Real Estate Price Prediction and Seasonal Forecasting

By:
Angad Singh
Sonali Johari
Pranav Prajapati
Gunjan Misra

Introduction

In today's world, everyone is trying to buy a house either for the purpose of investing or starting a family in their new home. According to the US Census Bureau, there were 127 million houses in the United States of America. [1] The price of these houses tends to fluctuate based on a number of reasons than just its location and total square footage. This project is aimed at trying to predict the prices of homes of a given region with a number of features apart from just its location or size. It includes some features like the number of rooms, the building type, the building quality, the year the house was built, etc.

Being able to predict the price of a house can be beneficial for a number of reasons, for example, one can understand what is the price of the house at which they should buy a new home, one can also understand similarly what the exact price for selling the same home should be. It will shed light onto what the important features of the homes are because of which a house might be priced higher or lower on the market. Along with prediction, we planned to see the general trends of the real estate industry as the years have passed on and how the values of homes and houses across the United States have changed over time. It can help a person figure out the right time for investing or selling their homes along with knowing the best times during particular seasons and months of the year.

The project was divided into two parts which used two different data provided openly by Zillow, they are the leading real estate and a rental marketplace which is being used increasingly by people all over the country in their search of new houses. The first part of the project was to predict the prices of the houses based on the many features provided in the data and to check for the scalability of data on a local machine as well as the clusters using various regression techniques.[15] [16] The second part was to seasonally forecast the prices of the house using time series data. [17]

Data Preparation

The data was collected from the Kaggle website and was provided by Zillow for a competition they had organized. [2] The data consisted of 2 million rows and 84 features. Some of these features can be seen in Figure 1 below. The data was for the state of California. The data for the seasonal forecasting was the Zillow Home Value Index which is a measure adjusted seasonally of the estimated price of a house across a given area and type.

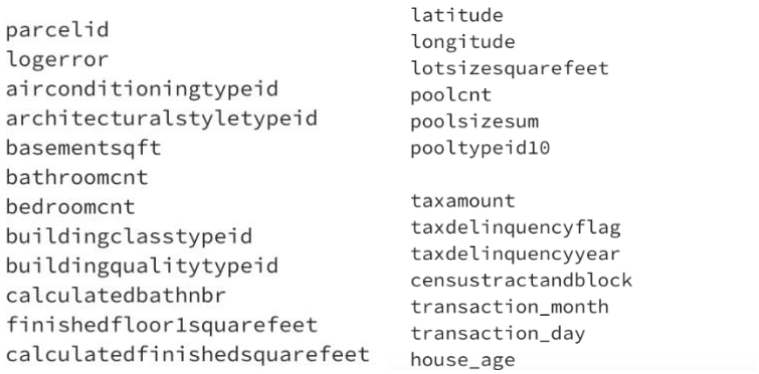

Figure 1: Dataset Features

One of the most important parts of working with data is making sure that the data is suitable for running an analysis on it. The process of modifying the data is known as data cleaning which involves detecting and removing corrupt records along with identifying incomplete and missing values and appropriately replacing them or discarding them. [3] In the data provided by Zillow, it was observed that there were a lot of flag columns or flag features. These are the columns in which if there is a value present, it indicates the existence of the item and a missing value indicates the absence of the same item. [4]

A lot of the features in the dataset like hashottuborspa, fireplaceflag and taxdelinquencyflag were flag columns. The NaN values for these columns were replaced with a '0' to indicate that these features were not present in the house and other values were changed to '1' to indicate that the house did come with those features. There were a huge number of columns having missing values which were not flag values and hence replacing them with a 0 did not make much sense. These values were replaced with either mean values or random values from the same column to try to maintain some uniformity in the data.

Apart from taking care of missing values and flag columns, we decided to create new features from existing features to reduce the number of features. This would help in reducing the overall size of the data which would subsequently reduce the amount of time the model takes to run. Some of the features created were as follows:

- House Age: We calculated the value by subtracting the year the house was built from 2017 i.e. when the data was collected.
- Total Rooms: Counting all the number of rooms instead of keeping different columns for the number of bedrooms, bathrooms, etc.
- Location of House: The data could be divided into three distinct locations based on the FIPS code.
- Seasons: This value was derived from the data of the transaction taking place

After creating new features, we performed exploratory analysis on the data to see what we could analyze and observe from an overview of the data.

