CSE 587 DATA INTENSIVE COMPUTING-PROJECT PHASE 2
PREDICTING THE EXISTING UNITS IN SANFRANCISCO BUILDING PERMITS DATA

REVISED GOAL:

The goal of this project is to predict the "Existing Units" of the buildings in the San Francisco data. The "Existing Units" in the dataset is a numerical type inconsistent data which is further cleaned and pre-processed. This is a Regression Model

DATA PRE-PROCESSING:

In the phase 1, general data preprocessing was carried out like the missing values are filled based on the percentage. The zip codes are filled based on the latitude and longitude. 10 columns were dropped since those columns are least important and their impact to the model is very low. The missing values in the "Existing Units" feature was filled and the inconsistent data was scaled down to a definite range. All the input features are in float or int format, also the categorical variables are encoded.

FEATURE SELECTION:

The features are selected by checking the correlation metric for the target variable with each other features, since if any of the feature is highly correlated with the target variable it may lead to overfitting and the prediction accuracy will be extremely high. Hence to overcome this, the features with high correlation are removed. For our model, the confusion matrix was checked and from that it is inferred that there are 9 features that have less correlation with the target variable.

SPLITTING THE DATA:

The train and test data for modeling is split using sklearn "`model_selection`". The test size is set to 0.1. This function randomly shuffles the data and partitions it into training and test sets based on a specified ratio or number of samples

MODELS USED:

The following 7 machine learning models are used to predict and analyze the "Existing Units":

→ Decision Tree
→ Random Forest
→XGBoost Regressor
→Lightgbm Regressor
→AdaBoost Regressor
→Neural Networks
→Linear Regression

MODELING & ANALYSIS:

1.DECISION TREE:

WHY DECISION TREE?

Decision trees can be used for regression modeling due to their ability to capture non-linear relationships between features and target variables. One main reason this used to model our dataset is that decision tree are robust to outliers. Unlike linear regression models, which assume a linear relationship between features and targets, decision trees can capture complex and non-linear patterns in the data. This makes decision trees suitable for regression tasks where the relationships between features and targets are not strictly linear.

HYPERPARAMTER TUNING:

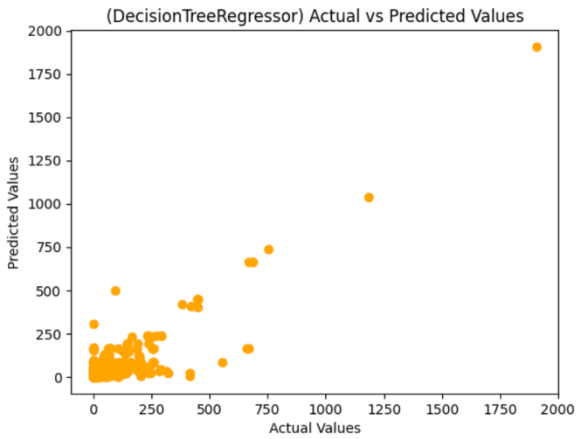
The model's r2 score was low at the starting and after several phases of hyperparameter tuning the r2 score got improved. One such impact was the max_depth parameter, when the depth of the decision tree was increased there was a steady increase in the r2 score of the model. Adding to this, the model cannot accept missing values in both the target and input features. The random_state parameter also has an impact and on different values of random state, the r2 score changed significantly.

RESULTS:

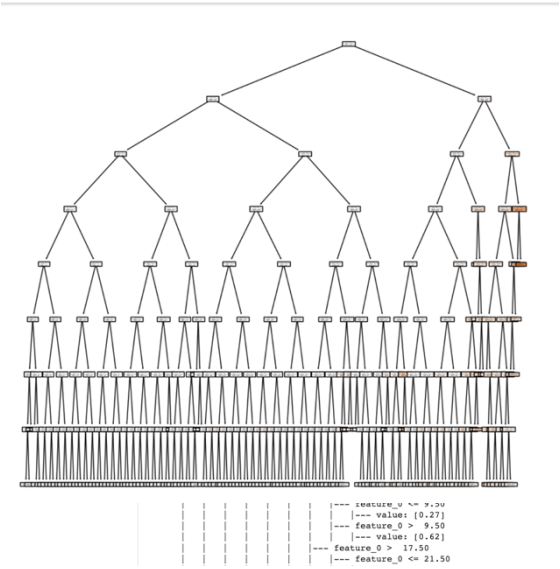
In our model, the decision tree function from sklearn is used with parameters such as max_depth, random state are used. When the model is fit against x_train and y_train data, the model returns a r2 score of 0.76 and MSE of 1056.24. An R2 score of 0.7622 indicates that the model explains about 76.22% of the variability in the "Existing Units", while the remaining 23.78% of the variability is still unexplained or attributed to other factors.

```
print("Mean Squared Error (Decision tree): ", mse)
print("R-squared (Decision tree):  ", r2)
```

```
Mean Squared Error (Decision tree):  1056.245720837943
R-squared (Decision tree):   0.7630316521827423
```



From the Actual vs Predicted graph we can infer that the model is learning in an increasing fashion, there are some noise in the starting. Since the accuracy is 76% the graph is in such a way that it is increasing but not the exact ideal one.



