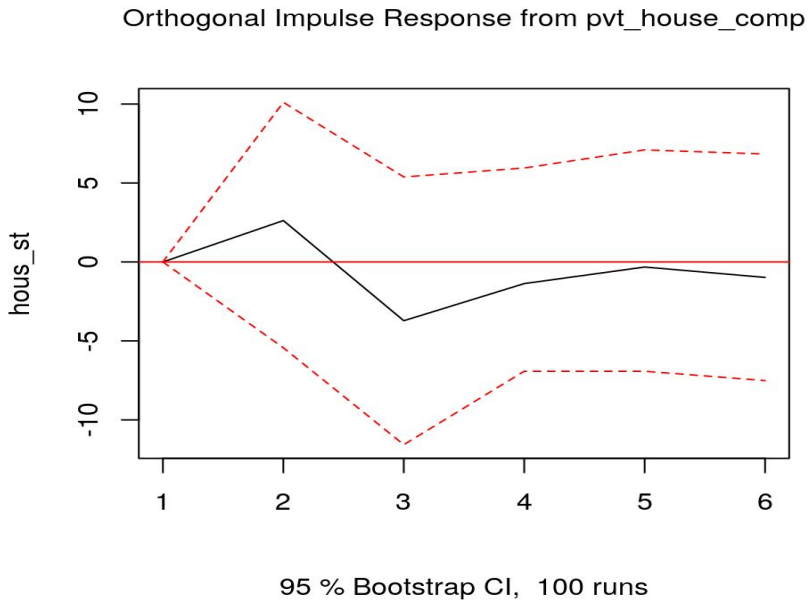


Figure 10: Impulse response from private houses completed on housing starts

The impulse response of private houses completed on housing starts shows that with a one unit shock by private houses completed causes housing starts to slightly increases, then fall towards zero into the negative territory at a constant rate as the effect of shock dies, followed by a slight upward movement as housing starts reverts to the mean.

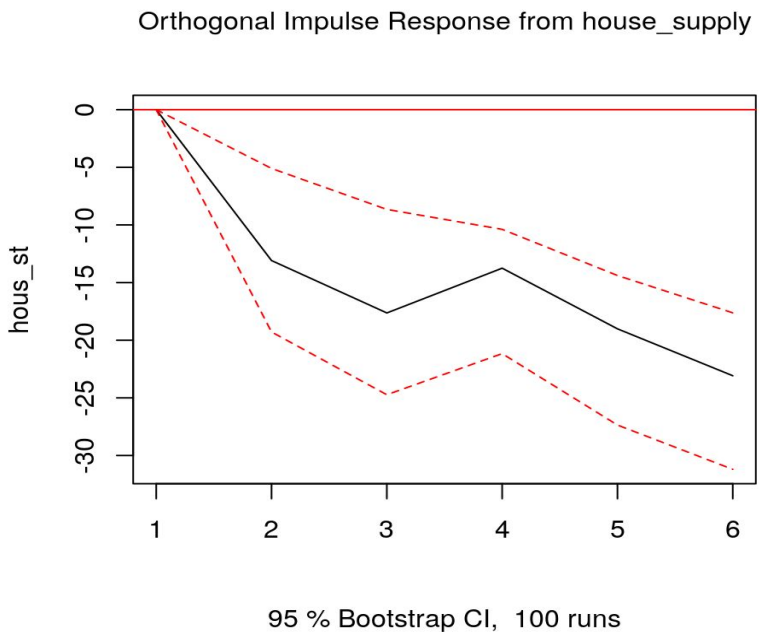


Figure 11: Impulse response from housing supply on housing starts

An impulse (shock) to housing starts from housing supply at time zero has large effects the next period as housing starts falls precipitously. The downward momentum persists, and even after six periods of shock, the value of housing starts does not revert to the mean.

3.2.11. Granger Causality

As VARX models forecast each variable, we can describe the relationship between the variables using granger causality test. The F-test on the lags of other variables implements the granger causality. It tests the null hypothesis that all the lags of of a variable X are do not have a predictive power for Y, i.e. it does not not contain useful information to predict Y. If feedback loop is present, i.e. if Y has predictive power for X, then X and Y are both endogenous, and we would have to make a VARX model. It does not test if X causes Y, but examines if the lags included are informative in predicting Y. For instance, federal funds rate can granger cause housing starts if housing starts can be more accurately predicted by the lagged values of both housing starts and federal funds rate, rather than the lagged values of housing starts alone. Thus, the granger causality test examines if lagged values of a variable can enhance the forecasts of another variable. There are three steps involved in performing the Granger Causality F-test. Firstly, the restricted model by regressing y on y lags without x lags is:

$$y_t = a_1 + \sum_{j=1}^{m} \gamma_j y_{t-j} + \varepsilon_t$$

Secondly, the unrestricted model by adding x lags and regressing is:

$$y_t = a_1 + \sum_{i=1}^{n} \beta_i x_{t-1} + \sum_{j=1}^{m} \gamma_j y_{t-j} + \varepsilon_t$$

Lastly, we use the F-test to test the null hypothesis that $\beta = 0 \ \forall \ i$

In the first case, the F-test rejects the null hypothesis that housing starts do not granger cause private houses completed, mortgage rate and housing separately as the p-value of $1.597 \times 10^{-11} < 0.05$. Likewise, the test rejects the null hypothesis that mortgage rate does not granger cause housing starts, private houses completed and housing supply as the p-value of 2.065×10^{-10} is very small. However, private houses completed does not granger cause housing starts, mortgage rate and housing supply as the p-value of 0.6742 is very high.

Null hypothesis	**p-value**
Housing starts do not granger cause private houses completed, mortgage rate and housing supply	1.597478e-11
Private houses completed do not granger cause housing starts, mortgage rate and housing supply	0.6742
Mortgage rates do not granger cause private houses completed, housing starts and housing supply	2.065e-10
Housing supply do not granger cause private houses completed, mortgage rate and housing starts	4.975e-07

Table 6: p-values of the four different null hypothesis in a Granger Causality Test

3.3. Machine Learning Models[24]

In this section, I have predicted housing starts in the United States using a stacked ensemble of the following machine learning methods: K-Nearest Neighbors, Ridge Regression, Support Vector Regression, and Artificial Neural Networks. Machine learning relies on cross-validation to prevent overfitting and underfitting. Albeit these model generate more accurate predictions than econometric methods, they are hard to interpret. Together these aforementioned individual learning algorithms, or base learners in the ensemble enhance the predictive accuracy and robustness, which is otherwise not possible from using each of them separately.

Super Learning, or stacking is a class of algorithms that finds the optimal combination of the base learners. Ensemble machine learning methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms. In setting up the ensemble, I specified the base algorithms used with specific parameters of each model and train each of them in the training set. Then, I conducted a *k*-fold cross validation[25] on each of the learners and obtained the cross-validated predictions from each of them. Models which are more accurate are assigned higher weights in stacked ensemble. The predictive accuracy increases when the ensemble uses diverse set of initial base learners with different hyperparameters and feature subsets[26]. However, such fine-tuning to generate complex models may cause the problem of overfitting where the model perfectly fits the training set, but generalizes poorly in the test set. Partially caused by collinearity between predictors, this is can problematic in model stacking as we combine multiple predictors together. To prevent overfitting, I used cross-validation and regularization.

[24] Chapter 3.3.1 - 3.3.3, 3.3.5 - 3.3.7 were expanded considerably from the ideas built in the project.

[25] Unsure of the best way to evaluate time series models while forecasting, economists and statisticians often evaluate a model's performance in the test set or out of sample (OOS) set. Cross validation does not account for unit roots and serially correlated variables when using econometric models. The former method evaluates in only one set, while the latter (CV method) evaluates in multiple sets. OOS evaluation is a standard procedure as conventional models such as ARIMA are completely iterative i.e. they start estimating from the beginning of the series. Opsomer et. al (2001) explain that if errors are highly autocorrelated, then cross validation "underestimates bandwidths in a kernel estimator regression framework", overfitting the model. Nonetheless, such problems are immaterial when applying ML methods, so we use CV.

[26] Different from dimensionality reduction, **feature selection** reduces the number of attributes (or variables). In contrast to dimensionality reduction, such as, Principal Component Analysis where we create new combinations of attributes, in feature selection we exclude variables without altering them.

3.3.1 Cross Validation and Hyperparameter Optimization

We use *k*-fold cross-validation (CV) as a resampling procedure to evaluate the performance of machine learning models when sample size is limited. Splitting the dataset into *k* number of groups, we apply the model in the $k-1$ groups and tested the model in the k^{th} group. It estimates how the model is expected to perform when we make predictions in the test set. The $k-1$ groups constitute the training set and the k^{th} group constitutes the test set. After fitting the model in the training set, the error rates are calculated on the test set. Unlike a longitudinal dataset, I cannot use the conventional approach of cross-validation in a time series dataset. So, I divided the historical data set on the training set and validation set by the "time slices" method, such that the training data will lie in the first time period and the validation set in the next one. This works similar to cross validation but uses the first 180 observations of the time series in the first training data set and the next 12 observations as the validation data. The second training set has 181 observations i.e. the first 180 observations and the additional one observation. The second validation set has 12 observations from 182^{nd} to 194^{th} observation, and the process continues. In total, the time slices methods has created 217 folds. Modeling in 211 datasets is computationally expensive, so instead of incrementing each fold by every month, I incremented the number of folds by every 12 months. Now, the second training set has $180+12=192$ observations, and the second validation set has 12 observations from 193^{rd} to 205^{th} observation. Thus, there are $217/12 \sim 18$ fold cross validation sets or 18-fold time slices.

In many machine learning models, there are certain parameters, called hyperparameters, whose values are set before starting the learning process in the training set. These are in contrast to other parameters that are derived after training. The simple algorithm of ordinary least squares regression has no hyperparameter, so it only relies on the given data to evaluate the coefficients. In contrast to OLS, the more complex algorithm, called ridge regression, adds a regularization hyperparameter to the OLS regression before training the model. We choose the values of the hyperparameters through "hyperparameter optimization." These parameters affect the computational time to run a model and its forecasting accuracy. Tuning the hyperparameters relies more on experimental results, than theory so there is no single best way choose the values. Usually, we try numerous different combinations and assess each model's performance. After specifying the hyperparameters, the algorithms or models are trained in the dataset.[27] One way to find the optimal configuration is via grid search, wherein I trained numerous models for different values of parameters and choose the best one. Here, R exhaustively searches through a pre-specified subset of

[27] Source: https://towardsdatascience.com/demystifying-hyper-parameter-tuning-acb83af0258f

hyperparameters of the learning algorithm and and checks the performance by doing a 18-fold cross validation on the training set.

3.3.2. Principal Component Analysis for Data Preparation

After CV, I noticed that the variables in the whole dataset are highly correlated with each other, causing multicollinearity. These are redundant variables as they add nothing new to the model. As this may bias OLS estimates, I used Principal Components Analysis (PCA). PCA reduces $p-$dimension dataset to an m-dimension dataset where $p > m$. It describes the original data using fewer variables or dimensions than initially measured. We project the original data and the differenced variables onto a new, orthogonal basis. [28] This removes multicollinearity.

A data matrix X of dimension $n \times p$ (where n is the number of observations and p is the number of variables) can have up to $min(n-1, p)$ principal components. The goal of PCA is to significantly reduce the number of variables used such that the least number of principal components explain maximum variability. Each component is a linear combination of the original variables. No information is redundant as all principal components are orthogonal to each other. Combined together, they form an orthogonal basis for the data space. As a form of multidimensional scaling, it linearly transformed variables into a lower dimensional space. This space retains maximum information about the original variables.

