# A study into the performance of different ML techniques applied to three datasets – Credit Card default, YouTube trending videos and House Sales in King County

*National College of Ireland*

*MSc in Data Analytics January 2021*

Sanket Sonu

x19206071@student.ncirl.ie

Prof. Pierpaolo Dondio

**GitHub:**
https://github.com/SanketSonu/Data-Mining-and-Machine-Learning-Projects

***Abstract:*** **This research paper consists of three different datasets Credit Card default, YouTube trending videos and House Sales in King County, USA. We are using different machine learning models to predict the default, likes and price respectively for our datasets. We have used both classification and regression datasets. Credit Card is a classification problem and YouTube, and House Sales are regression problem. This paper also focuses on improving accuracy and results after each machine learning models. Hyperparameters Tuning has been used in this project after using simple machine learning models. This research paper will show, how the performance of a machine learning model can be boosted using specific parameters.**

**Keywords: YouTube trending videos, Credit Card default, House sales, KNN, Logistic Regression, Decision Tree, Random Forest, SVM, Bagging, Boosting, AdaBoost, Gradient Boost, XG Boost, Multiple Linear Regression, StatsModel OLS and Hyperparameter tuning.**

## I. INTRODUCTION

In recent years, the demand for machine learning is increasing. There are needs to explore the capabilities of machine learning.

This Data Mining and Machine Learning – 1 project aims to show how Machine Learning models work with different datasets and from which model we are getting more Accuracy. 3 datasets are used in this project, out of which 2 are regression problems – YouTube and House Sales, where likes and sales prices are to be predicted using various machine learning models and for classification data, we will use Credit Card default detection to predict customers who do not repay after using Credit Card.

### *A. Credit Card Default:*

Nowadays, banks are showing interest in customer activity by tracing their Credit Card records. Bank use this for various reasons, to give extra benefits, discount offers etc. to the customer. This is all about creating a positive bond with good customers so that they can subscribe to a bank and can use most of it. However, there are few customers, who use Credit Card like a normal person but after months of using them, they deny repaying the Credit Card bill because of which the bank is facing a huge loss. This dataset contains information on default payments (Payment of the previous bill), demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005. This dataset is sourced from Kaggle: Default of Credit Card Clients Dataset | Kaggle. Our project topic is from bank marketing data. Now, the question is:

- Can we predict if a customer can pay the bill or not?

### *B. House Sales in King County, USA:*

Real Estate is one of the profitable business in this changing world. Everyone dreams to buy a big and beautiful house. Class of house is a new type of luxury status in today's world. People nowadays, invest from thousands to millions of USD $ to purchase a beautiful house.

However, 'price' varies from property to property. Price mostly depends on many variables like land size, property age, no of rooms, swimming pool etc. This dataset is sourced from Kaggle: House Sales in King County, USA | Kaggle We are working on this question for this dataset:

- Can we predict 'price' of houses based on given features?

### *C. YouTube Trending Videos:*

YouTube is the most popular video-sharing platform. YouTube is a full-time career for many people. They work, days and nights to create a good video with trending contents. Many YouTubers are earning millions from their YouTube videos.

YouTube earn money from advertisements, sponsors, sales, and a donation from fans. YouTube amount to Video Creators based on likes, views, and many factors, to get more views and likes, people create videos based on a trending topic, to gain popularity. This dataset is sourced from Kaggle: Trending YouTube Video Statistics | Kaggle. Now, the question is:

- Can we predict the number of likes on the trending videos?

## II. RELATED WORK

### *A. Credit Card Default:*

'FRAUD' in any sector is unauthorized and Credit Card Fraud are also unauthorized. Here, FRAUD means Customer fails to pay their bill and use other persons credit card without knowing them.

[1] The study "Credit Card Fraud Detection using Machine Learning and Data Science" aims to calculate the probability of default.payment.next.month. To solve this problem author used various Machine Learning approaches such as Artificial Neural Networks, Fuzzy Logic, Genetic Algorithm, Logistic Regression, Decision Tree, Support Vector Machine, Bayesian Networks, Hidden Markov Model, K-Nearest Neighbour. Among the various Machine Learning approaches, the author was able to get 99.6% accuracy, however, the precision remains at 33%. This high percentage of accuracy is only because of imbalanced data between the number of valid and genuine transactions.

[2] The study "Detecting Credit Card Fraud by Decision Trees and Support Vector Machines" aims to compare the performance of decision tree and support vector machine for detecting frauds from Credit Card dataset. According to the author, most Credit Card fraud detection systems are using SVM and Neural Networks. In this study, the performance classifier models are built by using Decision Tree, C&RT and CHAID and various SVM methods with different Kernel functions such as Polynomial, Sigmoid, RBF and Linear are used.

[3] The study "Artificial Neural Network Technique for Improving Prediction of Credit Card Default: A Stacked Sparse Autoencoder Approach" aims to detect default payment of Credit Card records. The author said that because of the imbalanced nature of Credit Card data, it is very challenging to predict exact fraud with classification models. The author has used an autoencoder which consists of two functions, that is, an encoder and a decoder. The author has also used the AdaMax algorithm which uses infinite norm, which was used to optimize autoencoder.

The author used SMOTE to tackle the problem while using this imbalanced dataset. The author used SMOTE with few algorithms and got the best result for Random Forest that is 89% accuracy and 89% F1 Score.

### *B. House Sales in King County, USA:*

Real Estate needs too much attention, as the price of land values are going up day by day and depends on many features.

[8] The study "House Price Prediction" demonstrates the prediction of the price using various machine learning models. Authors have used Linear Regression, Lasso Regression, Ridge Regression, Random Forest and Artificial Neural Network.

[9] The study “Housing Prices Prediction with a Deep Learning and Random Forest Ensemble” aims to predict sales. Authors have used Random Forest Ensemble and Bidirectional LSTM – Recurrent Neural networks.

[10] The study “House Price Forecasting using Data Mining” aims in predicting Price using the Linear Regression algorithm. The author has used the Mumbai dataset to observe the customers interest by predicting price based on few variables.

### *C. YouTube Trending Videos:*

Nowadays, YouTubers earn a lot of money from their videos but for that, they need to get more likes and views, there is a competition to be in the latest trending video section, to get more views and likes.

[15] The study “YouTube Videos Prediction: Will this video be popular?” aims to predict the performance of a trending YouTube video. The author has used Gradient Boost and backward search. The author also used Random Forest with tuning parameters and had f1 score of 0.736.

[16] The study “A regression approach for prediction of YouTube views” aims to predict views of a video. The author has used the OLS method compared to the Gradient descent method. However, the author was able to get prediction only above average.

[17] “Predicting the popularity of online videos using Support Vector Regression” aims to predict the popularity of a video. The author has used Support Vector Regression with Gaussian functions. The author predicted the popularity of video by 93 %.

## III. METHODOLOGY

### *A. Credit Card Default:*

**Dataset**:

The Credit Card dataset has 25 variables and 30,000 observations. The target variable in this dataset is ‘default.payment.next.month’. This is the list of variables:

’ID’, 'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6', ‘default.payment.next.month’.