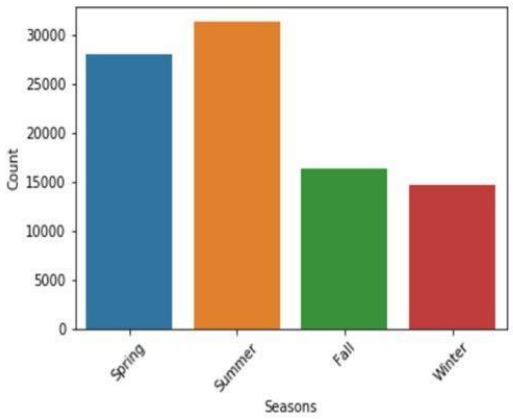

Figure 2a: Transactions by season

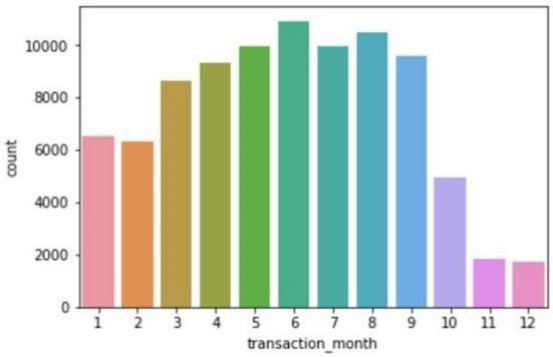

Figure 2b: Transactions by month

Feature Selection

Before running the regression models chosen, we decided to find out which of the features of the dataset were the most important features. This process was performed by XGBoost [8], also known as eXtreme Gradient Boosting. It is an ensemble model of gradient boosting trees which is also a weighted sum of weak learners. One of the most important features of this algorithm and one of the main reasons why we chose this algorithm is that it has an inbuilt function which provided us with the features in the order of their importance. The algorithm has three important metrics according to which it chose the importance of the features: [5]

- Weight: It represents the number of time a feature occurs in the trees of the model. For example, if a feature A occurs in 2,3,5 splits in tree numbers 1,2,3 respectively then the weight would be 2+3+5 = 10. The final weight is calculated as its percentage weight over weights of all the other features.
- Cover: It represents the relative number of observations of this feature. For example, a feature A is used to decide for the leaf node for 2,10,3 observations in tree numbers 1,2,3 and that means the count will be 15. The counts will be calculated for all the features present and expressed as a percentage.
- Gain: This is the measure which helps us understand the relative contribution of each feature in the tree. Higher the value, greater is the importance of the feature. It is the improvement in accuracy which is brought about by adding a feature to a split. If there are wrongly classified elements and a feature A creates two branches that help improve the accuracy of the model, the feature is given more importance.

The following figure shows the most important features selected by XGBoost. The most important features were Value Ratio, Tax Score and Land Tax Value Dollar Count.

We chose a cut off of 120 for the f-score for choosing the features that were used in building the models.