In this unsupervised learning problem, I obtained the principal components from a raw dataset using the following steps:[29] The original dataset of $d+1$ dimensions is converted to a d dimension dataset. Then, I computed the mean and covariance matrix of X and Y of the whole dataset using the following formula:

$$cov(X, Y) = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{x})(Y_i - \bar{y})$$

[28] Let V be a subspace of Rn for an n-dimensional space, and B1={v1,v2,....,vr} is a collection of vectors from V. $B1$ is a basis for V if is both linearly independent and spans V. The vectors in $B1$ are linearly independent when they are not a multiple of each other. $B1$ spans V when every vector in V is a linear combination of those in $B1$. Let $B2$ be another basis for V. If $B1$and $B2$ are perpendicular or orthogonal to each other, then these vectors form an orthogonal basis.

[29] There are two categories of learning processes: supervised, and unsupervised. In supervised learning, the dataset has output values (i.e. we have the prior knowledge of response variable). Using the output values, the supervised learning model best estimates how the explanatory and response variables are associated. For instance, logistic regression, support vector machines, random forests and artificial neural networks. On the other hand, when the dataset does not include the values of the output, the unsupervised learning model such PCA infers the structure of data and is useful for exploratory analysis.

This gives a square matrix of $d \times d$ dimension, and after which I got the eigenvectors and their corresponding eigenvalues from the above covariance matrix. A vector whose direction is constant when applying linear transformation is known as an eigenvector. For a square matrix *A*, and vector *v,* and scalar λ such that $Av = \lambda v,$ then λ is the eigenvalue associated with eigenvector *v* of *A*. The roots or solutions of the equation $det\,(A - \lambda I) = 0$ gives the eigenvalues of A. From the eigenvalues, I solved for eigenvectors and arranged them from highest to lowest order of eigenvalues.

Thereafter, I chose k eigenvectors with the largest eigenvalues to form a $d \times k$ dimension of matrix W. The eigenvectors with the lowest eigenvalues have the least information about the data's distribution, so I had dropped those. [30]

PC_i reflects as much information from the original variables. If Σ is a covariance matrix of the ten explanatory variables $X = \{hous\ st,\ CPI,\ \ldots,\ house\ supply\}$ with eigenvalues of Σ: $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{10} \geq 0,$ and their respective eigenvectors: $e_1,\ e_2,\ \ldots,\ e_{10}$, then the i^{th} PC is:

$$PC_i = e_i{}^T X = e_{i1}\, income + e_{i2}\, fed\, fundsR + e_{i3} CPI + \ldots + e_{i\,10}\, house\ supply,\ i = 1, \ldots 10.$$

$\frac{\lambda_k}{\sum_{i=1}^{10} \lambda_i}$ is the proportion of total information explained by the k^{th} principal component.

In PCA, every variable is centered at zero so that we can easily compare each principal component to the mean. Centering also removes problems arising the scale of each variable. The components are always sorted by how important they are, so the most important components will always be the first few. In this dataset, $PC1$ accounts for more 60 percent of total variance in the data. The cumulative proportion shows how much variance is accumulated. The first principal component captures most of the variance in the data set, and the subsequent principal components capture the remaining variability, thus the proportion of variance decreases with each principal component.

Dimension	**Eigenvalue**	**Percentage of variance**	**Cumulative Variance**
1	6.0739086	60.7390863	60.73909
2	1.8835272	18.8352721	79.57436
3	1.1807085	11.8070849	91.38144
4	0.4287866	4.2878663	95.66931

[30] Source: https://medium.com/@aptrishu/understanding-principle-component-analysis-e32be0253ef0

5	0.2441449	2.4414486	98.11076
6	0.1353575	1.3535751	99.46433
7	0.0299749	0.2997494	99.76408
8	0.0182776	0.1827762	99.94686
9	0.0038175	0.0381750	99.98503
10	0.0014966	0.0149660	100.00000

Table 7: Eigenvalues and the variance explained by each principal component

A common technique to determine the number of PCs to use is to eyeball the scree plot below wherein we observe the "elbow point", where the proportion of variance explained (PVE) plummets. An eigenvalues less than 1 would mean that the component actually explains less than a single explanatory variable. The first 3 components have an eigenvalue greater 1 and explains almost 91 percent of variance. Alternatively, the first 6 components explain 99.46 percent variance. Thus, I have reduced dimensionality from 12 to 6 while only "losing" 0.53567 percent of variance.[31]

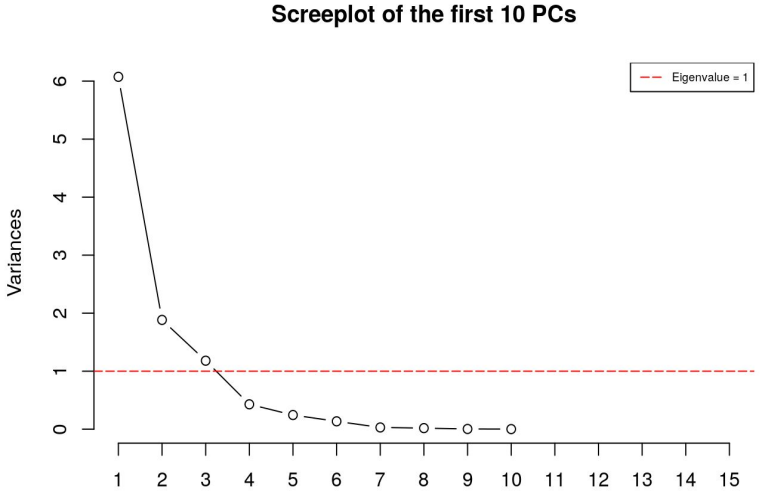


Figure 12: Visualize eigenvalues (scree plot): shows the percentage of variances explained by each principal component.

[31] Albeit the plot suggests that we should use three PCs only as they explain enough variation, I chose six PCs because the MAPE and percent bias in the the test sets were considerably lower when using the latter. So, all the models performed performed better with six PCs.

After creating a new dataset with these six principal components and the response variable, housing starts, I split the dataset into training and test – the first 80 percent of the observations are in the training set and the last 20 percent are in the test set. The individual and stacked models are fit in this training set. Next, I have explained the individual models used to make the stacked ensemble – K - Nearest Neighbors, Support Vector Regression, Ridge Regression and Artificial Neural Networks.

3.3.3. K- Nearest Neighbors

K-Nearest Neighbors (KNN) is one of the simplest machine learning models that is mostly used to classify data points based on how the neighbors are classified (separated into different categories), but also used in regression to predict values. KNN algorithm stores all the available cases and classifies the new case based on how similar it is to the *k* nearest cases. To find the nearest neighbors, we calculate the Euclidean distance between the new point (a, b) and each point in the training set (x_i, y_i):

$$distance = \sqrt{(a - x_i)^2 + (b - y_i)^2}$$

We chose the *k* value which determines the number of neighbors to consider before establishing the value to the new observation. *k* is a hyperparameter that is chosen to get result in the best possible fit for the dataset. Based on the distance, we choose the closest *k* data points. When *k* is very small, and if there are slight perturbations in the training set, the decision boundary changes considerably (becomes overly flexible), increasing the variance but reducing the bias. Alternatively, when the value of *k* is very high, the model poorly performs. The method becomes less flexible when *k* grows and the decision boundary smoothens or resembles linearity, causing high bias but low variance. While the predictions become relatively stable, they usually are the values which most frequently occur (prior believes). The final prediction $f(x_0)$ for the point x_0 is the average value of the *k* training observations y_i that are closest to x_0, represented by N_0.[32]

$$f(x_0) = \frac{1}{k} \sum_{x \in N_0} y_i$$

[32] Source: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

In the KNN model with six predictors on the training set, k is set from 0 to 10. Ten different models with all ten values of k are trained and ultimately we select the model with the lowest error. When $k = 2$, the error rate is lowest and thereafter, it rises. The test set MAPE of 27.9% and a negative bias of 18.15%.

Now, I will discuss another prominent but more advanced technique – support vector machines. SVMs are applied to both classification and regression. When it is applied to a regression problem, it is called support vector regression.

3.3.4. Support Vector Regression

Just as we minimize the error rates in simple regression, in Support Vector Regression, the goal is to fit the error within a particular threshold.

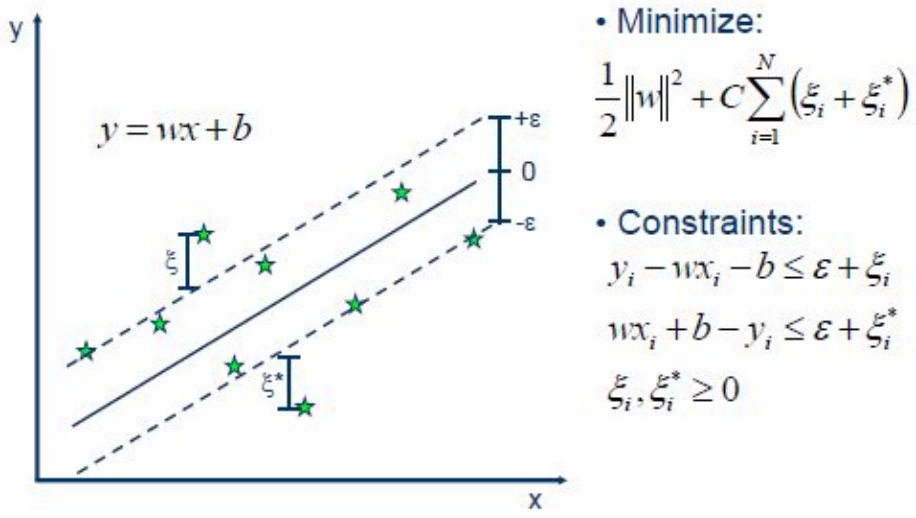


Figure 13: Hyperplane, boundaries and margin in a support vector regression[33]

In figure 13, the perpendicular distances between the blue middle line and the observed values (shown by the green stars) nearest to the blue line are called margins. Most of the green stars lie within the black dotted lines (boundary lines). We consider all the points that are within the boundary line when moving along the SVR. The decision boundary is the margin of tolerance. In SVR, we only consider those points

[33] Source: http://www.saedsayad.com/support_vector_machine_reg.htm

that minimize errors, fitting a better model. These stars are known as support vectors, which are closest to the boundary lines. Hyperplane in a two-dimensional space is the blue line in the middle that is useful to predict the target values. In an *n*-dimensional space, it is a $(n-1)$ subspace . The line of best fit is the hyperplane with the maximum number of points. In SVM, the hyperplane separates between data classes.

The equation of hyperplane is: $wx+b=0$. Each of the two boundary line is at a distance of $+\varepsilon$ and $-\varepsilon$ from the hyperplane. Therefore, the equations of the boundary lines are: $wx+b=+\varepsilon$, *and*, $wx+b=-\varepsilon$. Thus, if $y=wx+b$, the equation that satisfies the SVR for a linear hyperplane is: $\varepsilon \le y-wx-b \le \varepsilon$. We find a decision boundary at ε distance from the hyperplane such that support vectors are within the boundary lines. Maximizing the margin is analogous to minimizing the complexity of the model. For that, we regularise the solution by minimising *w*.