**Data Cleaning:**

Firstly, Null values need to be checked, however, after checking this dataset, there are no null values. Then, column ID has been removed from the data because it contains just a normal serial number of observations.

**Correlation:**

(Fig. 1) We can demonstrate that no variables are strongly correlated with the Target variable (default). The ‘PAY_’ variables have a strong correlation between them and have a weak positive correlation with the target variable(default). All the ‘BILL_AMT’ variables have a good positive correlation between them. Also, ‘LIMIT_BAL’ has a good positive correlation with ‘BILL_AMT’ variables.

**Pre-processing:**

We have split this data into 75% and 25% for train and test sets respectively using sklearn.model_selection train_test_split. We created x_train, x_test, y_train and y_test.

Also, we have transformed the data using sklearn. preprocessing MinMaxScaler. Here, we have used x_train and x_test to transform them into xtrain_scaler and xtest_scalar because there are many observations with large ranges such as ‘TOTAL_PAY’, ‘LIMIT_BALANCE’ and ‘TOTAL_BILL’. We are using MinMaxScaler to scale our variables and convert them in the range 0-1. Since all variables are transformed using the same MinMaxScaler in the same range 0-1, so the degree to which it affects our target variable will become equal and will avoid variables from being biased because of large range values.

**Models, Predictions, and Analysis:**

We have used 5 machine learning models for this dataset.

**1. KNN (K-nearest neighbors):**

First, we checked for which K, our model has the best accuracy. We performed KNN two times, once with normal train and test sets and a second time with transformed (MinMaxScaler) train and test sets.

- During our first model, we found results as:
  - K: 28
  - Accuracy: 78.56 %
- During our second model of KNN with transformed sets:
  - K: 29
  - Accuracy: 81.78 %

After plotting confusion matrix (Fig. 2) with transformed sets, we were able to get:

- Accuracy: 81.78 %
- Precision: 63.40 %
- Recall: 33.06 %

(Fig. 3) Shows ROC Curve and it can be observed that "Area Under the Curve" is 0.75 We can observe a decent ROC curve.

**2. Logistic Regression:**

Next, Logistic Regression was used for this dataset. This model is tuned by giving 'C' [0.001,0.01,0.1,0.5,1.0] and assigning the solver as 'liblinear'. This model is hyperparameter tuned using sklearn's GridSearchCV. We performed this model two times, once with normal train and test sets and a second time with transformed (MinMaxScaler) train and test sets.

- During our first model, we found results as:
  - C: 0.001
  - Accuracy: 78.81 %
- During our second model with transformed sets:
  - C: 1.0
  - Accuracy: 82.68 %

After plotting confusion matrix (Fig. 4) with transformed sets, we were able to get:

- Accuracy: 82.68 %
- Precision: 68.74 %
- Recall: 33.37 %

(Fig. 5) Shows ROC Curve and it can be observed that "Area Under the Curve" is 0.73 ROC Curve can be used to compare with other models, it shows the area under the curve. Large values on Y-Axis demonstrates lower false negatives and higher true positives.

**3. Decision Tree:**

Next, Decision Tree is used for this dataset. We have used 3 variants of models: 1st Full tree with normal test and train sets, 2nd Pruned tree with normal test and train sets, and 3rd Pruned tree with transformed (MinMaxScaler) train and test sets.

- Decision Tree (Unpruned) with normal test and train sets:
  - Accuracy: 73.34 %
- Decision Tree – which was pruned using max_depth for 1 to 20 range and used normal train and test sets here.
  - Accuracy: 83.04 %
  - max_depth: 4
- Decision Tree – which was pruned using max_depth for 1 to 20 range and this time using transformed (MinMaxScaler) train and test sets.
  - Accuracy: 83.06 %
  - max_depth: 3

After plotting confusion matrix (Fig. 6) with transformed sets, we were able to get:

- Accuracy: 57.21 %
- Precision: 24.61 %
- Recall: 47.10 %

(Fig. 7) Shows ROC Curve and it can be observed that "Area Under the Curve" is 0.55 ROC Curve can be used to compare with other models, it shows the area under the curve. We

got not good results here, it is clearly visible in ROC that area under the curve is less.

**4. Random Forest:**

Next, we used Random Forest. We used two models of Random Forest for the dataset. Both times, we used transformed (MinMaxScaler) train and test sets. 1st model is Simple Random Forest and 2nd model is hyperparameter tuned Random Forest model.

- During our first simple model, we found results as:
  - Accuracy: 79.44 %
- For second model, we have used hyperparameter tuning, for this we changed max_depth to 3 (because we had got depth 3 for decision tree which showed best accuracy), random state = 5 and n_estimators values as [10,50,80,100,150,200,250,300]. This model is hyperparameter tuned using sklearn's GridSearchCV.
  - Accuracy: 81.80 %
  - n_estimator: 10

After plotting confusion matrix (Fig. 8) with transformed sets, we were able to get:

- Accuracy: 81.80 %
- Precision: 69.05 %
- Recall: 25.44 %

(Fig. 9) Shows ROC Curve and it can be observed that "Area Under the Curve" is 0.77 ROC Curve can be used to compare with other models, it shows the area under the curve. Large values on Y-Axis demonstrates lower false negatives and higher true positives.

**5. SVC (Support Vector Classifier):**

Next, we used Support Vector Machine for this dataset. This model is tuned using 4 kernel values ['rbf', 'linear', 'poly' and 'sigmoid']. We used this model with transformed (MinMaxScaler) train and test sets.

- Accuracy: 81.64 %
- Kernel: poly

After plotting confusion matrix (Fig. 10) with transformed sets, we were able to get:

- Accuracy: 81.64 %
- Precision: 69.71 %
- Recall: 23.48 %

(Fig. 11) Shows ROC Curve and it can be observed that the "Area Under the Curve" is 0.69 ROC Curve which we can use to compare with other models, it shows the area under the curve. From the ROC curve, we can demonstrate that area under the curve is less.

**6. Bagging with all classifiers using Cross Validation:**

Next, we used the bagging method, and this will create all models using different data and a weighted average will be used to determine the result. We have used all 5 ML models (KNN, Logistic, Decision Tree, Random Forest and SVM). We hyperparameter BaggingClassifier with all 5 ML models, max_samples=0.25, max_features=10, random_state=3. For cross val score, parameters are passed like cv = 10 and n_jobs = -1. We can see from the below (Fig. 12) that for KNN, Logistic and Decision Tree, accuracy is increasing, and the standard deviation is decreasing for KNN, Decision Tree and Random Forest.

To choose the best classifier, we will use Sklearn's VotingClassifier, which will help us to combine different ML classifiers and will perform a vote on all classifiers.

(Fig. 13) Shows results we can observe that RandomForest had the best accuracy 81.30 % with a very low standard deviation of 0.01.

**7. Boosting with all classifiers using Cross Validation:**

Lastly, we used boosting technique. This boosting is not random, and the current performance of the model will depend on previous models. We used Ada Boost Classifier, Gradient Boosting Classifier and XG Boost Classifier. We have used all 5 ML models (KNN, Logistic, Decision Tree,

Random Forest and SVM). We hyperparametered cross val score with all 5 ML models, cv = 10 and scoring = 'accuracy'. We also tuned 'EnsembleVoteClassifier' with voting = 'hard' and for all 3 boosters Ada boost, Gradient boost and XG Boost.