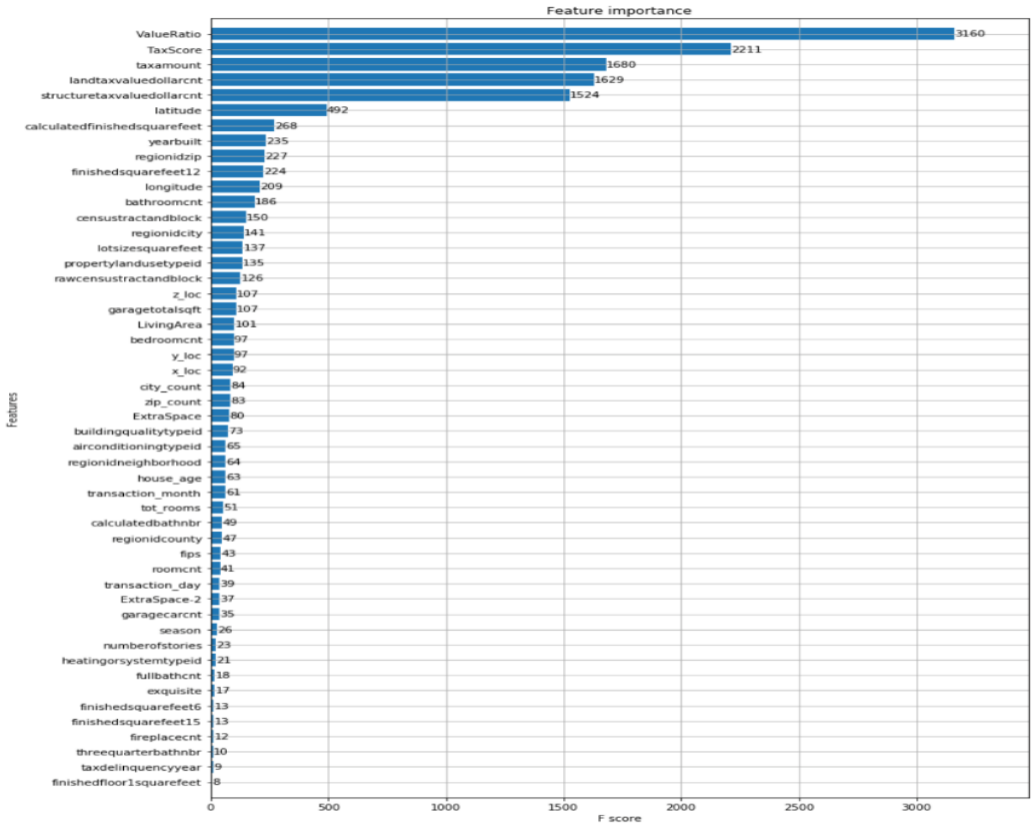

Figure 3: Feature Selection

Methodology

For building the prediction model, we used PySpark - A Python API for Spark [6]. In PySpark we create a SparkConf object to configure Spark. Using that object, we create a Spark session. The first steps were to create a Spark data frame of the cleaned data. Since the read operation is different Pandas, some columns were read differently. ValueRatio, regionidzip were some of the columns that had to be converted from string to double type.

Extraction, transforming and creation of features was done using a VectorAssembler transformer. It combines a given list of columns into a single vector column. It is useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models

features	taxvaluedollarcnt
[3.428099E7,23741...	360170.0
[3.4136312E7,5791...	119906.0
[3.37558E7,73362....	244880.0
[3.3485643E7,2649...	434551.0
[3.3870089E7,1567...	2447951.0
[3.3899475E7,4697...	111521.0
[3.4207204E7,1990...	306000.0
[3.35496E7,143230...	210064.0
[3.36127E7,80983....	190960.0
[3.4164671E7,5055...	105954.0
[3.407222E7,53108...	1090127.0
[3.4189804E7,4056...	70119.0
[3.36709E7,202253...	601000.0
[3.3913333E7,1762...	254817.0
[3.3676787E7,2551...	345023.0
[3.4176362E7,3120...	480000.0
[3.37764E7,14083....	70316.0
[3.3770148E7,1780...	253138.0
[3.4164085E7,3072...	416279.0
[3.3858594E7,1372...	260249.0

only showing top 20 rows

Figure 4: Feature Vectorization

The next step was to create a train and test split on the dataset. We decided to have an 80/20 split ratio that would allow us to test multiple models on the training data. Since we wanted to depict the best models for prediction, we leveraged Spark's parallel processing capabilities to understand which model scales better and performs faster on local as well as clusters. Hence we performed the analysis on all the models using 25,50,75 and 100 percent of data for each and every run.

We performed the regression algorithms on the local machine as well as on clusters. For the clusters, we used the Databricks Interactive UI platform [10] [12]. The Interactive clusters are used to analyze data collaboratively with interactive notebooks. The figure below shows that 166 jobs were run with a total time of 4.9 min.

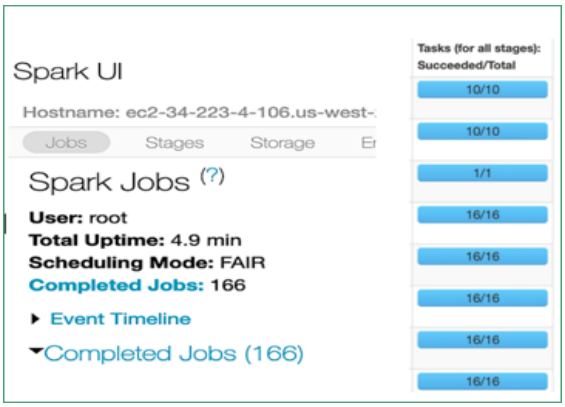

Figure 5: Spark job using DataBricks

We decided to use 3 Regression algorithms to predict the house prices - 'taxvaluedollarcnt' variable.