The SVR model uses two parameters: the cost and loss values to avoid overfitting. The values for cost parameter used in the model are 4,8,16 and 32. For each cost value, we use L1 and L2 type of loss parameters. The two types of loss functions are:

In L1-SVM, we optimize the following: $minimize\ \frac{1}{2}|w|^2 + C\sum_{i=1}^{M}(\xi_i + \xi^*_i)$

In L2-SVM, we optimize the following: $minimize\ \frac{1}{2}|w|^2 + \frac{C}{2}\sum_{i=1}^{M}(\xi_i + \xi^*_i)^2$

$w=$ regularization term added to avoid overfitting,

$C \in R$ is the penalty parameter. We can tune model performance to ensure a balance between the regularization term and loss function by setting various values of C.

The L1 loss function is called the least absolute deviations (LAD) or error (LAE) and the L2 loss function is also known as the least squares error. The L1 loss function is more robust than L2 as it is resistant to outliers in the data. As L2 squares the error in the model, the overall error is larger $\{(\xi_i + \xi^*_i)^2$ vs $(\xi_i + \xi^*_i)\}$ than the L1 loss function. So, the model is more sensitive to individual observations and adjusts the model to minimize the error. In that case, if an observation is an outlier, the model will be modified to minimize the single outlier case at the cost of other observations. This is because the errors from other other observations are smaller as opposed to the single outlier case. Alternatively, the L2 loss function

always yields one stable solution. A model is considered unstable if the slope of the regression line considerably changes when there is small change in data, creating multiple solutions. The L2 loss function is stable as a small change in data point will only slightly alter the regression line, giving only a single solution. [34]

In support vector machines, one drawback is that generally we cannot linearly define the classifier or hyperplane i.e. in most real life cases, we cannot actually draw a straight line or a plane that separates two two categories of data points. In that case, the hyperplane will be a wavy curve or a surface (non-linear decision boundary). To make a linear classifier, we perform a kernel trick, in which we lift the feature space to a higher or possibly an infinite dimensional space.

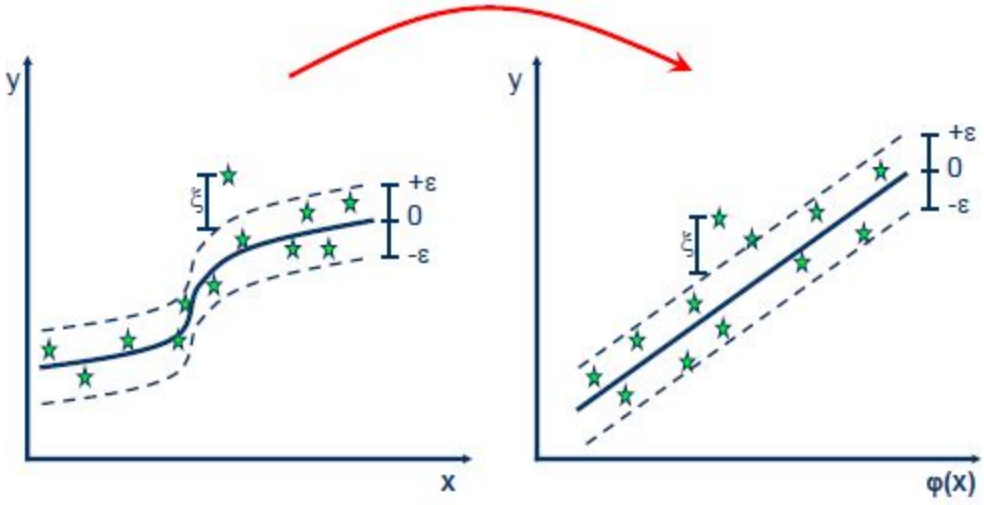


Figure 14: The kernel trick transforms the non-linear (left) to a linear decision boundary[35]

Similarly, in support vector regression, we draw a hyperplane that minimizes the loss function. Hyperplanes change when the loss function changes. We apply the kernel trick to lift the feature space, or convert the lower dimension data into a higher dimension, resulting in a non-linear decision boundary. From the analysis, the model outputs the MAPE in the test set is 9.53% and bias is 3.33% when the cost is 16 and loss is L2.

[34] Source: http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/
[35] Source: https://towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496

Ridge regression is another model that uses the L2 penalty on the weights of the equation, i.e. it minimizes the L2-regularized squared error instead of only the squared error.

3.3.5. Ridge Regression

When predictors in a regression are strongly correlated, regression coefficients of a variable depends on other predictors in the model. Therefore, the specific independent variable does not reflect the effects of the predictor on the regressor. It only partially or marginally effects the regressor based on other predictors in the model. Ridge regression adds bias to alleviate multicollinearity. [36]We fit a model with all p predictors and regularize or constrain the coefficient estimates, i.e. use a method that shrinks the coefficient towards 0 to reduce the variance of the variable. Similar to least square estimates, the ridge regression coefficient estimates β^R by minimizing:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j{}^2 = RSS + \lambda\sum_{J=1}^{P}\beta_j$$

The penalty term, $\lambda \geq 0$ is the tuning parameter (or the hyperparameter) that is separately determined. $\lambda\sum_{j=1}^{p}\beta_j{}^2$ is called the shrinkage penalty, which is small when $\beta_1, \beta_2, ...\beta_p$ are close to 0. The shrinkage penalty is not applied to the intercept, β_0 . When $\lambda = 0$, the ridge regression generates least square estimates, but as $\lambda \to \infty$, the effect of the shrinkage penalty increases, which shrinks the coefficients towards 0. Ridge regression generates different set of coefficient estimates β^R for each value of λ, unlike a single set of coefficient estimates generated from least squares. Increasing λ, decreases the flexibility of the model, lowering the variance but raising the bias. Unlike the OLS regression, the scale of parameters highly impact the coefficients of ridge regression, so we should standardize the predictors to the same scale before applying this method. [37] In ridge regression, when $\lambda = (-4, 10)$, the test set MAPE is 11% and there is a negative bias of 5.07%. The lowest error rates resulted when the values of hyperparameters were $\alpha = 0$ and $\lambda = 32$. Next, I will discuss the final model incorporated in the ensemble, known as Artificial Neural Networks.

3.3.6. Artificial Neural Network

[36] Source: https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Ridge_Regression.pdf
[37] Source: http://statweb.stanford.edu/~tibs/sta305files/Rudyregularization.pdf

The human brain consists of about 80,000 is a convoluted network of neurons. A neuron is a cell that transmits electrochemical signals or nerve impulse to process information. Artificial Intelligence (and its subset: Machine Learning) "gives cognitive powers to computers to program them to learn and solve problems". Thus, we can simulate human intelligence via computers. A neuron is a central processing unit that computes mathematical operations in a set of inputs to get an output.

A neural network is a network of interconnected nodes, called neuron. Each neuron is a variable and the intermediary variables are called derived variables. Just as logistic regressions, logit models, models with polynomial transformations and lagged variables are derived from original models and variables, so are the neurons in the hidden layers. All the arrows representing neurons represent parameters, called weights. The output layer contains all the output variables or the output neurons. The layers between input and output layers are called "hidden layers" and they include hidden neurons. The hidden layer is the middle layer which processes information and yields output. In the model, the inputs are $PC_1, PC_2, ..., PC_6$, and the weights are $w_1, w_2, ...w_6$. Thereby, the output is $y = f(x) = \sum PC_i w_i$, where $i = \{1, 2, 3, 4\}$ inputs. In neural networks, hyperparameters are the variables which determine the structure of the network and how to train the network. These are the number of hidden layers, and the learning rate, respectively. Before training the model i.e. prior to optimizing the bias and weights, hyperparameters are set. We add more layers until the test rate does not improve further.

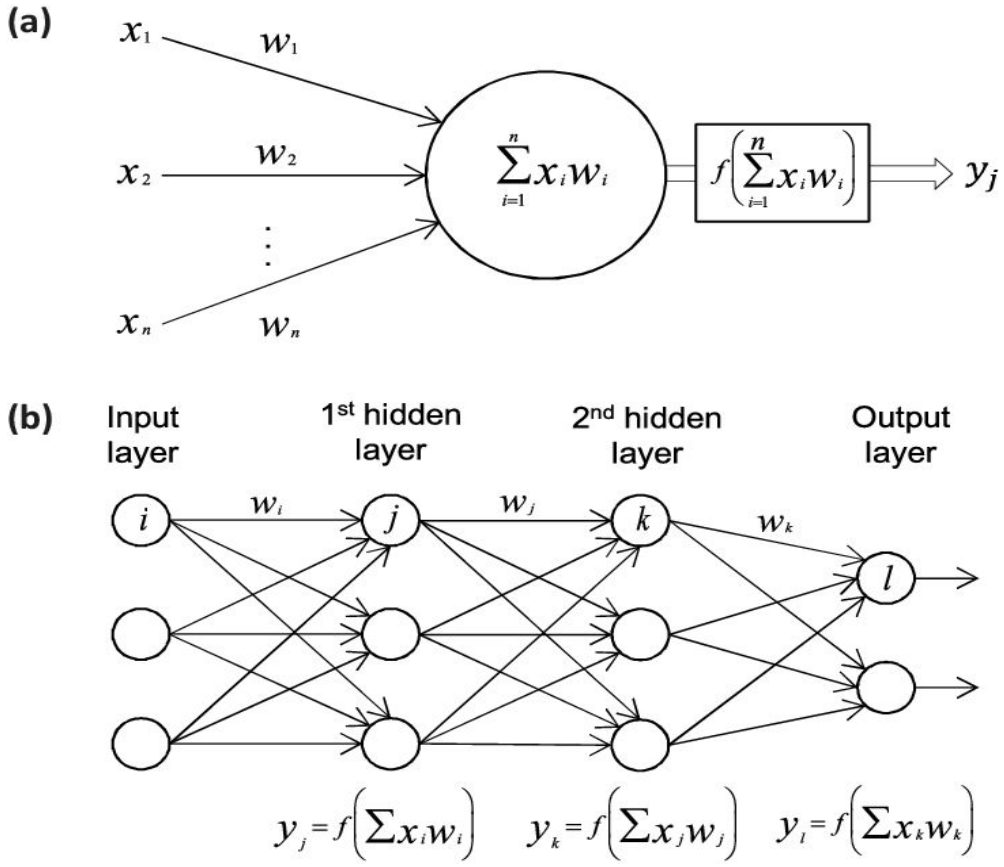


Figure 15 : Basic structure of an Artificial Neural Network[38]

The more hidden layers or neurons, the more the network will be able to capture the complicated relationship. In the hidden layer, such a variable is a function of the previous layer. The top neuron in the hidden layer is a function of all the input variables, such as their weighted average. The second neuron in that layer is also a weighted average but uses different weights. Each neuron in the second hidden layer gets inputs from the first hidden layer. Therefore, the top neuron in the second hidden layer might be a weighted average of all the neurons in the first hidden layer. Finally, the output neuron is the weighted average of the neuron in the last hidden layer. So, eventually, the output variable is a complicated function of the input variable.