To choose the best classifier, we will use Sklearn's VotingClassifier, which will help us to combine different ML classifiers and will perform a vote on all classifiers. (Fig. 14) According to the results, Gradient boost came out to be best with 82.10 % accuracy and 0.011 standard deviation.

### *B. House Sales in King County, USA:*

**Dataset:**

This dataset has 21 variables and consist of 21,613 observations. The target variable is 'price'. List of variables are : 'id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade', 'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode', 'lat', 'long', 'sqft_living15', 'sqft_lot15'

**Data Cleaning:**

First, Null Values need to be checked as it is very important to remove Null values for Multiple Regression. However, we did not find any Null values in this dataset. Then 2 columns: 'id' and 'date' are removed from DataFrame as these contain useless information.

**Correlation:**

(Fig. 15) We can demonstrate that all variables are in good correlation with 'price'. Only 'zipcode' has a negative correlation of -0.05 but are very near to 0 with the target variable. 'sqft_living', 'grades' and 'bathrooms' are having a positive strong correlation with the target variable 'price'.

**Pre-processing:**

We used train_test_split from sklearn library to split our data into 75% and 25% for train and test sets respectively. We created x_train, x_test, y_train and y_test. The Random state for train and test is 3.

**Models, Predictions, and Analysis:**

First, we have visualized a few of the crucial information. (Fig. 16) Shows the bedrooms count, and it can be observed that most of the properties are having 3 bedrooms and 4 bedrooms. (Fig. 17) Shows the bathroom count, and it can be observed that most of the houses are having 2.5, 1, and 1.75 bathrooms. (Fig. 18) Shows property with waterfront and we can observe that the maximum of the houses is not having a waterfront and only a few have a waterfront feature. (Fig. 19) Shows how many floors maximum properties have, and we can observe that most of the properties are having 1 and 2 floors.

We have used 4 regression models for this dataset:

**1. Multiple Linear Regression:**

We have used Multiple Linear Regression for this dataset. This model provided an average result. Below are the results:

- RMSE: 444.30
- R2 Score: 0.71
- Accuracy: 70.78 %

(Fig. 20) Shows the lmplot for this multiple linear regression model and it plots a straight line, but this is not much close to 45 degrees.

**2. Decision Tree:**

Next, we used Decision Tree for our model. For this, we used 2 variants of model unpruned simple decision tree model and tuned regressor with multiple max_depth. Results are:

- Decision Tree (Unpruned):
  - RMSE: 422.72
  - R2 Score: 0.76
  - Accuracy: 76.05 %
- Decision Tree (Pruned): which was pruned using max_depth for 1 to 20 range. This model is hyperparameter tuned using sklearn's GridSearchCV.
  - Max_depth: 11
  - RMSE: 406.80
  - R2 Score: 0.79
  - Accuracy: 79.46 %

(Fig. 21) Shows the lmplot which is a straight line and closer to 45 degrees. This plot turns out to be much better than the Multiple Linear Regression model.

**3. Random Forest:**

We have used Random Forest for this dataset. We have used 2 variants of Random Forest; 1st is normal Random Forest and 2nd is Hyperparameter tuned, Random Forest. We are using GridSearchCV from sklearn. For the 2nd model, we have used parameters like 'n_estimators' and 'max_depth'. We will iterate through all parameters and find the best one. Results are:

- Random Forest (Simple):
  - RMSE: 351.26
  - R2 Score: 0.89
  - Accuracy: 88.58 %
- Random Forest (Tuned): n_estimators = [140,160,180,200,220] and max_depth = [10,15,20,25,30]
  - Best n_estimators: 180
  - Best max_depth: 30
  - RMSE: 351.30
  - R2 Score: 0.89
  - Accuracy: 88.58 %

(Fig. 22) Shows the lmplot and it can be observed that this time we got a straight line which is close to 45 degrees. Random Forest with tuned parameters looks very efficient for this dataset.

**4. StatsModel OLS:**

StatsModel is the last model we are using to get the best 'price' prediction. First, we are using a basic model and from (Fig. 23) we can observe the P values of all Independent Variables. It is observed that only the floor is having P > 0.05, i.e, 0.063. So, for the next model we will remove the 'floor' variable and run this model again to get very good results.

- StatsModel OLS:
  - Accuracy = 70 %
- StatsModel OLS after removing 'floors' (P>0.05):
  - Accuracy = 90.50 %

(Fig. 24) Clearly shows that after removing the 'floor' variable we are getting 90.50 % accuracy which is the highest among all other models. Also, the F-Statistics value is very small and close to 0.

***C. YouTube Trending Videos:***

**Dataset:**

This dataset has 16 variables and 40,949 rows. We are using USVideos dataset. The target variable is "likes". List of variables:

'video_id', 'trending_date', 'title', 'channel_title', 'category_id', 'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comment_count', 'thumbnail_link', 'comments_disabled', 'ratings_disabled', 'video_error_or_removed', 'description'.

**Data Cleaning:**

First, Null values need to be checked as it is very important to predict likes. We only have 570 Null values for the Description variable. There are so many irrelevant variables such as dates, links, ID, and description. We are removing all variables because we need numerical values for regression models. Dropped 'video_id', 'trending_date', 'title', 'channel_title', 'category_id', 'publish_time', 'tags', 'thumbnail_link', 'comments_disabled', 'ratings_disabled', 'video_error_or_removed', 'description'. We will only work with most important variables: 'likes', 'views', 'dislikes' and 'comment_count'.

**Correlation:**

(Fig. 25) We can demonstrate that all variables are in good correlation with the target variable. Views have 0.85 correlation with likes, dislikes have 0.45 correlation with likes and comment count has 0.80 correlation with target variable likes.

We can conclude that none of the independent variables is highly correlated among themselves.

**Pre-processing:**

We have split this data into 75% and 25% for train and test sets respectively using

sklearn.model_selection train_test_split. We created x_train, x_test, y_train and y_test. The Random state for train and test is 3.

**Models, Predictions, and Analysis:**

We have used 3 regression models for this dataset:

**1. Multiple Linear Regression:**

We have used Multiple Linear Regression for the YouTube USVideos dataset. The linear Regression model provided a good result. The R2 score value is close to 1.0 and accuracy is also above average.

- RMSE: 272.56
- R2 Score: 0.89
- Accuracy: 89.14 %

(Fig. 26) Shows the lmplot, which fits the regression model with conditional parameters, and it can be observed that a straight line can be drawn.

**2. XG Boost Regressor:**

We used 2 XGB Regressor models for this dataset. The First will be a very simple XGB Regressor model and the second will be tuned XGB Regressor by changing parameters.Simple XGB Regressor:

- RMSE: 229.42
- R2 Score: 0.95
- Accuracy: 94.55 %

- Tuned XGB Regressor: (n_estimators=5000, learning_rate = 0.001)
  - RMSE: 227.84
  - R2 Score: 0.95
  - Accuracy: 94.70 %

XGB Regressor provided a very good result when compared to linear regression. The hyperparameter tuned XGB Regressor model gave 0.15 % more accuracy than the Simple XGB Regressor. Also, it can be observed that for both the models R2 value is very close to 1.0 which is very good for a model. Also, the RMSE value for Tuned XGB Regressor is low than the Simple XGB Regressor.