Linear Regression

features	taxvaluedollarcnt	prediction
[3.3340045E7,2447...	545689.0	307768.25000008056
[3.3340937E7,5048...	841483.0	633742.890399165
[3.3400699E7,4332...	151198.0	117141.75803738972
[3.3401249E7,2073...	289911.0	302774.57267815573
[3.34074E7,85157....	177418.0	133837.57197393104
[3.3408179E7,1916...	2300000.0	2531503.71356121
[3.3409415E7,4345...	170870.0	116892.18593838718
[3.3410986E7,6885...	933547.0	1005762.7720437376
[3.3412124E7,3213...	426276.0	507640.06756739784
[3.3415205E7,4944...	102039.0	130175.43674833188
[3.3416144E7,8571...	1525391.0	1307335.1066968888
[3.3416775E7,6700...	798970.0	954189.612486517
[3.3417686E7,6075...	886288.0	883344.9265224049
[3.3417795E7,9731...	110941.0	168639.03250959935
[3.3425517E7,2156...	329206.0	387067.52087844023
[3.3425517E7,2757...	393354.0	460457.4818006675
[3.3426467E7,6696...	92108.0	64345.30841725459
[3.3430979E7,2676...	354728.0	387999.2106029247
[3.3434457E7,7199...	1047177.0	1081044.6362109333
[3.3438694E7,4679...	591353.0	652834.4904413959

only showing top 20 rows

Figure 6: Linear Regression results of predicted prices

Linear Regression is useful for finding relationships between two continuous variables. We used this algorithm to understand whether our model's features are linear or nonlinear.[13] There are 4 main assumptions of regression that we had to see if our model satisfies. The results section ahead does show us that there is significant evidence to satisfy the assumptions. With the feature selection, we made sure there were no features that we correlated and could cause some bias in the model.

1. There is a linear relationship between the dependent variables and the regressors, meaning the model you are creating actually fits the data.
2. The errors or residuals of the data are normally distributed and independent from each other.
3. There is minimal multicollinearity between independent/explanatory variables.
4. Homoscedasticity. This means the variance around the regression line is the same for all values of the independent variable.

The chart below shows the computation time of the model on local and cluster.

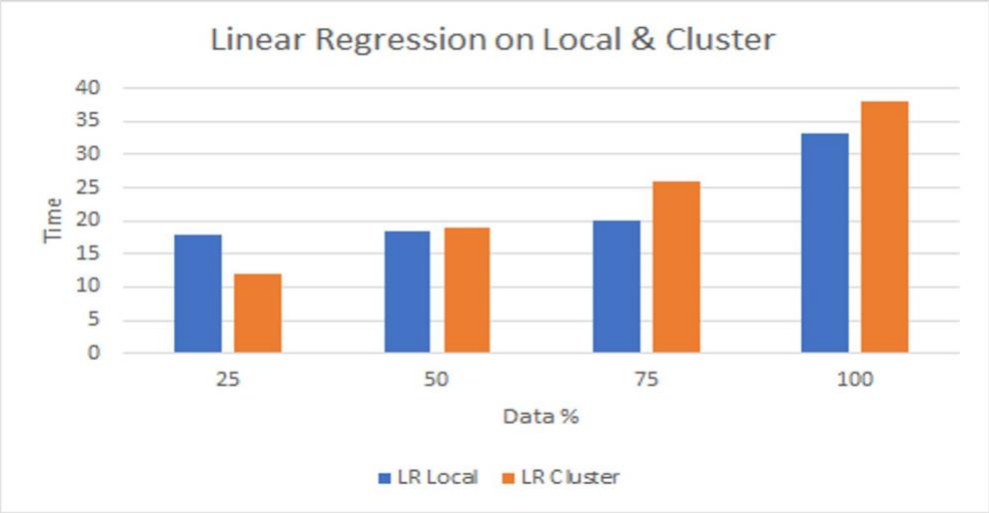

Figure 7: Computation time comparison of local machine and cluster - Linear Regression

Random Forests Regression

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging. Bagging, in the Random Forest method, involves training each decision tree on a different data sample where sampling is done with replacement.

We used Random Forest for several reasons. They are parallelizable, meaning that we can split the process to multiple machines to run. This results in faster computation time. Boosted models are sequential in contrast, and would take longer to compute.
Random forest handles outliers by essentially binning them. It is also indifferent to non-linear features.