Activation function is the function applied on this output. Next layers' inputs is the output generated from previous layers' neurons. A mathematical function that converts inputs into output and processes the neural network is an activation function. In the absence of activation functions, neural networks behave

[38] Source of the figure: https://www.researchgate.net/figure/a-The-building-block-of-deep-neural-networks-artificial-neuron-or-node-Each-input-x_fig1_312205163

like linear functions, where the output is directly proportional to the input. One type of activation function is sigmoid or logistic, which I have used the model. The *sigmoid* function produces a sigmoidal curve, that is *S* shape. The model is logistic and values are in the range: $(0, 1)$. The formula is: $f(x) = \frac{1}{1+e^{-x}}$

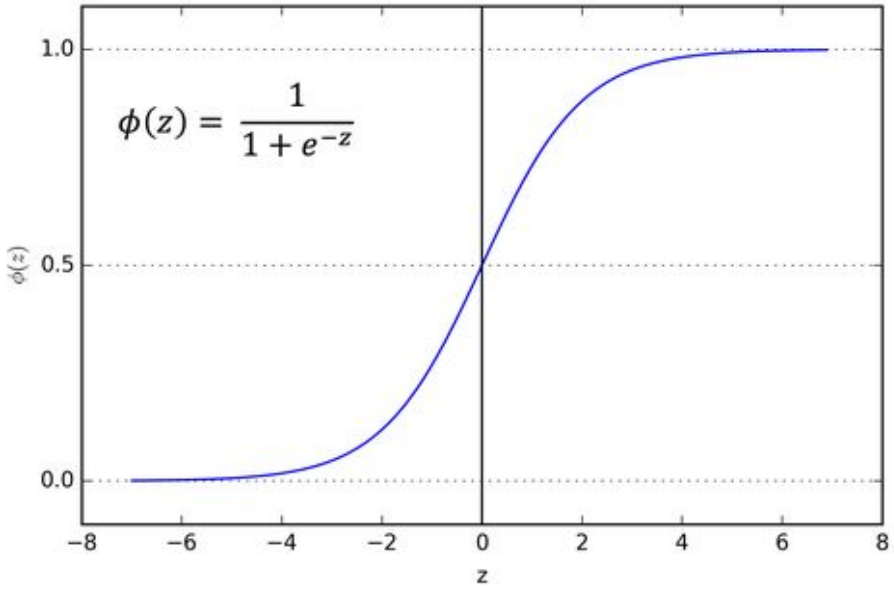


Figure 16: Sigmoid (logistic) function applied in the ANN model[39]

The activation function processes from input to outer layer via hidden layer(s) through a mechanism called forward propagation. First, the $Y = \sum_{i=1}^{n}(input_i \times weight_i) + bias \ \forall \ Y \in (-\infty, \infty)$, is applied at each layer, and then the value obtained from the activation is propagated to the next layer. The figure below depicts a simplified neural network and the technique behind the "black box". The entire neural network is a function of outputs approximated from each individual neuron. In supervised learning, we present the inputs, output and can modify the weights to minimize the gap between the observed and the predicted output. Alternatively, in unsupervised learning, we cannot modify the weights. In this case, the neural network amends it own weights, thereby, similar inputs result in similar outputs. The network solely observed differences and patterns in the output without any external help.

In neural networks, the weight parameters are closely analogous to the coefficient estimates in a regression model. These weights describe the association between variables and dictate the importance or influence of each variable that is processed in the network. Thereby, variables which are irrelevant or do not influence the response variable significantly, are suppressed by lower (and even negative) weights, and vice-versa. As the number of weights in neural networks exceed number of coefficients in regression

[39] Source of picture: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

model, neural networks are extremely flexible. Hence, they are useful for modeling non-linear functions with many interactions.

After obtaining the output in the output layer, I calculated the error by subtracting the original from the predicted output. In forward propagation, error helps to correct the biases and weights. The gradient descent determines the amount of weight that should be changed. Alternatively, the process of backpropagation involves partial derivatives of each neuron's activation function to identify the gradient or the slope in the direction of the updated weights. The gradient informs how much the error changes a weight changes. Backpropagation keeps changing the weights iteratively until the error is reduced by the maximum amount, called learning rate. The scalar parameter, learning rate sets the adjustment rate to reduce the errors faster. Higher is the value of learning rate, faster the algorithm will reduce the errors.

At each step, gradient descent helps to find the global cost minimum, where the error is the lowest. Then, the model fits the data very well, giving more accurate predictions. If $h(x)$ is the predicted value of y, which is the observed value, then the error for one data point is: $error = (h(x) - y)^2 = predicted - actual$. Repeating this process for all the data points in the dataset and summing all the errors to one combined error gives the Cost Function, $J(w)$.

$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})$, where:

$m =$ the number of training examples, i.e. observations in the training set

$x^{(i)} =$ input vector for the i^{th} training example

$y^{(i)} =$ the class label for the i^{th} training example

$w =$ weights $(w_1, w_2, ..., w_n)$

$h_w(x^{(i)}) =$ algorithm's prediction for the i^{th} training example using the parameter θ.

The goal is to minimize the cost function (bottom of the curve) so that the error is close to 0 by changing the value of w at each step. The diagram below shows how gradient descent works:

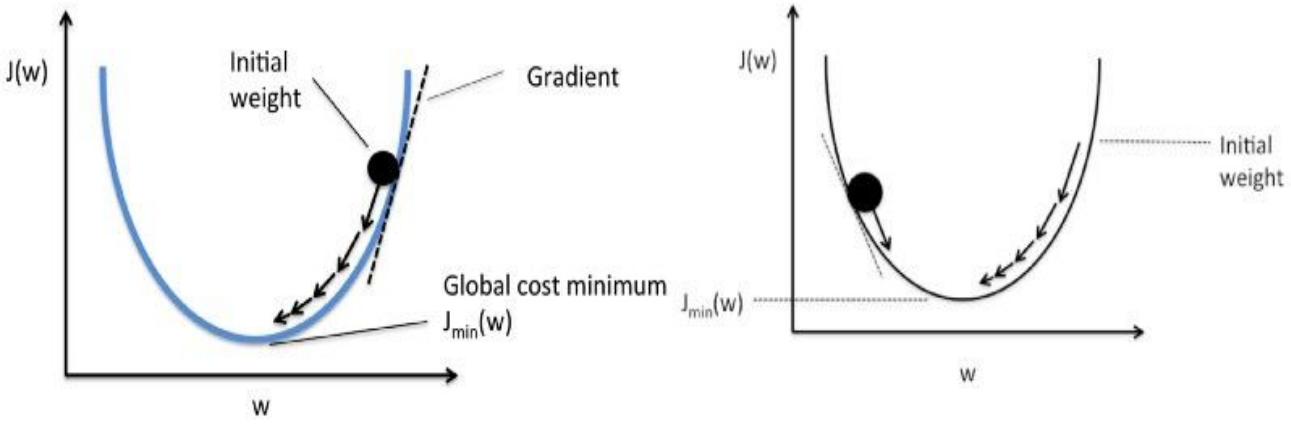


Figure 17: Using gradient descent in the cost function to adjust weights[40]

We draw a tangent line from the point and find the slope of the line. The slope identifies how much change is needed by taking the partial derivative of the cost function with respect to w.

$\frac{df}{dw}$ = slope = change in the function when w changes. If the derivative is positive (positive slope), the error rises as we increase the weights, so we should reduce the weight, and vice-versa. If the derivative is 0, the error has reached the stable point and we should not change the weight.[41]

The derivative of cost function with respect to w_0 is: $\frac{d}{dw_0} J(w_0, w_1) = \frac{1}{m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})$

This gives the change in w_0. Then, we multiply the change with the learning rate variable, called alpha (usually $\alpha = 0.01$). It signifies the magnitude of the step size to get the minimum value. To get the new w_0 value, we subtract the change from the earlier value of w_0.

40

Source:https://medium.com/deep-math-machine-learning-ai/chapter-1-2-gradient-descent-with-math-d4f2871af402

41 Source:
https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e

$$w_0 = w_0 - \alpha \frac{d}{dw_0} J(w_0, w_1)$$

Likewise, the derivative of cost function with respect to w_0 is: $\frac{d}{dw_1} J(w_0, w_1) = \frac{1}{m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}).\, x^{(i)}$

To get the new w_0 value, we subtract the change from the earlier value of $w_1 : w_1 = w_1 - \alpha \frac{d}{dw_1} J(w_0, w_1)$.

We apply the gradient descent to update the weights until the error rate is minimized; thus, iterating the process until convergence: $\{ \ldots \; w_j = w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}).\, x_j^{(i)} \; \ldots \}$

The learning rate α defines the speed at which a network updates its hyperparameters. The learning process is diminished when the learning rate is low, but it smoothly converges. Alternatively, the learning process is fast when the learning rate is high, but may diverge. [42]

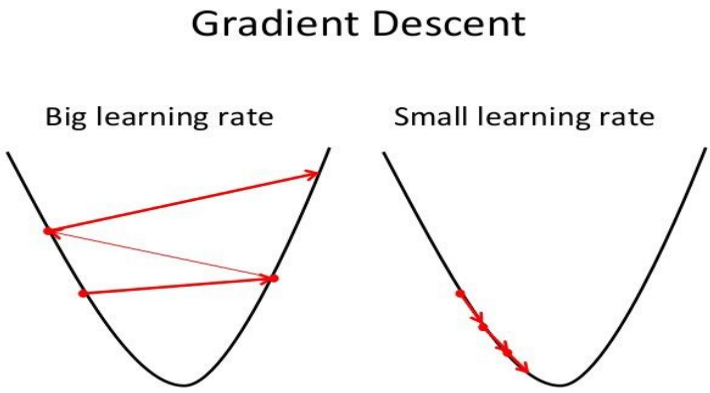


Figure 18: Learning rates α with different speeds[43]

Before building neural network models, I had scaled or normalized data to a standard format to improve the accuracy and speed of training set performance. First, I combined the six principal components with the housing starts variable into a dataframe and then scaled all the explanatory variables: $PC1$ to $PC6$. Since, I have

[42] Source: https://medium.com/deep-math-machine-learning-ai/chapter-7-artificial-neural-networks-with-math-bb71116948b

[43] Source: https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a

applied the logistical function, I scaled or normalized the the principal components to the interval (0, 1). Then, I further split the dataset into training and test set and created time slices using the same procedure outlined for cross-validation. There are 2 ways to normalize: Z-score normalization, and the min-max normalization. I have used the latter, wherein for each observation x_i, I calculated the following:

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)}$$

$min(x)$ and $max(x)$ are the minimum and maximum value of all the observations of the variable, respectively. It transforms all the scores into the range of $[0, 1]$. Both the normalization methods are sensitive to outliers.

After pre-processing, I created a neural network model with explanatory variables as $PC1$ to $PC6$ and response variable as housing starts. I added a decay parameter to regularize the network towards reducing bias and variance, so that the models do not overfit the data and generated a neural network of one layer. With the size ranging from 1 to 10, and decay from 0.1 to 4, the the test set MAPE is 10.31% and bias is -6.63%.

Figure 19 is a neural interpretation diagram (NID). The black lines indicate positive weights between layers and grey lines indicate negative weights. The thickness of line increases as the magnitude of each weight increases. The first layer consists of input variables whose nodes are labelled as $PC1$ to $PC6$ for the six explanatory variables. It consists of one hidden layer with eight hidden nodes labelled as $H1$ to $H8$, and a bias node labelled as B1. The last layer consists of the output layer labeled as O1. It shows the bias nodes which are connected to the hidden and output layers.