(Fig. 27) Shows the lmplot for the XGB model, it fits the regressor model perfectly and provides a very good positive result close to a 45-degree line.

**3. Random Forest:**

We have used Random Forest Regressor for this dataset. Two models of Random Forest are introduced here, first is simple random forest regressor and second is hyperparameter tuned random forest with n_estimators = [140,160,180,200,220] and max_depth = [10,15,20,25,30].

- Simple Random Forest:
  - RMSE: 221.89
  - R2 Score: 0.95
  - Accuracy: 95.23 %
- Hyperparameter tuned model:
  - Best max_depth: 30
  - Best n_estimators: 140
  - RMSE: 221.76
  - R2 Score: 0.95
  - Accuracy: 95.24 %

Random Forest Regressor provided a very good result and best among all 3 ML models we used.

It has the lowest RMSE value and R2 is very close to 1.0 with very high accuracy.

(Fig. 28) Shows the lmplot for the XGB model, it fits the regressor model perfectly and provides a very good positive result close to a 45-degree line.

## IV. CONCLUSION & FUTURE WORK

This research paper is based on 3 datasets: Credit Card Fraud Detection, YouTube trending videos and House Sales in King County, USA.

The first dataset is the Credit Card dataset, where we had records from Taiwan from April 2005 to September 2005. This dataset was highly imbalanced and needed resampling or

transformation. We used MinMaxScaler to transform our train and test data. After applying 5 machine learning models with normal data and transformed data, it can be easily observed that the performance of all models was good with transformed train and test data. KNN, Logistic Regression and RandomForest gave the best accuracy around 82 %. Logistic Regression was best with 82.68 % accuracy, recall 84.27 % and 95.92 % recall. After applying bagging, it was observed that RandomForest came out to be the best with 81.30 % accuracy and a very low 0.01 standard deviation. In last, we used boosting for all 5 machine learning models and applied Ada boost, Gradient boost and XG boost. Out of these 3. Grad boost performed best when both accuracy and standard deviation were compared with other boosting methods. For Grad boost, we got 82.10 % accuracy and 0.011 standard deviation.

We have not introduced other ML models, which can give better results. In future, we are looking to apply other ML models with SMOTE to oversample our imbalanced data. We can also apply Neural networks and deep learning, as this can drastically improve the performance with some unique tunning parameters.

The second dataset is House Sales in King County, USA, where we predicted 'price'. This dataset had few variables which were removed during data cleaning and the correlation of all variables were good with target variables. We have used 4 machine learning models for this dataset, Multiple Linear Regression produced an average result and accuracy of 70.78 %, however, hyperparameter tuned Decision Tree also provided accuracy of around 79.46 %. Random Forest worked well and for both simple and hyperparameter tuned Random Forest Model, accuracy came out to be 88.58 %. However, after using StatsModel OLS, we found that the 'floors' variable has P values > 0.05, so we removed that variable and received a very good model with 90.50 % accuracy. StatsModel after removing the 'floors' variable turns out to be the best model for our dataset.

We have not introduced Neural Networks and Deep Learning for this dataset. In future, we can try those with SMOTE or MinMaxScaler in search of a good result.

The third dataset is YouTube trending video for USVideos, where we predicted 'likes'. This dataset had few variables that were useless to use like ID's, links, description etc. So, we removed those variables and continued our Regression models with important variables. We tested 3 machine learning models for this dataset, 1st was Linear regression which provided good accuracy of 89.14%, RMSE was 272.56 and R2 was very close to 1.0, i.e., 0.89. 2nd we used XGB Regressor, for this we used 2 models, 1st was a simple model and we got good results and 2nd model was tuned XGB Regressor with n_estimators=500 and learning_rate = 0.001, XGB Tuned Regressor model gave slightly better results, i.e., 94.70 accuracies, R2 0.95 and RMSE 227.84. 3rd we used Random Forest Regressor, in this we used 2 models, which was Simple Random Forest and Hyperparameter tuned model, we received very good results for these models, like RMSE 223.13, R2 0.95 and Accuracy 95.12 %, however after using hyperparameter model, we got slightly better and best results among all the models that we used, we got RMSE: 221.66, R2: 0.95 and Accuracy 95.25 % when max_depth was 30 and n_estimators was 180. Conclusively, Hyperparameter Tuned Random Forest was best in predicting target variable 'likes'.

We can use other ML models to get better performance in future. We can use Natural Language Processing for few variables like Description or Title, to get some pattern and check how Keywords also play important role in getting more views and likes, maybe Keywords or title or description is very similar to current trending topic?

## REFERENCES

[1] "Credit Card Fraud Detection using Machine Learning and Data Science" by Maniraj S P, Aditya Saini, Shadab Ahmed and Swarna Deep Sarkar. IJERT. ISSN: 2278-0181. Vol. 8 Issue 09, September-2019

[2] “Detecting credit card fraud by decision trees and support vector machines” by Y. G. Sahin and E. Duman, International MultiConference of Engineers and Computer Scientists, vol. 1, 2011, pp. 442-447.

[3] “Artificial Neural Network Technique for Improving Prediction of Credit Card Default: A Stacked Sparse Autoencoder Approach” by Sarah A. Ebiaredoh-Mienye, E. Esenogho, Theo G. Swart. International Journal of Electrical and Computer Engineering. Vol. 9, No. 4, Aug 2020.

[4] “The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients” by I-Cheng Yeh a, Che-hui Lien b

[5] “Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition”, IEEE Trans. Electronic Computers, vol. 14, pp. 326-334, June 1965.

[6] “Neural Networks for Maximum Likelihood Clustering”, Signal Processing, vol. 36, no. 1, pp. 111-126, 1994.

[7] “Credit Card Fraud Detection with Machine Learning Methods” by Gokhan Goy, Cengiz Gezer, Vehbi Cagri Gungor. ISBN:978-1-7281-3965-4

[8] “House Price Prediction” by Ahmad Abdulal, Nawar Aghi. https://www.diva-portal.org/smash/get/diva2:1456610/FULLTEXT01.pdf

[9] “Housing Prices Prediction with a Deep Learning and Random Forest Ensemble” by Bruno, Luckeciano, Williams and Samuel. IST.

[10] “House Price Forecasting using Data Mining” by Nihar, Ankit and Shreyash. International Journal of Computer Applications (0975 – 8887) Volume 152 – No.2, October 2016.

[11] “Real estate value prediction using multivariate regression models” by R Manjula, Shubham Jain, Sharad Srivastava and Pranav Rajiv Kher. IOP Conference Series: Materials Science and Engineering, Volume 263, Issue 4.

[12] “House Price Forecasting using Data Mining Techniques” by Atharva chogle, Priyanka Khaire, Akshata gaud3, Jinal Jain4. ISO 3297:2007 Certified Vol. 6, Issue 12, December 2017.