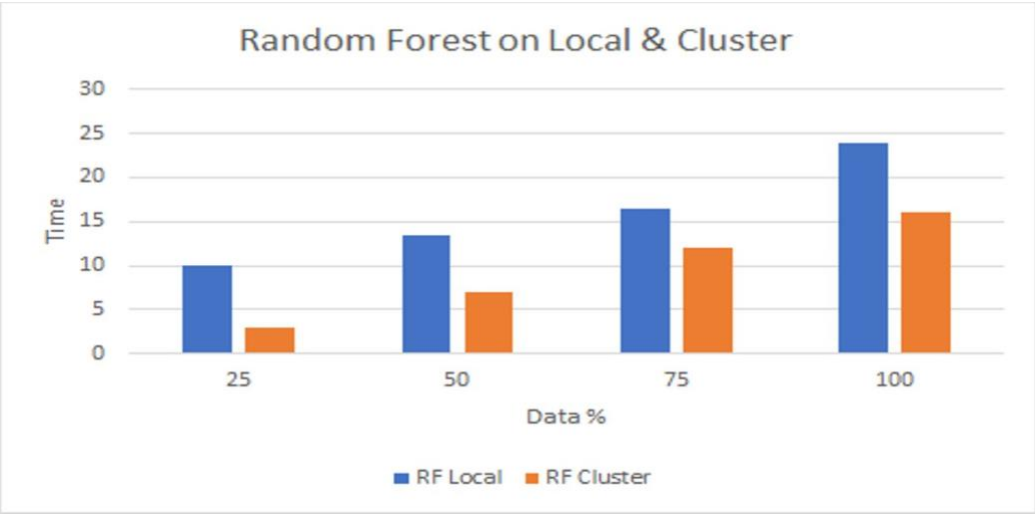

Figure 8: Computation time comparison of local machine and cluster - Random Forest

However, Random Forest is not as simple a model to interpret when it comes to Linear Regression.

Gradient Boosting Algorithm

Gradient Boosting involves three elements:-

1. A loss function to be optimized. Since this is a regression problem, our loss function will be squared error.
2. A weak learner to make predictions. These will be decision trees.
3. An additive model to add weak learners to minimize the loss function. Gradient descent is the best model to give us the results.

The model works great but takes a lot of time to compute. The GBT also has its own feature Importance ability and we used that function to compare results of the 2 GBT models. The features suggested by XGBoost [11] did not fit well and the R^2 value was very low. Hence we used the feature importance feature of GBT. The figure below shows us the results.

	Importance	Feature
1	0.328538	logerror
4	0.175176	basementsqft
3	0.088986	architecturalstyletypeid
9	0.087383	calculatedbathnbr
0	0.069305	parcelid
7	0.052497	buildingclasstypeid
10	0.050870	finishedfloor1squarefeet
5	0.050736	bathroomcnt
11	0.036420	calculatedfinishedsquarefeet
8	0.033127	buildingqualitytypeid

Figure 9: Feature Importance using XGBoost

The next figure depicts the computation time of the Gradient Boosted Trees.

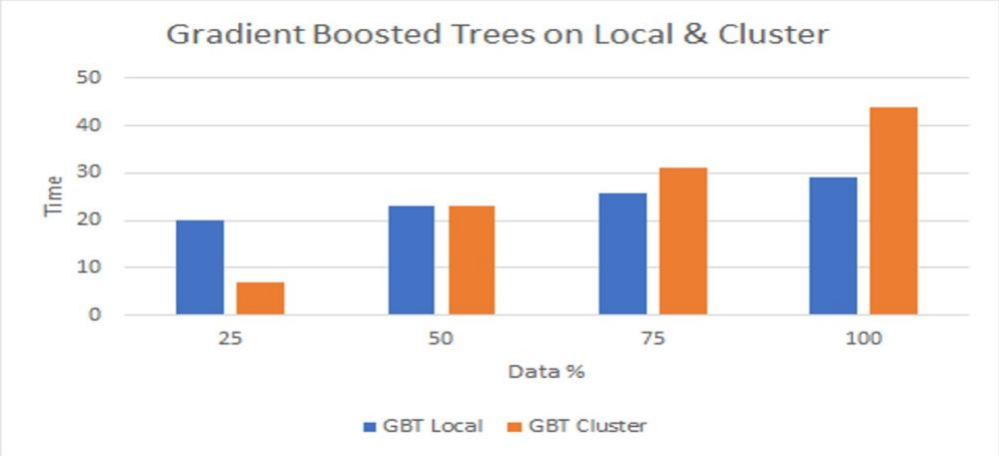

Figure 10: Computation time comparison of local machine and cluster - Gradient Boosted Tree

Results

The figure below shows the tabulated summary of all the three algorithms. As we can see from the results, the Random Forest model is the fastest in local as well as the cluster. However, the Linear Regression and GBT models run faster on the local machine.

Data %	RF (Local)	RF (Cluster)	GBT (Local)	GBT (Cluster)	LR (Local)	LR (Cluster)
25	10	3	20.15	7	18	12
50	13.4	7	22.98	23	18.5	19
75	16.51	12	25.81	31	20	26
100	23.88	16	29.2	43.8	33.15	38

Figure 11: Computation time comparison of local machine and cluster - Linear Regression

The next pictures below show us the RMSE and R squared values. Random Forests gives a very high range of RMSE. Which means that the actual and predicted price difference of the houses is very high**.**

Algorithm	R^2 scores
RF	0.777072872
LR	0.925
GBT	0.795388949

Figure 12: R^2 results of the algorithms

The R^2 scores values tell us that the Linear Regression model fits the best. Values closer to one depict most variability of the models.