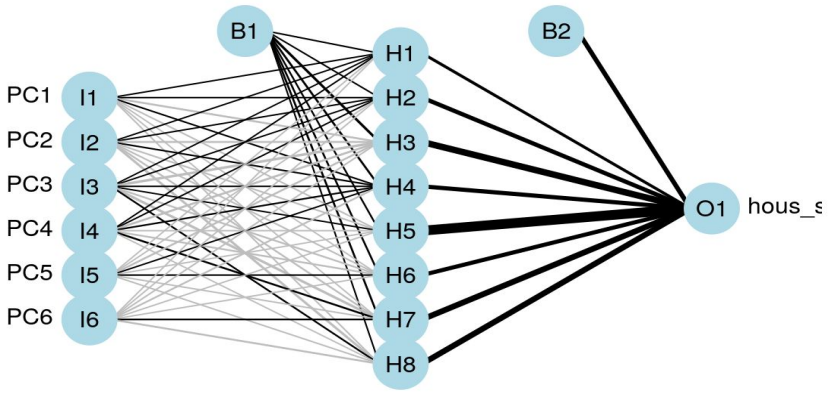


Figure 19 : Artificial Neural Networks with one hidden layer having eight nodes

3.3.7. Stacked Ensemble

After combining the results from multiple different models to predict housing starts, I created a stacked ensemble which is composed of the K-Nearest Neighbors regression, ridge regression, support vector regression and an artificial neural network. Then, I combined the validation-fold predictions from the component models into a new dataset as explanatory variables. Finally, I stacked a new model via ridge regression where response variable was housing starts and the explanatory variables were the predicted housing starts from each of the component models. Combining predictions from these independent, and less correlated models, reduces variance, and consequently, diminishes the overall test set bias and MAPE.

After obtaining the predictions in the test set and the error rates, I forecasted the values of housing starts from January to December, 2019. We need values of other variables to forecast housing starts in the future, so indirectly I had to predict the principal components as well. We cannot directly use machine learning algorithm for multi-step forecasting. I have made a multivariate multi-step time series forecasting model that forecasts housing over the next twelve months, given the recent and historic level of housing starts and other exogenous variables. So, I have used the ML models recursively to make multi-step forecasts. This procedure predicts one step at a time, feeds the predicted values in the model as inputs to

forecast the value for next month.[44] I had iterated this process twelve times to get values from January, 2019 to December, 2019.

4. Results

Checking the final models based on the cross-validated performance, I stacked them via ridge regression and obtained the predictions in the test set to compare them with the actual number of housing starts. The test set consists of values from July 2010 to December 2018. The test set MAPE from the stacked ensemble is 6.518% and the bias is -0.573%. The MAPE of the ensemble model is the lowest among the individual component models while the bias is closest to 0, indicating that the ensemble model is unbiased. This implies that on average the predicted value of housing starts deviate from the observed values by 6.518% housing starts per month. Among the econometric models, the $ARIMAX(2,1,3)$ has the lowest MAPE and percent. In contrast to the MAPE of the individual learning algorithms and the ensemble model, the econometric methods underperform.

Econometric and ML Models	**MAPE**	**Percent Bias**
$ARIMA(3,1,2)$	0.392	0.388
$ARIMAX(2,1,3)$	0.357	0.344
$ARMA(2,2)-GARCH(1,1)$	0.369	0.359
$VAR(3)$	0.355	0.346
KNN	0.279	-0.181
SVR	0.095	0.033
Ridge Regression	0.114	-0.050
ANN	0.103	-0.066
Ensemble of ML models	0.065	-0.005

[44] The predictions of each successive month are added at the end of the dataset, which help to make next month's prediction.

Table 8: Comparing the error rates (MAPE and percent bias) of the econometric, machine learning and ensemble models.

Figure 20 displays that the predicted and actual housing starts increase marginally over time in the test set.

Figure 20 : Actual and predicted housing starts from each of the component models in the test set

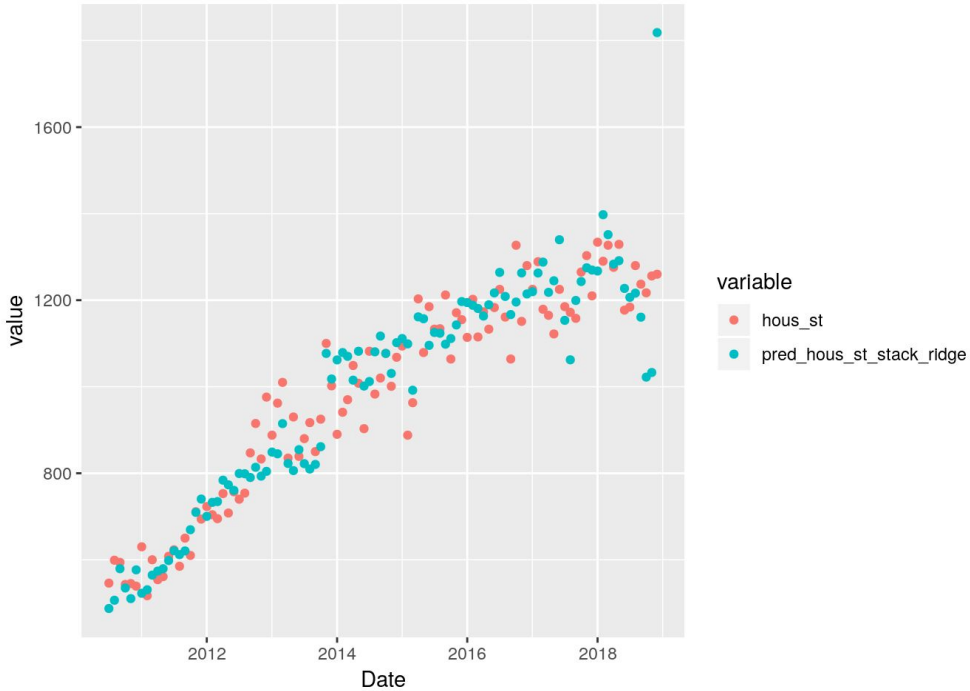


Figure 21 : Actual Housing starts from July 2010 to December 2018 and predicted Housing starts from the ensemble model

Based on the increasing order of MAPE, the models perform best in the following order: SVR, ANN, ridge regression, and KNN. This is also corroborated by the plot comparing actual and predicted values from individual models shows − KNN predictions are relatively flat prior to 2014 and substantially lower than the actual value of housing starts. Post-2014, the predictions are exceptionally high, albeit not constant as before. Equivalently, the predictions from ANN are analogous to a step-function: constant before 2012, then jumps up and then flattens between 2012 and late 2013, then surges again twice until it's gradient is positive after year 2015. The predictions from SVR and ridge regression seem to follow the same trend as the actual values of housing starts, but with a few outliers in the years 2017 and 2018.

Finally, Table 9 depicts the forecasts of housing starts (in 1000s of units) from the aforementioned and VARX models for the year 2019. The ARIMA model generates the highest forecasts where the values of housing starts are gradually rising until May, after which it fluctuates. The VARX model gives a more conservative forecast on a lower level, and follows a downward trend throughout the year, widening the gap between the values from VARX and those projected from the other three models. The forecasts from the ARMA − GARCH model are marginally lower than those of the ARIMA model but somewhat linearly climb at a very slow gradient. Finally, the forecasts from surges linearly, then plateaus to some extent, before fluctuating in June-July and then it surges. To sum up, except for VAR, other model forecasts are similar.

Year 2019	$ARIMA(3,1,2)$	$ARIMAX(2,1,3)$	$VARX(3)$	ARMA(2,2) − GARCH(1,1)
January	1263.583	1180.307	1216.2202	1255.681
February	1260.987	1225.467	1119.0588	1258.140
March	1274.743	1227.548	1115.0081	1259.314
April	1270.308	1228.654	1083.1347	1260.461
May	1277.965	1224.715	1041.3320	1261.594
June	1268.353	1222.757	1012.1100	1262.714
July	1271.857	1182.278	982.1425	1263.822
August	1262.992	1200.415	951.2494	1264.91

September	1268.083	1190.270	922.9862	1265.999
October	1262.739	1195.869	896.4428	1267.069
November	1269.046	1209.779	870.7747	1268.127
December	1264.631	1221.963	846.4608	1269.173

Table 9: Forecasts of housing starts (in 1000s) of units using the four econometric models

Finally, Table 10 has forecasts from the machine learning models. For all models, except for KNN, the forecasts appear to be cointegrated as they have the same stochastic trends. The lowest forecasts correspond to the SVR model with the lowest MAPE, followed by ridge, ANN and the ensemble model. They follow a downward trend and reach the nadir in October, after which they skyrocket in October-November. The forecasts from KNN are flat till September, then stumble for a month, before spiking and converging with the forecasts from other models. The two plots in the Appendix gives diagrammatically represents the forecasts from both econometric and ML models

Year 2019	KNN	Ridge	ANN	SVR	Ensemble
January	1452.5	1208.5	1407.7	1157.8	1267.4
February	1452.5	1274.4	1767	1233.1	1397.6
March	1452.5	1231	1698.3	1181.5	1351.4
April	1452.5	1222.2	1434.3	1174.5	1283.3
May	1452.5	1227.5	1454.4	1179	1291.2
June	1452.5	1170	1345.7	1115.3	1227.2
July	1452.5	1137	1350.4	1078.7	1206.5
August	1452.5	1147.2	1360.4	1092.2	1216.3
September	1452.5	1070.4	1341.2	1006.5	1160.7
October	1356	956.1	1157.6	886	1022.1
November	1356	942.6	1238	872.7	1032.8
December	1770.5	1775.1	1823.3	1812.8	1818.4

Table 10: Forecasts of housing starts (in 1000s) of units using the four ML and ensemble models

5. Limitations and Future Research

First, I will discuss the macroscopic challenges of using machine learning models in a low frequency data. Generally, machine learning algorithms work on large sized data, also known as "big data." Examples of high frequency data are geospatial, transactional, behavioral, geo-spatial etc. Personal devices and digital products often compile these data on vast databases containing wide array of variables. Larger time series in finance are datasets measuring daily global index prices, commodities, foreign exchange, CBOE Volatility index (VIX)[45] and the Dow Jones weekly returns[46], among others. Traditional economic indicators in macroeconomics pertaining to fiscal, labor, monetary and trade statistics which are mostly available on a quarterly or annual basis. This starkly contrasts with high frequency variables as very less data is accumulated. Moreover, splitting the dataset into training and test sets further shortens its length, possibly making results from the learner algorithms less reliable. The dataset for this analysis constituted only 511 observations, which is very small. Analyzing in a bigger dataset will yield better results as more information from the variables are considered and tested in the validation sets. In short series, insufficient data is withheld for testing the model, so applying cross-validation becomes hard. Deep learning architecture usually performs better in large datasets, whereas time series models such as VARX and ARIMA cannot handle huge datasets at a time. Well-suited for short-term forecasts, ARIMA models cannot accurately predict long term values as they depend on several external factors. For example, the movement of economic indicators such as mortgage rate, real estate loans, employment and inflation rate, among others will substantially influence housing supply and housing starts. Just by incorporating the lagged effects, the simplistic and restricted ARIMA model cannot fully capture changes in all dimensions, resulting in higher MAPE than that produced from VARX and ARIMAX models in the analysis. Unless the underlying dataset is simple enough – time series are stationary and variables fluctuate very less, machine learning approaches will relatively perform more efficiently.

[45] The **CBOE Volatility Index (VIX)** measures market expectations of near-term volatility conveyed by S&P. This time series dataset comprises of daily open, close, high and low.

[46] The **Dow Jones Industrial Average (DJIA) returns** dataset composes of weekly stock returns in percentage. We can train algorithms in this dataset to infer which stock will generate the maximum return in the following week.