[13] “Developing a Forecasting Model for Real Estate Auction Prices Using Artificial Intelligence” by Jun Kang , Hyun Jun Lee , Seung Hwan Jeong , Hee Soo Lee , and Kyong Joo Oh.

[14] “Forecasting Techniques for Sales Prediction” by Pachipala Yellamma, B Abhinav, B Jaya Vaishnavi, G Ushaswini, Malladi Srinivas. Vol. 29 No. 06 (2020): Vol. 29 No. 06 (2020).

[15] “YouTube Videos Prediction: Will this video be popular?” by Yuping Li1, Kent Eng1, Liqian Zhang. Stanford University, Stanford, CA 94305

[16] “A regression approach for prediction of YouTube views” by Lau Tian Rui, Zehan

Afizah Afif, Rd. Rohmat Saedudin and Aida Mustapha. Vol. 8, No. 4, December 2019, pp. 1502~1506

[17] “Predicting popularity of online videos using Support Vector Regression” by Tomasz Trzcinski and Przemysław Rokita. arXiv:1510.06223v4 [cs.SI]

[18] “Predicting the Popularity of Trending Videos in Youtube Using Sentimental Analysis” by G. Mohana Prabha, B. Madhumitha, R. P. Ramya. ISSN: 2278-3075, Volume-8, Issue-6S3, April 2019

[19] “Popularity Prediction of Videos in YouTube as Case Study: A Regression Analysis Study”. Conference: the 2nd international Conference.

[20] “YouTube Video Popularity: Predicting Video View Count from User-Controlled Features”. Jordy Snijders ANR 386791

[21] “A Peek into the Future: Predicting the Popularity of Online Videos” 10.1109/ACCESS.2016.2580911.

## APPENDIX:

Appendix will have all the plots and result related to all the datasets:

### *A. Credit Card Default:*

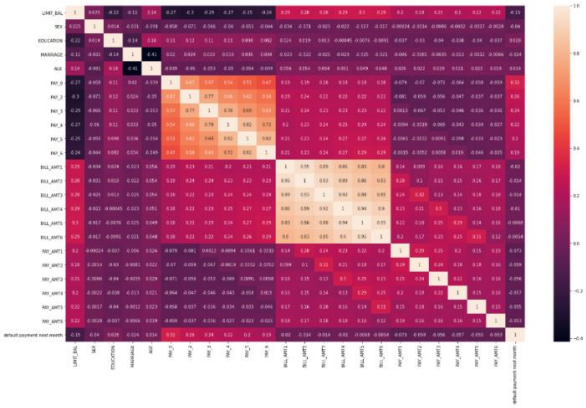

Fig 1. Correlation Matrix Plot

```
Confusion Matrix:
 [[5609  303]
 [1063  525]]
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      5912
           1       0.63      0.33      0.43      1588

    accuracy                           0.82      7500
   macro avg       0.74      0.64      0.66      7500
weighted avg       0.80      0.82      0.79      7500

Accuracy: 0.8178666666666666
Recall: 0.8406774580335732
Precision: 0.9487483085250338
```

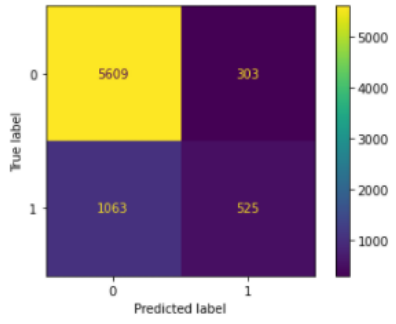


Fig. 2. KNN Confusion Matrix

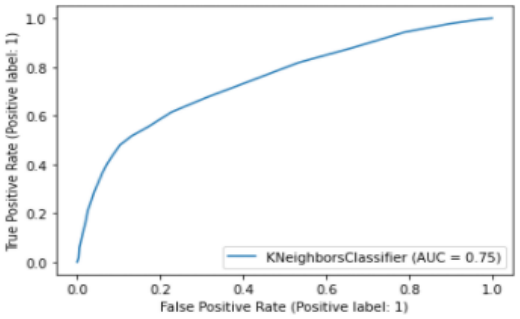


Fig. 3. KNN ROC Curve

```
Confusion Matrix:
 [[5671  241]
 [1058  530]]
              precision    recall  f1-score   support

           0       0.84      0.96      0.90      5912
           1       0.69      0.33      0.45      1588

    accuracy                           0.83      7500
   macro avg       0.77      0.65      0.67      7500
weighted avg       0.81      0.83      0.80      7500

Accuracy: 0.8268
Recall: 0.8427700995690296
Precision: 0.9592354533152909
```

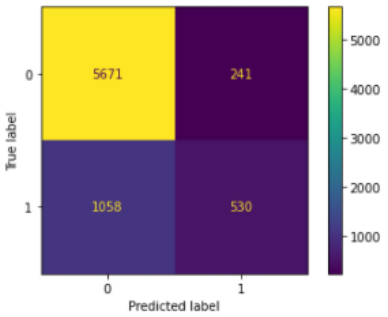


Fig. 4. Logistic Regression Confusion Matrix

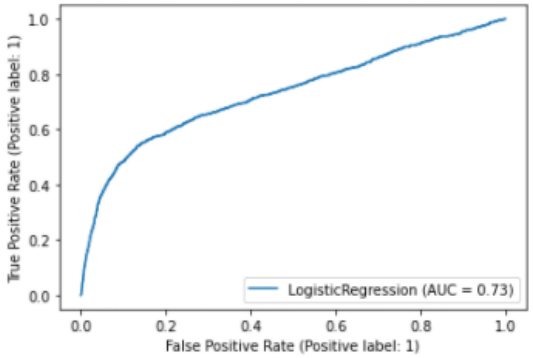


Fig. 5. Logistic Regression ROC Curve

```
Confusion Matrix:
 [[3429 2483]
 [ 726  862]]
              precision    recall  f1-score   support

           0       0.83      0.58      0.68      5912
           1       0.26      0.54      0.35      1588

    accuracy                           0.57      7500
   macro avg       0.54      0.56      0.52      7500
weighted avg       0.71      0.57      0.61      7500

Accuracy: 0.5721333333333334
Recall: 0.8252707581227436
Precision: 0.5800067658998647
```

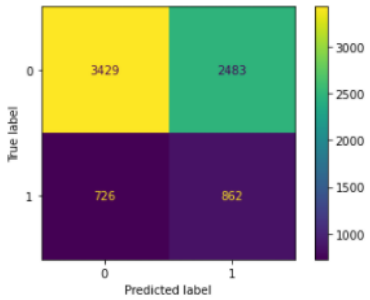


Fig. 6. Decision Tree Confusion Matrix

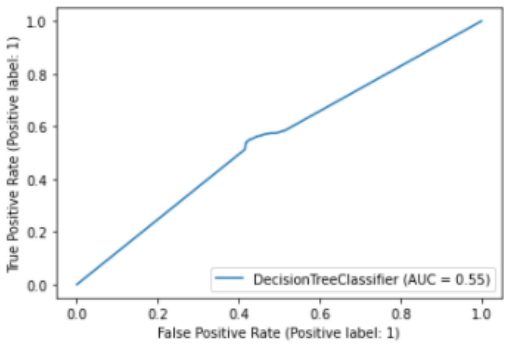


Fig. 7. Decision Tree ROC Curve

```
Confusion Matrix:
 [[5731  181]
 [1184  404]]
              precision    recall  f1-score   support

           0       0.83      0.97      0.89      5912
           1       0.69      0.25      0.37      1588

    accuracy                           0.82      7500
   macro avg       0.76      0.61      0.63      7500
weighted avg       0.80      0.82      0.78      7500

Accuracy: 0.818
Recall: 0.8287780187997108
Precision: 0.9693843031123139
```