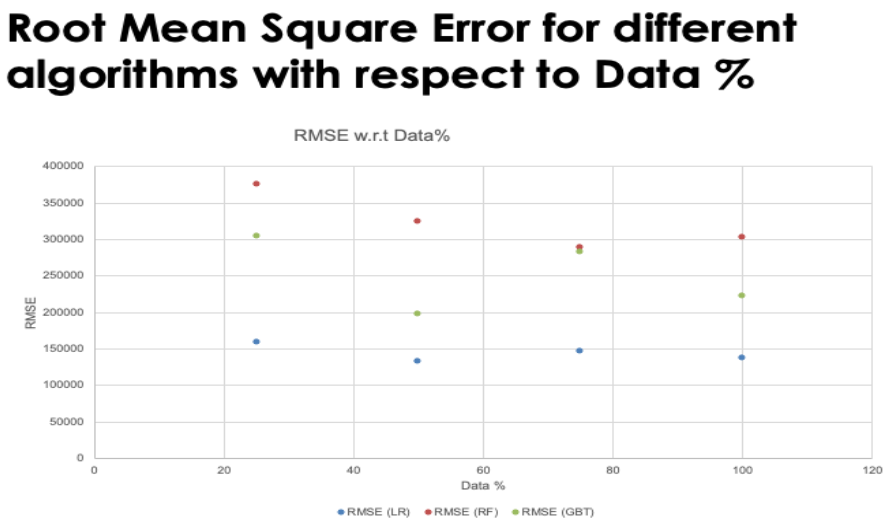

Figure 13: RMSE of the algorithms w.r.t. Data percentage

Forecasting with Seasonality

1. Introduction

Prediction of house prices as a static variable is good enough for general analysis and prediction but is not a good choice for real-world projects as the real estate market is highly volatile. Time series analysis [7] for such markets is a better option when it comes to being close to accurate predictions. We, therefore, decided to perform time-series analysis on our Zillow home value Index feature, which is a seasonally adjusted measure of the median estimated home value across a given region and housing type, using 'Prophet' [9], a procedure for forecasting time series data based on an additive model where nonlinear trends are fit with yearly, weekly and daily seasonality, plus holiday effects.

Our data was a good set for time series analysis using Prophet as the real estate market has strong seasonal effects and we had almost a decade worth of seasons of historical data.

2. Exploratory data analysis

It is important while working with time-series data to know the basic trends in the dataset. A preliminary exploratory data analysis of our dataset with respect to the "years" and "Zillow home value index (ZHVI)" gave us the result as shown in the figure:

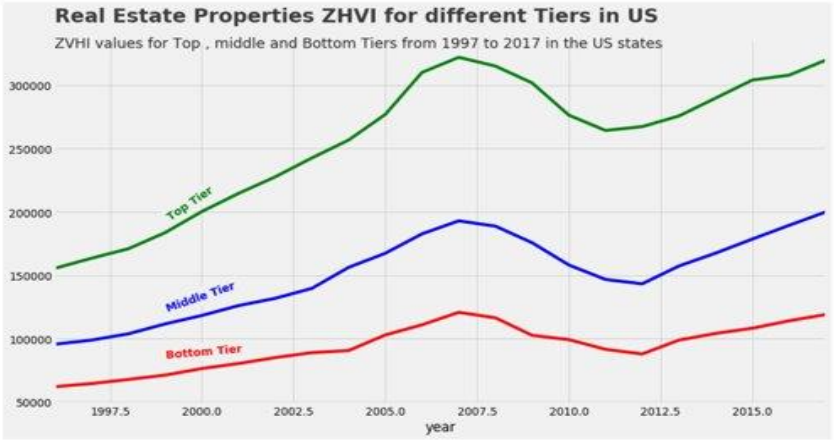

Figure 14: Real estate prices for different tiers

The homes are divided into three tiers, the bottom, middle and top. Each tier was categorized based on the value of homes, with the bottom tier < 100,000, middle tier < 200,000 and top tier > 200,000. It is apparent that the trend in the values is similar throughout all the three tiers. Hence, we can assume that our time series analysis can be applied to all tiers of home values. [14]