Now, I will deliberate on one of the microscopic deficiencies arising from applying the sigmoid activation function in the artificial neural networks model. From the graph of sigmoid function, the response variable responds very gradually as an explanatory variable changes, so the gradient sluggishly reduces. This causes the problem of "vanishing gradient," whereby the gradient is small or has disappeared. In this scenario, the network either does not learn further, or is exceedingly slow. We can use other types of activation functions such as tangent hyperbolic function .The tanh function is very similar to a sigmoid function:

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1 = 2\,sigmoid(2x) - 1, \quad \forall\, f(x) \in (-1, 1)$$

The tanh function has a stronger gradient compared to that of the sigmoid function, but the problem of "vanishing gradient" still persists, so we can use the rectified linear unit function (ReLu) activation function. Usually ARIMA models produce accurate results when the dataset has seasonality, trends or autocorrelation. However, multitudinous factors such as competitor's behavior, economic and climatic phenomenon, media effects or other short term fluctuations can complicate the nature of a time series. These factors manifest particularly when the analysis entails forecasting at a granular level− minute-by-minute, hourly, daily, weekly, etc. Therefore, we can implement more sophisticated methods such Long Short Term Memory (LSTM), which is a convoluted recurrent neural network (RNN) architecture that can learn forecast long sequences. The downside that is it tricky to configure and requires considerable preparation to properly format the data before the model can start learning.

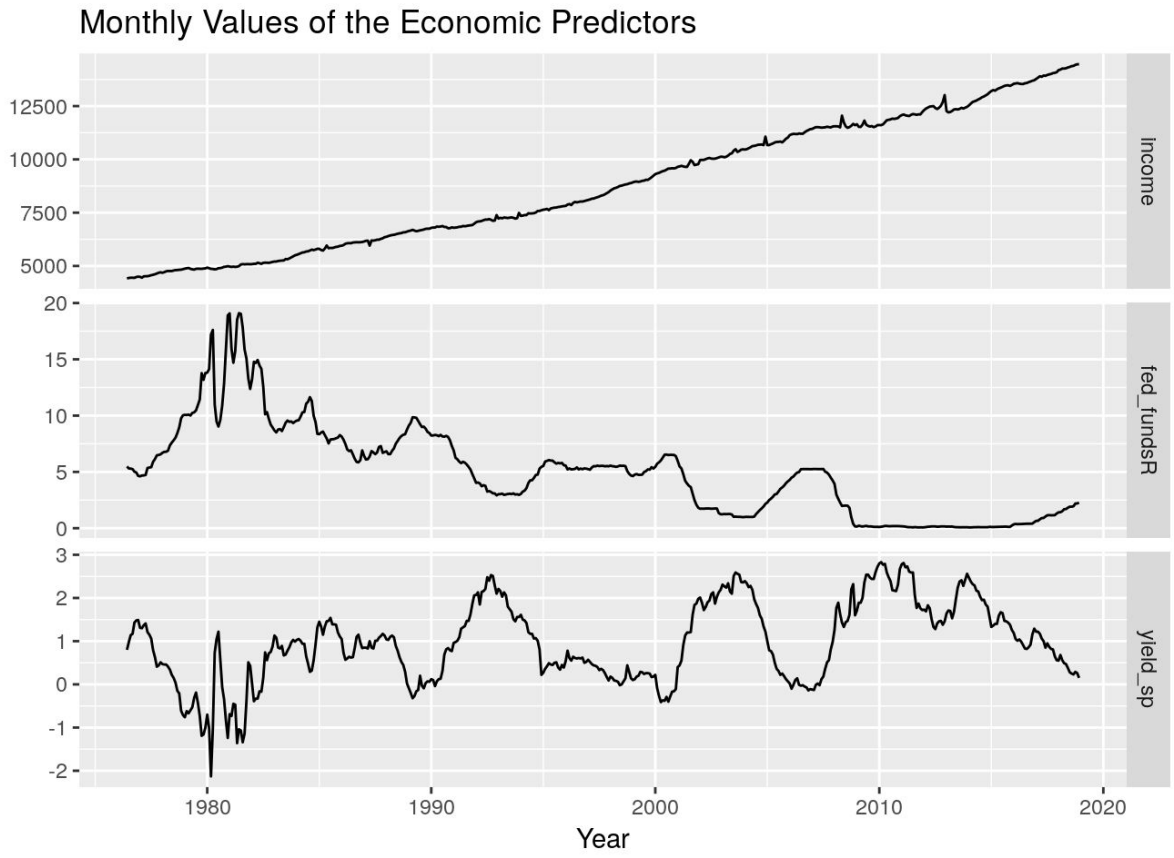
We can also do scenario based forecasting. Typically performed in risk management, we can produce baseline forecasts, create multiple "what if" situations for stress-testing, and generate forecasts for each scenario. These can help companies hedge both systemic risks arising from a downturn in business cycle such as a recession, and idiosyncratic risks emanating from within the specific industry. Finally, in this paper, I have also explained mathematically how to forecast from ARIMA models, but not from other models. It would be useful to know how the process works for other time series and machine learning models.

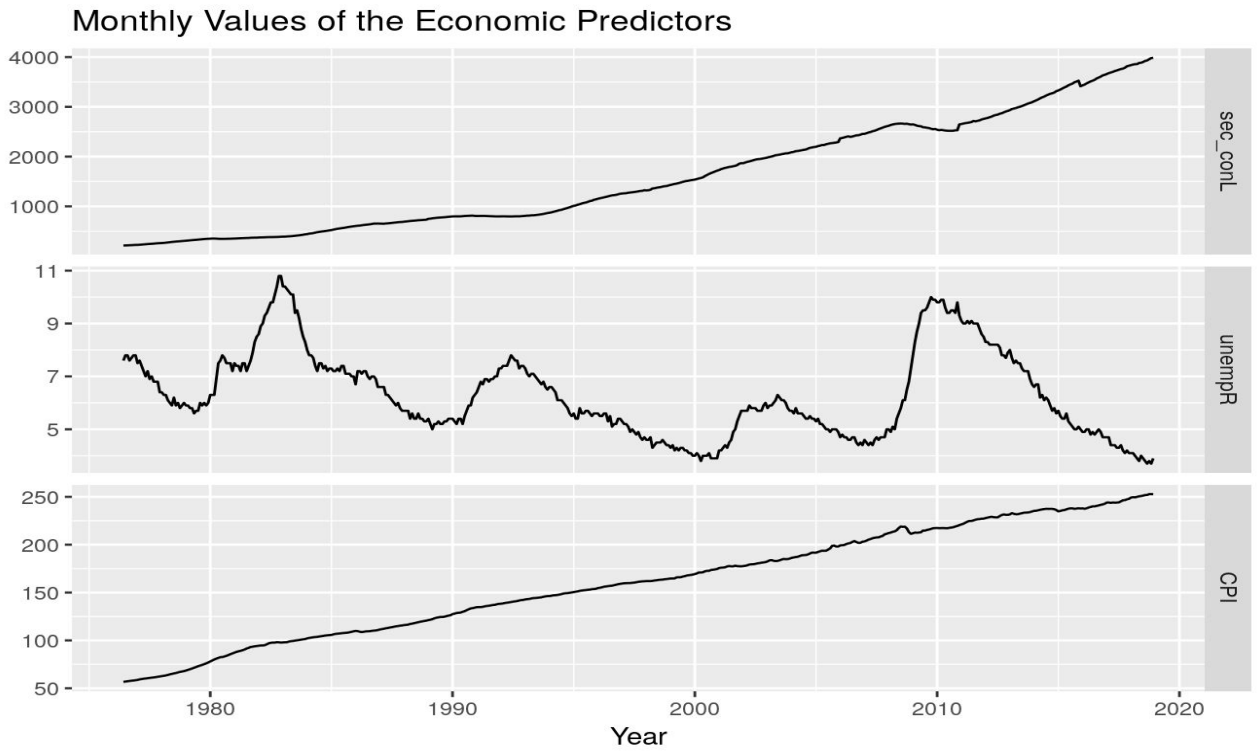
6. Conclusion

In this empirical analysis, I have conducted time series analysis and forecasted housing starts using various econometric and machine learning models. From the results, I observed that the ML models outperform the econometric models. The ensemble ML model had a MAPE of 6.51 percent compared to the MAPE of $ARIMAX(2,1,3)$ which was 35.7 percent − lowest among all other econometric models. This suggests that economists engaged in forecasting macroeconomic variables should explore forecasting from ML based models. These models can discover complex non-linear relationships in the data, without assuming anything about the exogenous factors. They can determine the relative importance of each variable without being affected by multicollinearity. However, we also have to be prudent when deciding which models to use, particularly, as they cannot be interpreted and can overfit in the training sets.

In both time series and ML models, performance of forecasts differ depending on the dataset. As we often assume that past patterns can indicate the behavior of a series, they are projected. Consequently, if the patterns continue, the forecasts will be precise, but if the patterns abruptly deflect, the projections may heavily differ from the actual value, as noticed from the flat KNN predictions. This creates a "black swan" event wherein the event deviates beyond its generally expected path and is hard to project. So, we have to retrain the model repeatedly to account for newer information. ML models can capably model any type of patterns, compared to time series forecasting methods − where we have first ensure homoskedastic errors and same probability distribution throughout the series. Hence, I have conducted several tests such as (G)ARCH for conditional heteroskedasticity, and Johansen test of cointegration, among others. Thereafter, I have forecasted housing starts.

Appendix


Monthly Values of the Economic Predictors
12500
10000
7500
5000
Income
20
15
10
5
0
fed_fundsR
3
2
1
0
-1
-2
yield_sp
1980
1990
2000
2010
2020
Year



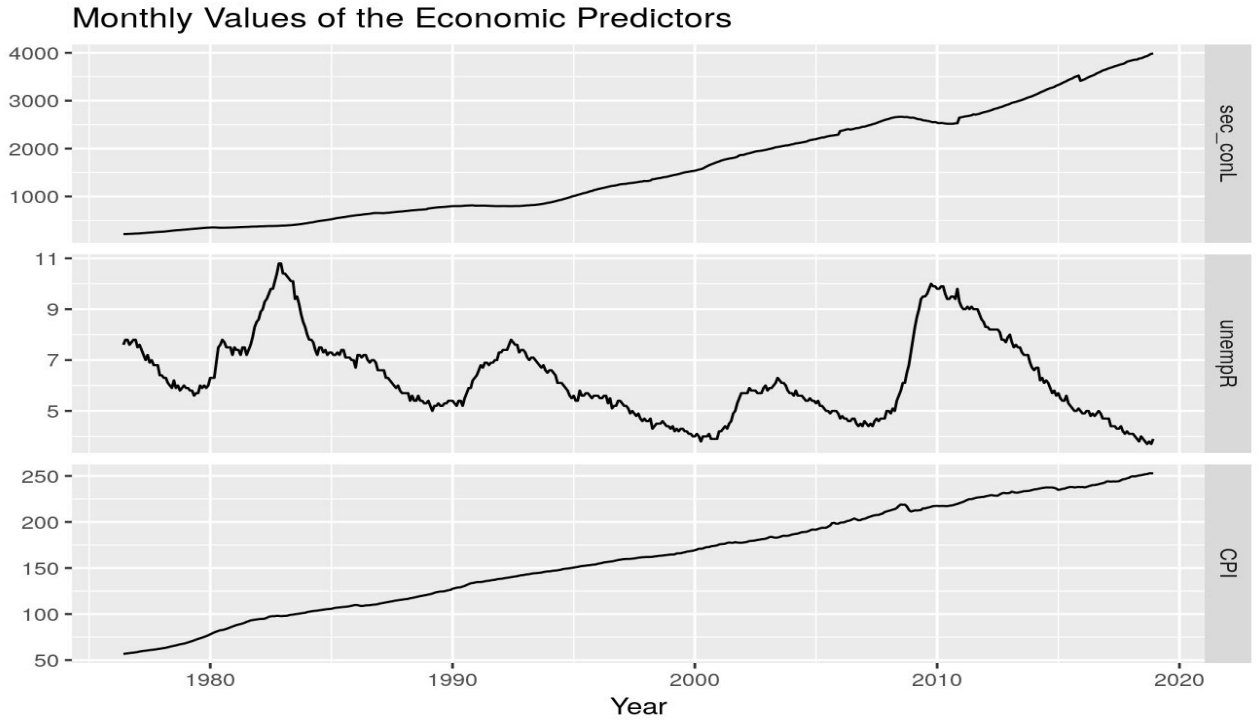


Figure 1: The non-stationary plots show the values of economic variables over time (June 1976-December 2018)