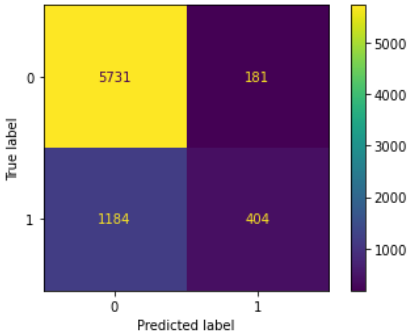


Fig. 8. Random Forest Confusion Matrix

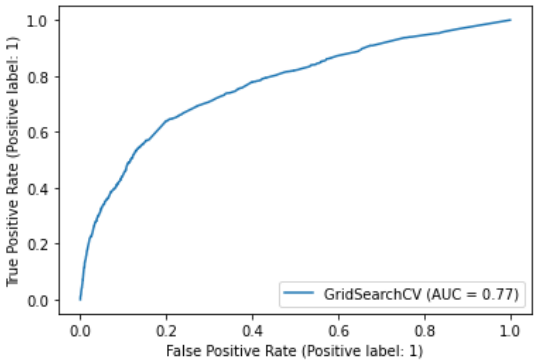


Fig. 9. Random Forest ROC Curve

```
Confusion Matrix:
 [[5750  162]
 [1215  373]]
              precision    recall  f1-score   support

           0       0.83      0.97      0.89      5912
           1       0.70      0.23      0.35      1588

    accuracy                           0.82      7500
   macro avg       0.76      0.60      0.62      7500
weighted avg       0.80      0.82      0.78      7500

Accuracy: 0.8164
Recall: 0.8255563531945441
Precision: 0.9725981055480379
```

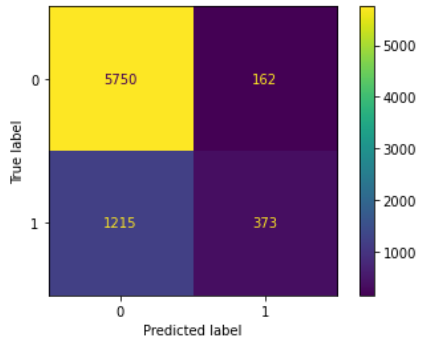


Fig. 10. SVC Confusion Matrix

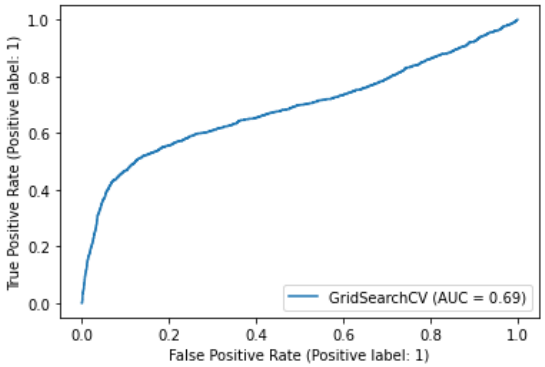


Fig. 11. SVC ROC Curve

```
Accuracy of: 0.756, std: (+/-) 0.007 [KNeighborsClassifier]
Accuracy of: 0.783, std: (+/-) 0.002 [Bagging KNeighborsClassifier]

Accuracy of: 0.779, std: (+/-) 0.001 [LogisticRegression]
Accuracy of: 0.781, std: (+/-) 0.002 [Bagging LogisticRegression]

Accuracy of: 0.727, std: (+/-) 0.011 [DecisionTreeClassifier]
Accuracy of: 0.803, std: (+/-) 0.007 [Bagging DecisionTreeClassifier]

Accuracy of: 0.816, std: (+/-) 0.010 [RandomForestClassifier]
Accuracy of: 0.815, std: (+/-) 0.008 [Bagging RandomForestClassifier]

Accuracy of: 0.779, std: (+/-) 0.000 [SVC]
Accuracy of: 0.779, std: (+/-) 0.000 [Bagging SVC]
```

Fig. 12. Bagging with all Classifiers

```
Accuracy: 0.750 (+/- 0.01) [KNN]
Accuracy: 0.776 (+/- 0.00) [Logistic Regression]
Accuracy: 0.722 (+/- 0.01) [Decision Tree]
Accuracy: 0.813 (+/- 0.01) [Random Forest]
Accuracy: 0.776 (+/- 0.00) [SVC]
Accuracy: 0.787 (+/- 0.00) [Ensemble]
```

Fig. 13. Bagging Results

```
Accuracy: 0.817, std: (+/-) 0.009 [Ada Boost]
Accuracy: 0.821, std: (+/-) 0.011 [Grad Boost]
Accuracy: 0.815, std: (+/-) 0.009 [XG Boost]
Accuracy: 0.821, std: (+/-) 0.010 [Ensemble]
```