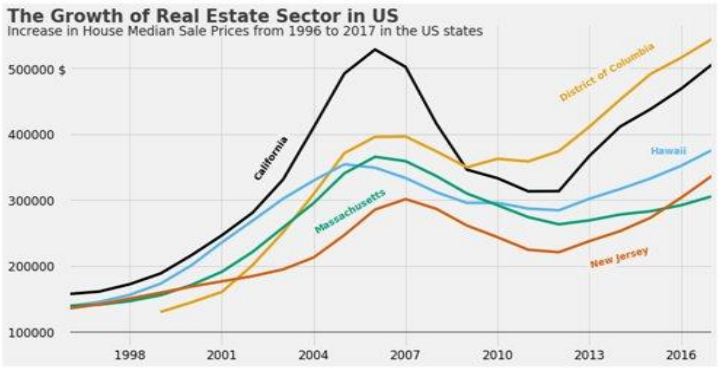

Figure 15: Growth of real estate prices for different states

Figure 15 is a confirmation of our assumption that the real estate market is highly volatile as it is evident that the prices of homes can vary hugely within a few year's time. The most volatile state observed here is California, with an increase of real estate prices by an amount of $500,000 in 2005 which drops to $300,000 by the year 2012.

3. Prophet

The input to Prophet is always a data frame with two columns "ds" and "y". The "ds" column is of the datestamp format that is expected by Pandas, and ideally has the format YYYY-MM-DD for a date or YYYY-MM-DD HH-MM-SS for a timestamp. The "y" column is a numeric measurement we wish to forecast. The two columns we have used in our model are "ZHVI" i.e. Zillow home value index, and "datestamp".

4. Prediction Results

Prediction results were made on the "ZHVI" column. By default, Prophet also includes historical dates so that our model fits well. The predict method of Prophet assigns each row in future to a predicted value called "yhat". A confidence interval value is also predicted by the method which is known "yhat_upper" and "yhat_lower" which are basically uncertainty levels. Figure 16 shows the final results of our prediction, which states that the real-estate market is only going to rise in the coming years.

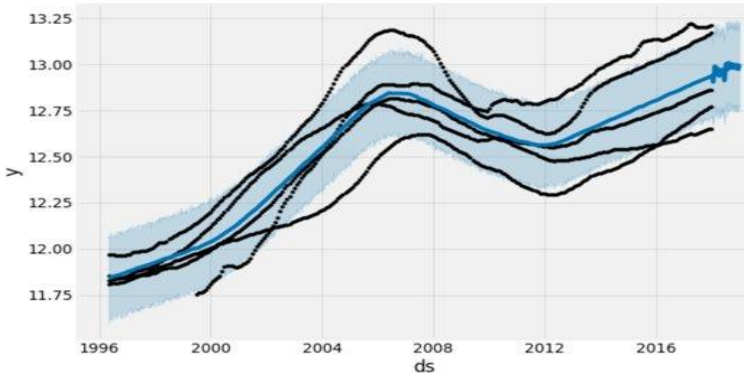

Figure 16: Price trends forecasted by Prophet

5. **Trends**

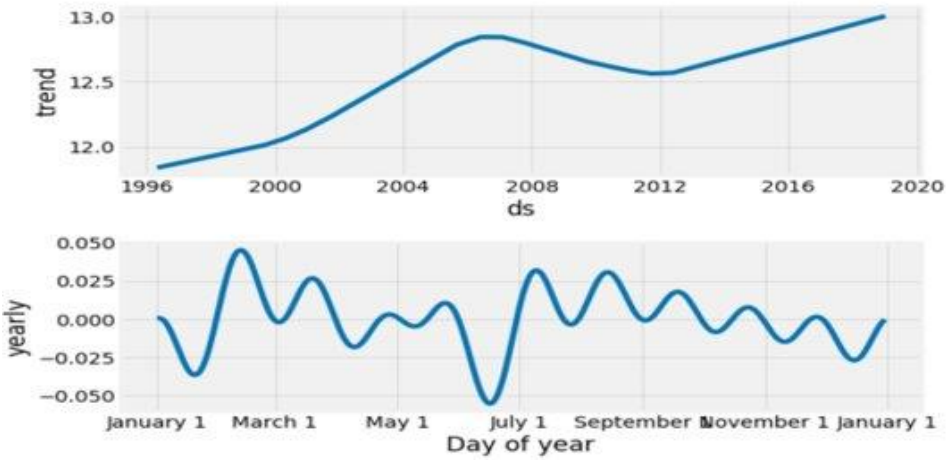

Figure 17: Forecasting components generated by Prophet

In the yearly trend, the house prices have shot right up, dipped for around 6 years and then has steadily increased since then. The daily trend shows that the summer months are generally a bad market for selling but a good time to be purchasing a house as the prices are the lowest during June-July.

Deployment

We have chosen to have an easy deployment of the model where one can easily run the Jupyter notebook of this project directly from the user's desired browser the link to the project can be found here.