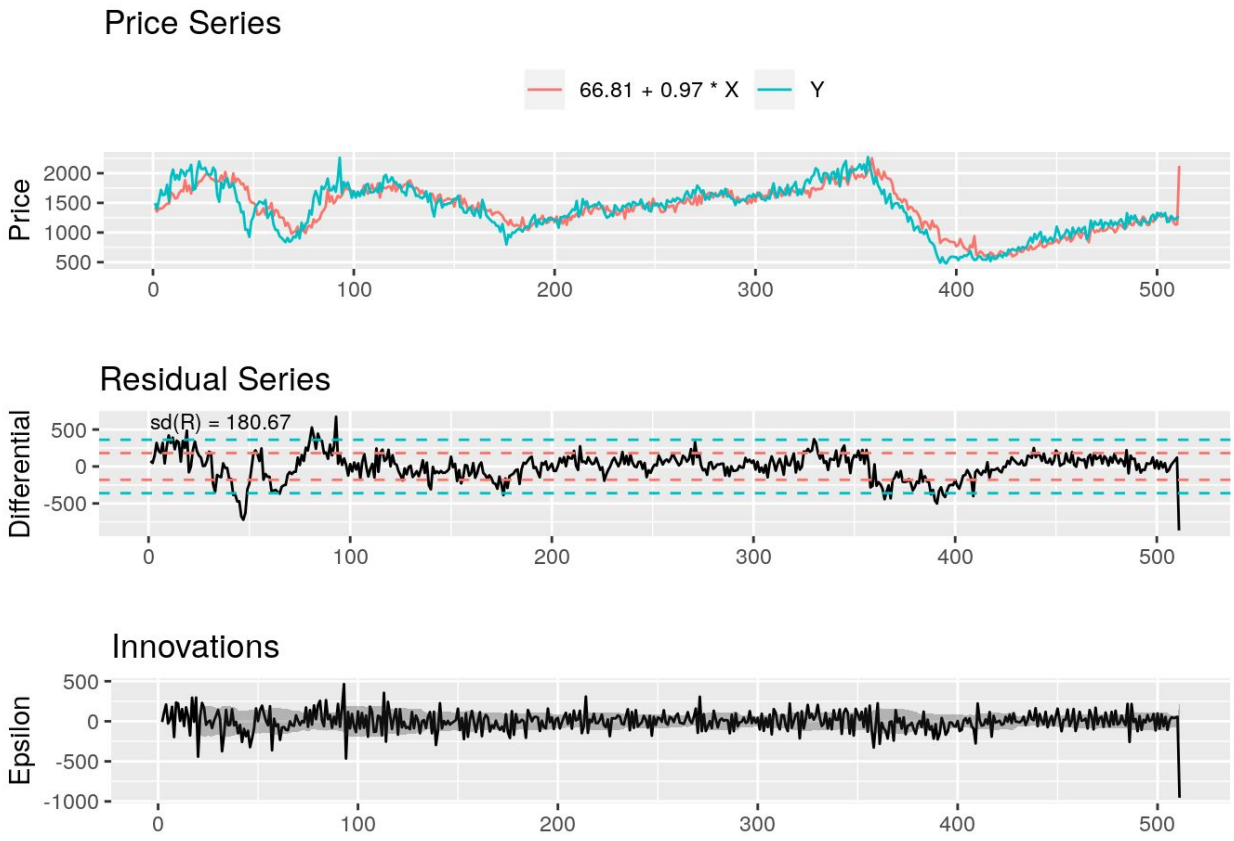


Figure 2: Plots showing the equation, residuals and innovations from the cointegration model between housing starts and private houses completed

hous_st	hous_st	pvt_house_comp	mortgR	house_supply
ECT1	-0.0268(0.0425)	0.3199(0.0347)	0.0003(0.0001)	0.0003(0.0002)
ECT2	-0.0363(0.0375)	-0.3288(0.0307)	-0.0002(9.8e-05)	-0.0002(0.0002)
Intercept	163.2674(34.76)	-9.3985(28.4199)	-0.2161(0.0905)	-0.2639(0.1756)
hous st_{t-1}	-0.5518(0.0587)	-0.2430(0.0480)	1.7e-06(0.0002)	-0.0003(0.0003)
hous st_{t-2}	-0.3224(0.0576)	-0.2111(0.0471)	-0.0001(0.0001)	1.2e-05(0.0003)
hous st_{t-3}	-0.1176(0.0489)	-0.1207(0.0400)	-0.0002(0.0001)	-5.1e-05(0.0002)

$mortgR_{t-1}$	-43.9864(18.0267)	2.5715(14.7387)	0.5653(0.0469)	0.6693(0.0911)
$mortgR_{t-2}$	-11.1202(19.6690)	7.4237(16.0815)	-0.3211(0.0512)	-0.0641(0.0993)
$mortgR_{t-3}$	-42.0914(18.5365)	-1.7610(15.1556)	0.1238(0.0482)	0.1072(0.0936)
pvt hous comp$_{t-1}$	0.0496(0.0596)	-0.4583(0.0487)	0.0002(0.0002)	0.0008(0.0003)
pvt hous comp$_{t-2}$	-0.0044(0.0663)	-0.2872(0.0542)	0.0002(0.0002)	0.0007(0.0003)
pvt hous comp$_{t-3}$	-0.0131(0.0597)	-0.1894(0.0488)	0.0001(0.0002)	0.0002(0.0003)
hous supply$_{t-1}$	-13.1016(10.4502)	-4.5648(8.5442)	-0.0506(0.0272)	-0.3478(0.0528)
hous supply$_{t-2}$	-21.3106(10.5554)	-5.8704(8.6302)	-0.0625(0.0275)	-0.2075(0.0533)
hous supply$_{t-3}$	-14.1123(9.9092)	-4.0074(8.1018)	-0.0483(0.0258)	0.0047(0.0501)

Table 1: Coefficients of the VECM

```
================================================================================
                                        Dependent variable:
                                ---------------------------------------------
                               hous_st, pvt_house_comp, mortgR, house_supply
                                   (1)         (2)          (3)          (4)
--------------------------------------------------------------------------------
hous_st.l1                      0.443***   0.097**     0.0002**     0.00003
                                (0.046)    (0.038)     (0.0001)     (0.0002)

pvt_house_comp.l1                0.040     0.283***     0.0001       0.001*
                                (0.064)    (0.053)     (0.0002)     (0.0003)

mortgR.l1                       -32.268*    0.294       1.512***     0.639***
                                (17.514)   (14.556)     (0.045)      (0.087)
```

house_supply.l1	-29.253***	-3.226	-0.018	0.661***
	(9.570)	(7.954)	(0.025)	(0.047)
hous_st.l2	0.222***	0.044	-0.0001	0.0003
	(0.050)	(0.041)	(0.0001)	(0.0002)
pvt_house_comp.l2	-0.039	0.250***	0.00001	-0.0002
	(0.063)	(0.053)	(0.0002)	(0.0003)
mortgR.l2	-0.614	6.542	-0.802***	-0.639***
	(28.479)	(23.670)	(0.074)	(0.141)
house_supply.l2	-7.547	1.884	-0.003	0.139**
	(11.436)	(9.505)	(0.030)	(0.056)
hous_st.l3	0.227***	0.137***	-0.0001	-0.0001
	(0.048)	(0.040)	(0.0001)	(0.0002)
pvt_house_comp.l3	0.025	0.183***	-0.0001	-0.0005
	(0.060)	(0.050)	(0.0002)	(0.0003)
mortgR.l3	30.923*	-5.984	0.274***	0.013
	(17.955)	(14.923)	(0.047)	(0.089)
house_supply.l3	13.121	5.803	0.032	0.215***
	(9.800)	(8.146)	(0.025)	(0.048)
const	321.561***	-41.733	0.022	-0.805***
	(62.661)	(52.080)	(0.162)	(0.309)
trend	-0.193***	0.034	-0.0003	0.001**
	(0.070)	(0.058)	(0.0002)	(0.0003)

Observations	508	508	508	508
R2	0.944	0.954	0.995	0.922
Adjusted R2	0.943	0.953	0.994	0.919
Residual Std. Error (df = 494)	97.370	80.928	0.252	0.481
F Statistic (df = 13; 494)	642.237***	794.994***	6,916.869***	446.179***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2: R output: VARX(3) model

b->h1	i1->h1	i2->h1	i3->h1	i4->h1	i5->h1
6.9973094	2.7541821	1.9130187	0.1593454	2.6374881	-0.7964275
i6->h1	b->h2	i1->h2	i2->h2	i3->h2	i4->h2
-1.0175347	6.6981651	3.0004355	2.4396672	0.8289042	2.6990313
i5->h2	i6->h2	b->h3	i1->h3	i2->h3	i3->h3
-0.2761348	-1.7851033	60.6841860	-30.0739451	-30.8723839	-67.0854827
i4->h3	i5->h3	i6->h3	b->h4	i1->h4	i2->h4
-41.9800897	-28.9049180	-8.6273998	35.2036697	9.3985052	10.3136393
i3->h4	i4->h4	i5->h4	i6->h4	b->h5	i1->h5
6.6504071	14.3804396	2.4946905	-20.5704051	17.0209935	-8.9758188
i2->h5	i3->h5	i4->h5	i5->h5	i6->h5	b->h6
-13.2607918	0.8004982	-6.0931299	-1.1598040	-2.6605575	25.1301763
i1->h6	i2->h6	i3->h6	i4->h6	i5->h6	i6->h6
-1.8421162	-12.1820600	-21.3783462	-21.6272392	4.7187724	-2.5071266
b->h7	i1->h7	i2->h7	i3->h7	i4->h7	i5->h7
29.6180954	-0.3854124	-21.5384474	-64.6874797	29.0406152	-4.1728956
i6->h7	b->h8	i1->h8	i2->h8	i3->h8	i4->h8
16.9318749	5.3045046	-7.3714461	-54.5435554	18.5750584	-0.3864044
i5->h8	i6->h8	b->o	h1->o	h2->o	h3->o
-5.6130632	-28.4729414	180.5661546	87.3308392	168.1559459	298.7879217
h4->o	h5->o	h6->o	h7->o	h8->o	
169.1409014	503.7799004	162.6350091	257.4076078	253.5629605	

Table 3: R output: Weights of the first ANN model with 8 nodes in the 1st hidden layer (size = 8)