Fig. 14. Boosting with all Classifiers Result

## *B. House Sales in King County, USA:*

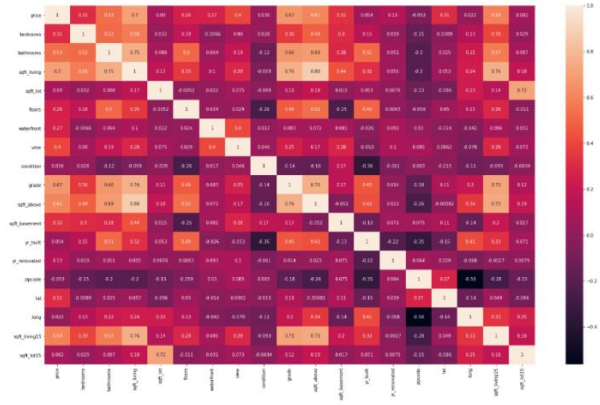

Fig. 15. Correlation Matrix Plot

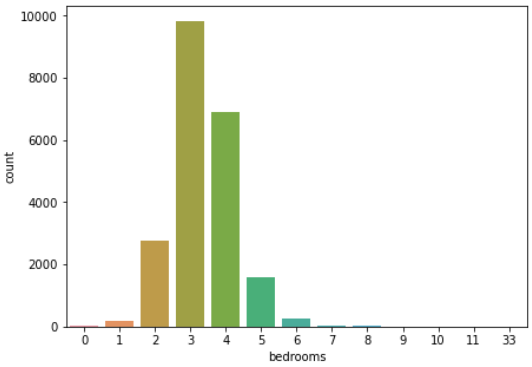


Fig. 16. Bedroom Count

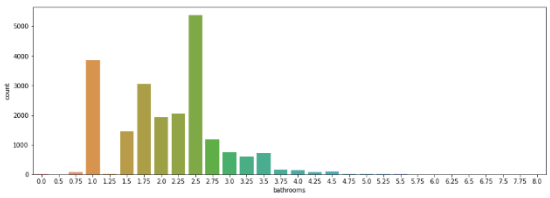

Fig. 17. Bathrooms Count

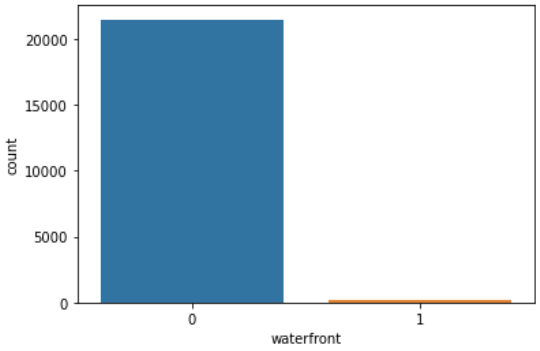


Fig 18. Waterfront Count

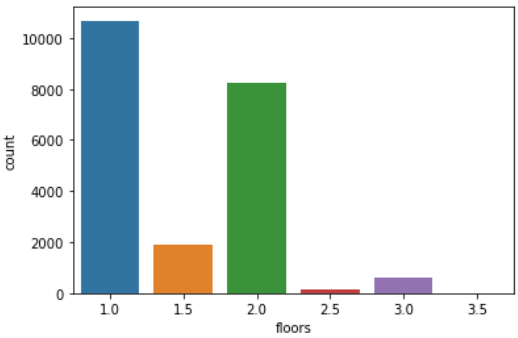


Fig 19. Floors Count

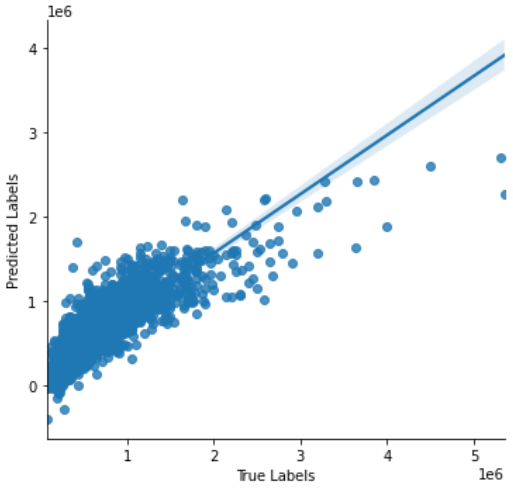


Fig. 20. Multiple Linear Regression

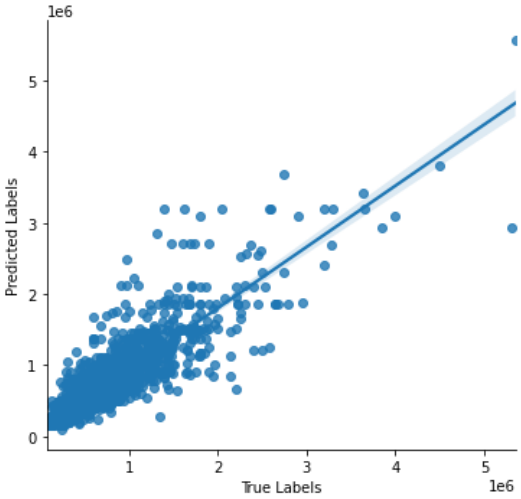


Fig. 21. Decision Tree Regressor

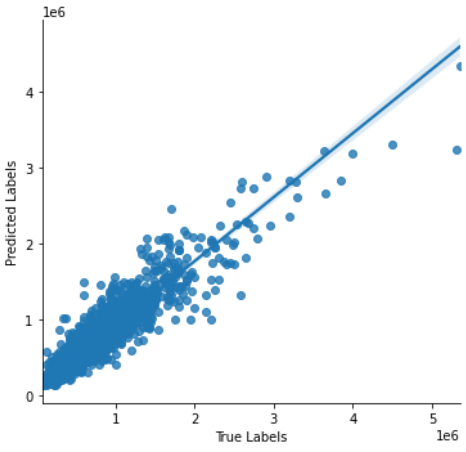


Fig. 22. Random Forest Regressor

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | price | R-squared: | 0.700 |
| Model: | OLS | Adj. R-squared: | 0.700 |
| Method: | Least Squares | F-statistic: | 2960. |
| Date: | Wed, 28 Apr 2021 | Prob (F-statistic): | 0.00 |
| Time: | 07:24:54 | Log-Likelihood: | -2.9460e+05 |
| No. Observations: | 21613 | AIC: | 5.892e+05 |
| Df Residuals: | 21595 | BIC: | 5.894e+05 |
| Df Model: | 17 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | 6.69e+06 | 2.93e+06 | 2.282 | 0.022 | 9.44e+05 | 1.24e+07 |
| bedrooms | -3.577e+04 | 1891.843 | -18.906 | 0.000 | -3.95e+04 | -3.21e+04 |
| bathrooms | 4.114e+04 | 3253.678 | 12.645 | 0.000 | 3.48e+04 | 4.75e+04 |
| sqft_living | 110.4408 | 2.270 | 48.661 | 0.000 | 105.992 | 114.889 |
| sqft_lot | 0.1286 | 0.048 | 2.683 | 0.007 | 0.035 | 0.223 |
| floors | 6689.5501 | 3595.859 | 1.860 | 0.063 | -358.599 | 1.37e+04 |
| waterfront | 5.83e+05 | 1.74e+04 | 33.580 | 0.000 | 5.49e+05 | 6.17e+05 |
| view | 5.287e+04 | 2140.055 | 24.705 | 0.000 | 4.87e+04 | 5.71e+04 |
| condition | 2.639e+04 | 2351.461 | 11.221 | 0.000 | 2.18e+04 | 3.1e+04 |
| grade | 9.589e+04 | 2152.789 | 44.542 | 0.000 | 9.17e+04 | 1e+05 |
| sqft_above | 70.7873 | 2.253 | 31.414 | 0.000 | 66.371 | 75.204 |
| sqft_basement | 39.6597 | 2.647 | 14.985 | 0.000 | 34.472 | 44.847 |
| yr_built | -2620.2232 | 72.659 | -36.062 | 0.000 | -2762.640 | -2477.806 |
| yr_renovated | 19.8126 | 3.656 | 5.420 | 0.000 | 12.647 | 26.978 |
| zipcode | -582.4199 | 32.986 | -17.657 | 0.000 | -647.074 | -517.765 |
| lat | 6.027e+05 | 1.07e+04 | 56.149 | 0.000 | 5.82e+05 | 6.24e+05 |
| long | -2.147e+05 | 1.31e+04 | -16.349 | 0.000 | -2.4e+05 | -1.89e+05 |
| sqft_living15 | 21.6814 | 3.448 | 6.289 | 0.000 | 14.924 | 28.439 |
| sqft_lot15 | -0.3826 | 0.073 | -5.222 | 0.000 | -0.526 | -0.239 |

| | | | |
|---|---|---|---|
| Omnibus: | 18384.201 | Durbin-Watson: | 1.990 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 1868224.491 |
| Skew: | 3.566 | Prob(JB): | 0.00 |
| Kurtosis: | 47.985 | Cond. No. | 8.67e+16 |