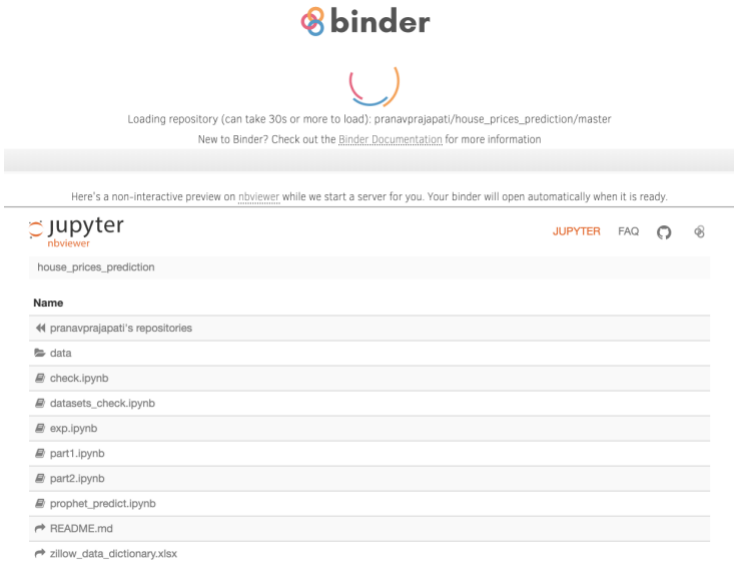

Figure 18: Deployed Project

Conclusion

Linear regression is highly interpretable, assumptions are met. Easy deployment makes it our first choice for predictive analysis. Hence, we would recommend this model. The prices are likely to go upwards and then dip in 10 years. Monthly seasonal forecasting analysis shows that during the months of summer there is a dip again in the housing prices. We learned the following things from the study:- Different models run on different scenarios and there is not always the best fit model. Simpler models like Linear Regression need to be tried out before trying out complex models as we also need high interpretability.

References

[1] Statista, Number of households in the U.S. 1960-2018, Available: https://www.statista.com/statistics/183635/number-of-households-in-the-us/

[2] Zillow, Zillow Prize: Zillow's Home Value Prediction (Zestimate), Available: https://www.kaggle.com/c/zillow-prize-1

[3] Wikipedia, Data cleansing, Available: https://en.wikipedia.org/wiki/Data_cleansing

[4] Microsoft. Modeling Flags (Data Mining), Available: https://docs.microsoft.com/en-us/sql/analysis-services/data-mining/modeling-flags-data-mining?view=sql-server-2017

[5] Amjad Abu-Rmileh, The Multiple faces of 'Feature importance' in XGBoost, Available: https://towardsdatascience.com/be-careful-when-interpreting-your-features-importance-in-xgboost-6e16132588e7

[6] https://spark.apache.org/docs/latest/api/python/index.html

[7] Franses, Philip Hans. "Seasonality, non-stationarity and the forecasting of monthly time series." *International Journal of forecasting* 7.2 (1991): 199-208.

[8] Machine Learning Mastery- "Data Preparation for Gradient Boosting with XGBoost in Python", Available: https://machinelearningmastery.com/data-preparation-gradient-boosting-xgboost-python/

[9] https://facebook.github.io/prophet/

[10] Azure Databricks Hands-on, Available: https://medium.com/@jcbaey/azure-databricks-hands-on-6ed8bed125c7

[11] Using XGBoost in Python, Available: https://www.datacamp.com/community/tutorials/xgboost-in-python

[12] Apache Spark Tutorial- Getting Started with Apache Spark on Databricks, Available: https://databricks.com/spark/getting-started-with-apache-spark

[13] Regression - Forecasting and Predicting, Available: https://pythonprogramming.net/forecasting-predicting-machine-learning-tutorial/

[14] Udacity- Time Series Forecasting, Available: https://www.udacity.com/course/time-series-forecasting--ud980

[15] Hujia Yu, Jiafu Wu, "Real Estate Price Prediction with Regression and Classification". Available: http://cs229.stanford.edu/proj2016/report/WuYu_HousingPrice_report.pdf

[16] Neelam Shinde, Kiran Gawande, " Valuation Of House Prices using Predictive Techniques". Available: http://www.iraj.in/journal/journal_file/journal_pdf/12-477-153396274234-40.pdf

[17] Eric Ghysels, Alberto Plazzi, Walter Torous, Rossen Valkanov, "Forecasting Real Estate Prices". Available: https://rady.ucsd.edu/faculty/directory/valkanov/pub/docs/HandRE_GPTV.pdf