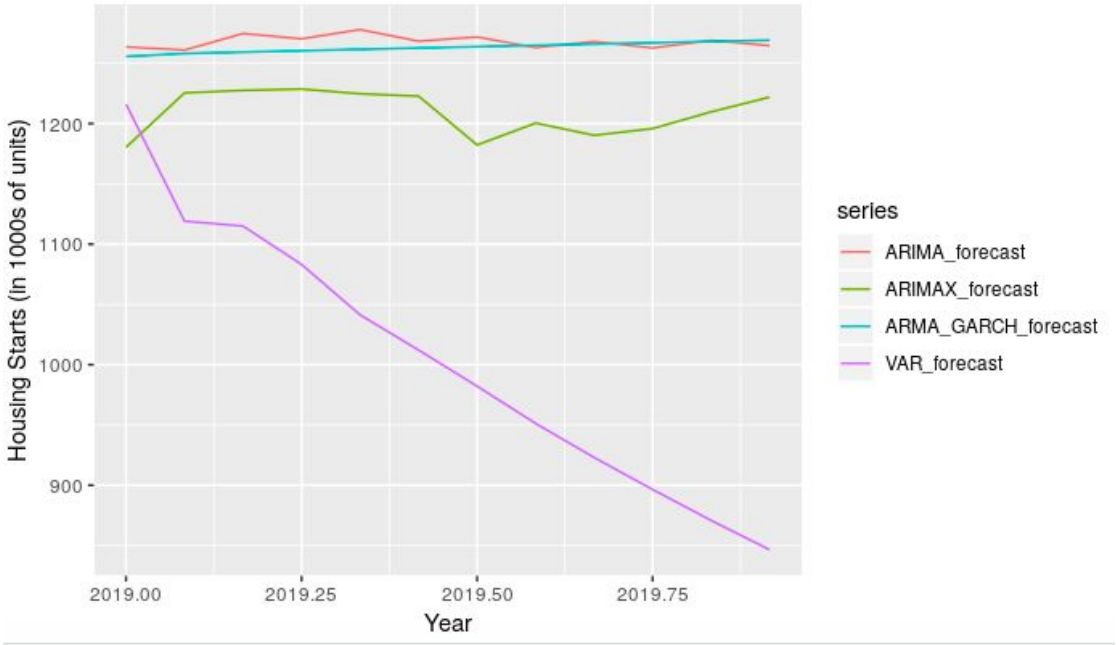


Figure 4: The forecasts from ARIMA, ARIMAX, VARX and ARMA - GARCH models for year 2019

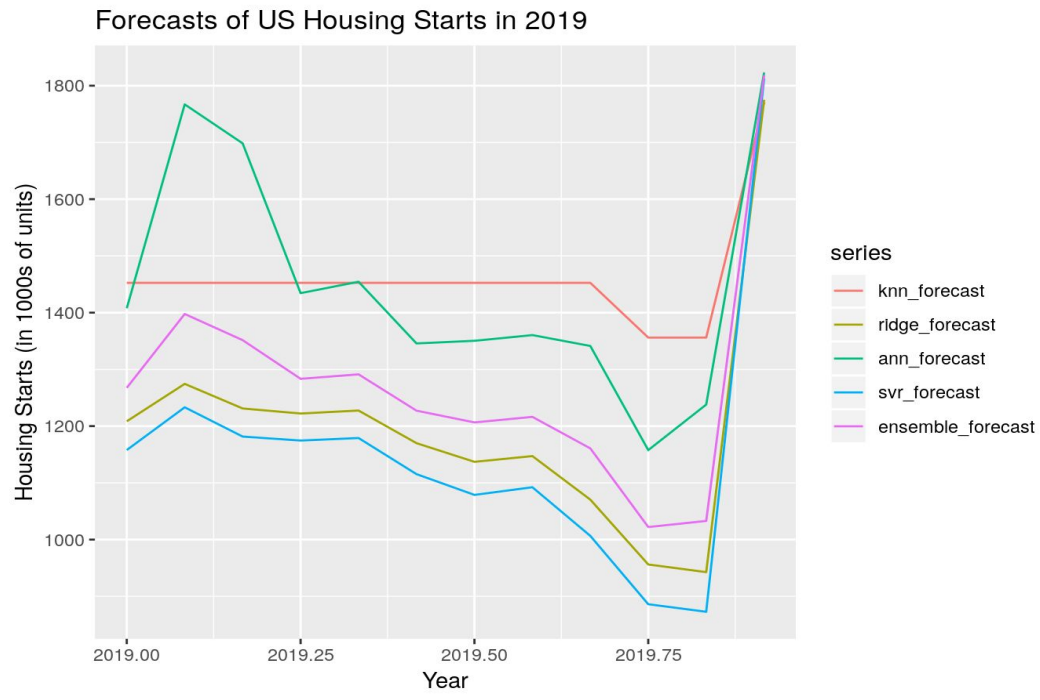


Figure 5: The forecasts from KNN, ANN, Ridge, SVR and ensemble models for year 2019

Bibliography

1. Brady, Eugene A. "Forecasting Housing Starts." *Business Economics* (1971): 18-22.
2. Coccari, Ronald L. "Time series analysis of new private housing starts." *Business Economics* (1979): 95-109.
3. Hansen, James V., James B. McDonald, and Ray D. Nelson. "Some evidence on forecasting time-series with Support Vector Machines." *Journal of the Operational Research Society* 57, no. 9 (2006): 1053-1063.
4. Henly, Samuel, and Alexander Wolman. "Housing and the Great Recession: a VAR accounting exercise." (2011).
5. Joseph, Anthony, and Maurice Larrain. "Housing Starts Forecast of Retail Sales through the 2007-2009 Recession." *Procedia Computer Science* 12 (2012): 271-275.
6. Khalafallah, Ahmed. "Neural network based model for predicting housing market performance." Tsinghua Science and Technology 13, no. S1 (2008): 325-328.
7. Kemme, David M., and Saktinil Roy. "Did the recent housing boom signal the global financial crisis?." *Southern Economic Journal* 78, no. 3 (2012): 999-1018.
8. Kawaller, Ira G., and Timothy W. Koch. "Housing as a Monetary Phenomenon: Forecasting Housing Starts Using the Monetary Aggregate Targets." *Business Economics* (1981): 30-35.
9. Qiu, Xueheng, Le Zhang, Ye Ren, Ponnuthurai N. Suganthan, and Gehan Amaratunga. "Ensemble deep learning for regression and time series forecasting." In *Computational Intelligence in Ensemble Learning (CIEL), 2014 IEEE Symposium on*, pp. 1-6. IEEE, 2014.
10. Gu, Jirong, Mingcang Zhu, and Liuguangyan Jiang. "Housing price forecasting based on genetic algorithm and support vector machine." *Expert Systems with Applications* 38, no. 4 (2011): 3383-3386.
11. Dua, Pami, Stephen M. Miller, and David J. Smyth. "Using leading indicators to forecast US home sales in a Bayesian vector autoregressive framework." *The Journal of Real Estate Finance and Economics* 18, no. 2 (1999): 191-205.
12. Hsu, Ming-Wei, Stefan Lessmann, Ming-Chien Sung, Tiejun Ma, and Johnnie EV Johnson. "Bridging the divide in financial market forecasting: machine learners vs. financial economists." *Expert Systems with Applications* 61 (2016): 215-234.
13. Pavlyshenko, Bohdan M. "Machine-Learning Models for Sales Time Series Forecasting." *Data* 4, no. 1 (2019): 15.

14. Makridakis, Spyros, Evangelos Spiliotis, and Vassilios Assimakopoulos. "Statistical and Machine Learning forecasting methods: Concerns and ways forward." *PloS one* 13, no. 3 (2018): e0194889.
15. Engle, Robert. "GARCH 101: The use of ARCH/GARCH models in applied econometrics." *Journal of economic perspectives* 15, no. 4 (2001): 157-168.
16. Park, Byeonghwa, and Jae Kwon Bae. "Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data." *Expert Systems with Applications* 42, no. 6 (2015): 2928-2934.
17. Ahmed, Nesreen K., Amir F. Atiya, Neamat El Gayar, and Hisham El-Shishiny. "An empirical comparison of machine learning models for time series forecasting." *Econometric Reviews* 29, no. 5-6 (2010): 594-621.
18. Jung, Jin-Kyu, Manasa Patnam, and Anna Ter-Martirosyan. *An Algorithmic Crystal Ball:Forecasts-based on Machine Learning*. International Monetary Fund, 2018.
19. Luo, H. and Wang, S., 2017. Based on the PCA-ARIMA-BP hybrid model of stock price prediction research. *ANZIAM Journal*, *58*, pp.162-178.
20. Bergmeir, C., Hyndman, R.J. and Koo, B., 2018. A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, *120*, pp.70-83.
21. James, Gareth, Daniela Witten, Trevor J. Hastie, and Robert J. Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. New York: Springer, 2017.
22. "11.1: ARCH/GARCH Models." 11.1: ARCH/GARCH Models | STAT 510. Accessed April 26, 2019. https://newonlinecourses.science.psu.edu/stat510/node/85/.
23. Hyndman, Rob. "Forecasting: Principles and Practice." *8.8: Forecasting*, otexts.com/fpp2/arima-forecasting.html.
24. "11.1: ARCH/GARCH Models." 11.1: ARCH/GARCH Models | STAT 510. Accessed April 26, 2019. https://newonlinecourses.science.psu.edu/stat510/node/85/.
25. EViews Help. Accessed April 26, 2019. http://www.eviews.com/help/helpintro.html#page/content/VAR-Vector_Error_Correction_(VEC)_Models.html.
26. Brecque, Charles. "Demystifying Hyper-Parameter Tuning." Towards Data Science. October 01, 2018. Accessed April 26, 2019. https://towardsdatascience.com/demystifying-hyper-parameter-tuning-acb83af0258f.
27. Kumar, Rishav. "Understanding Principal Component Analysis." Medium. January 02, 2018. Accessed April 26, 2019. https://medium.com/@aptrishu/understanding-principle-component-analysis-e32be0253ef0.

28. Harrison, Onel. "Machine Learning Basics with the K-Nearest Neighbors Algorithm." Towards Data Science. September 10, 2018. Accessed April 26, 2019. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.
29. Bhattacharyya, Indresh. "Support Vector Regression Or SVR." Medium. June 29, 2018. Accessed April 26, 2019. https://medium.com/coinmonks/support-vector-regression-or-svr-8eb3acf6d0ff.
30. Sanjeevi, Madhu, and Mady. "Chapter 1.2: Gradient Descent with Math." Medium. September 26, 2017. Accessed April 26, 2019. https://medium.com/deep-math-machine-learning-ai/chapter-1-2-gradient-descent-with-math-d4f2871af402.
31. MOAWAD, Assaad. "Neural Networks and Backpropagation Explained in a Simple Way." Medium. February 01, 2018. Accessed April 26, 2019. https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e.
32. Hanck, Christoph, Martin Arnold, Alexander Gerber, and Martin Schmelzer. "Introduction to Econometrics with R." 16.4 Volatility Clustering and Autoregressive Conditional Heteroskedasticity. March 12, 2019. Accessed April 16, 2019. https://www.econometrics-with-r.org/16-4-volatility-clustering-and-autoregressive-conditional-heteroskedasticity.html
33. Soni, Devin, and Devin Soni. "Supervised vs. Unsupervised Learning." Towards Data Science. March 22, 2018. Accessed April 16, 2019. https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d.
34. "Federal Reserve Economic Data | FRED | St. Louis Fed." FRED. Accessed May 01, 2019. https://fred.stlouisfed.org/.
35. "Stack Exchange." Hot Questions - Stack Exchange. Accessed April 30, 2019. https://stackexchange.com/.