Fig. 23. StatsModel OLS

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | price | R-squared (uncentered): | 0.905 |
| Model: | OLS | Adj. R-squared (uncentered): | 0.905 |
| Method: | Least Squares | F-statistic: | 1.287e+04 |
| Date: | Wed, 28 Apr 2021 | Prob (F-statistic): | 0.00 |
| Time: | 07:25:35 | Log-Likelihood: | -2.9461e+05 |
| No. Observations: | 21613 | AIC: | 5.892e+05 |
| Df Residuals: | 21597 | BIC: | 5.894e+05 |
| Df Model: | 16 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| bedrooms | -3.562e+04 | 1887.390 | -18.872 | 0.000 | -3.93e+04 | -3.19e+04 |
| bathrooms | 4.234e+04 | 3136.467 | 13.500 | 0.000 | 3.62e+04 | 4.85e+04 |
| sqft_living | 109.9055 | 2.256 | 48.724 | 0.000 | 105.484 | 114.327 |
| sqft_lot | 0.1312 | 0.048 | 2.742 | 0.006 | 0.037 | 0.225 |
| waterfront | 5.833e+05 | 1.74e+04 | 33.600 | 0.000 | 5.49e+05 | 6.17e+05 |
| view | 5.249e+04 | 2126.373 | 24.686 | 0.000 | 4.83e+04 | 5.67e+04 |
| condition | 2.691e+04 | 2315.359 | 11.624 | 0.000 | 2.24e+04 | 3.15e+04 |
| grade | 9.581e+04 | 2133.801 | 44.903 | 0.000 | 9.16e+04 | 1e+05 |
| sqft_above | 72.5890 | 2.088 | 34.763 | 0.000 | 68.496 | 76.682 |
| sqft_basement | 37.3165 | 2.407 | 15.506 | 0.000 | 32.599 | 42.033 |
| yr_built | -2544.6464 | 67.021 | -37.968 | 0.000 | -2676.013 | -2413.280 |
| yr_renovated | 20.6412 | 3.643 | 5.666 | 0.000 | 13.500 | 27.782 |
| zipcode | -521.7152 | 17.738 | -29.413 | 0.000 | -556.482 | -486.948 |
| lat | 6.036e+05 | 1.07e+04 | 56.456 | 0.000 | 5.83e+05 | 6.25e+05 |
| long | -2.192e+05 | 1.3e+04 | -16.824 | 0.000 | -2.45e+05 | -1.94e+05 |
| sqft_living15 | 22.3571 | 3.355 | 6.664 | 0.000 | 15.782 | 28.932 |
| sqft_lot15 | -0.3807 | 0.073 | -5.204 | 0.000 | -0.524 | -0.237 |

| | | | |
|---|---|---|---|
| Omnibus: | 18359.519 | Durbin-Watson: | 1.991 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 1857062.968 |
| Skew: | 3.560 | Prob(JB): | 0.00 |
| Kurtosis: | 47.850 | Cond. No. | 1.96e+17 |

Fig. 24. StatsModel OLS after removing ‘floor’

### *C. YouTube Trending Videos:*

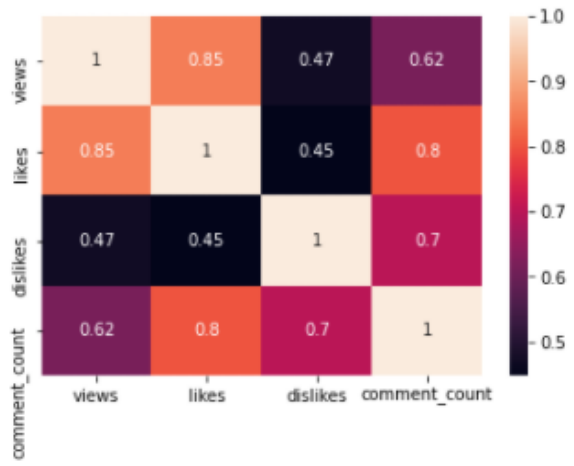


Fig. 25. Correlation Matrix Plot

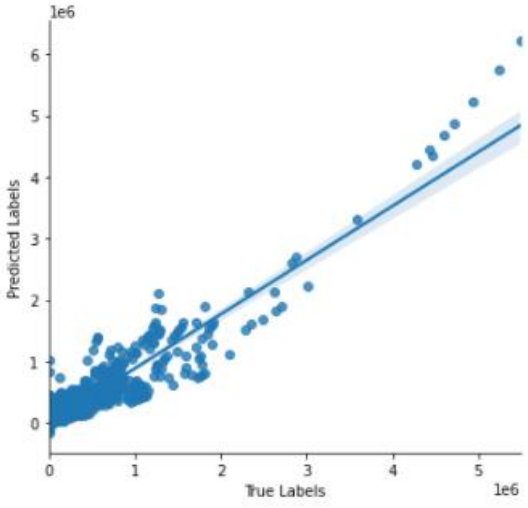


Fig. 26. Multiple Linear Regression

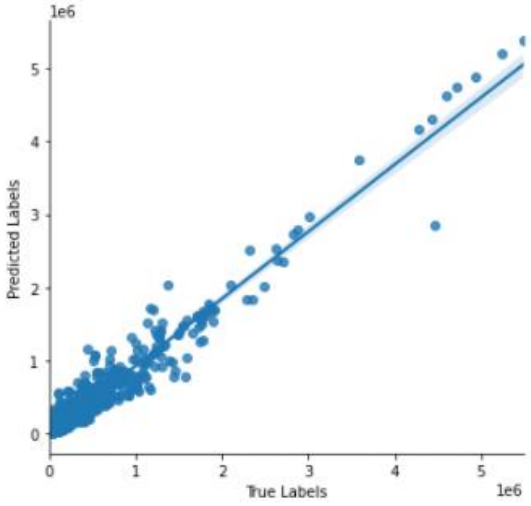


Fig. 27. XGB Regressor

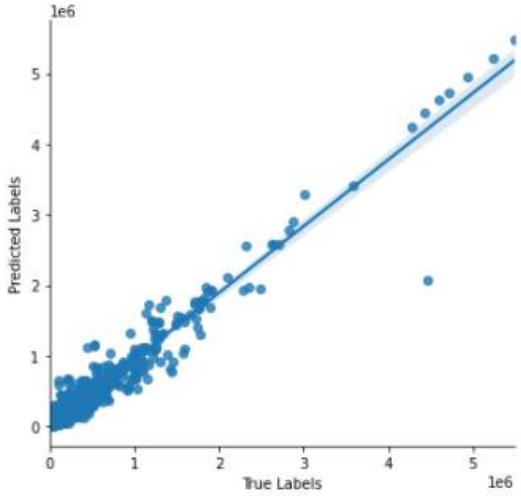


Fig. 28. Random Forest Regressor