From the text tree and the decision tree, Following the path from the root node ("Existing Units") to a leaf node in the decision tree diagram can show the specific sequence of decisions or splits that are made by the model to arrive at a prediction for a particular input instance. The prediction process is shown here.

Note: refer code for clear picture of the output

EFFECTIVENESS OF THE MODEL:

The model was highly effective since when other models such as linear regression model was not able to capture all details and made the best results, Decision Tree made the better results because though the model has a inconsistent spread of data, decision tree handled in a unique way and was robust to outliers.

2. RANDOM FOREST:

WHY RANDOM FOREST?

Random Forest is used for this data because it has been shown to produce high accuracy in a wide range of regression tasks. It is a combination of multiple decision trees, each trained on a subset of the data, and then combines their predictions, which can result in better overall accuracy compared to a single decision tree.

HYPERPARAMTER TUNING:

Similar tuning was done to Random Forest like Decision Tree. But when compared to decision tree, a higher r2 score was achieved in few steps of hyper-parameter tuning. Sklearn function is used with a parameter called estimators with value set to 100. Changing the estimators value had a mild impact in the r2 score. To check the effectiveness, one input feature with 12 missing values was added to the input. The random forest model performed well irrespective of the missing values.

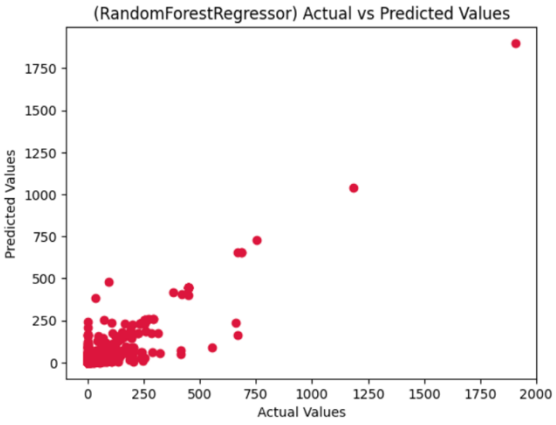
RESULTS:

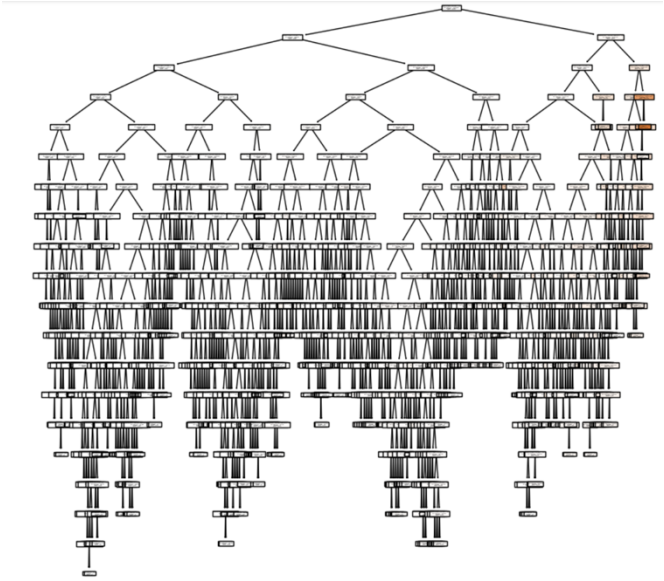
In our model, the random forest function from sklearn is used with parameters such as estimators are used. When the model is fit against x_train and y_train data, the model returns a r2 score of 0.78 and MSE of 941.5977.

```
print("Mean Squared Error: {:.4f}".format(mse))
print("R-squared Score: {:.4f}".format(r2))
```

```
Mean Squared Error: 941.5977
R-squared Score: 0.7888
```

An R2 score of 0.7888 indicates that the model explains about 78.88% of the variability in the "Existing Units", while the remaining 21.720% of the variability is still unexplained or attributed to other factors.





From the Actual vs Predicted graph we can infer that the model is learning in an increasing fashion, there are some noises in the starting. Since the accuracy is 78% the graph is in such a way that it is increasing but not the exact ideal one. On further preprocessing there can be perfect graph.

From the decision tree for one estimator of a random forest, Following the path from the root node ("Existing Units") to a leaf node in the decision tree diagram can show the specific sequence of decisions or splits that are made by the model to arrive at a prediction for a particular input instance. The prediction process is shown above.

EFFECTIVENESS OF THE MODEL:

To check the effectiveness, one input feature with 12 missing values was added to the input. The random forest model performed well irrespective of the missing values. This shows random forest can handle missing values.

By default, no extra works was done for a higher accuracy, the random forest itself yield higher accuracy. Like decision tree, the random forest algorithm is robust to outliers.

Random Forest uses an ensemble approach, combining the predictions of multiple decision trees.

3. XGBOOST REGRESSOR:

WHY XGBOOST?

XGBoost is used for out data because it is known for its exceptional performance in various machine learning task. It is optimized for speed and efficiency, making it one of the fastest gradient boosting algorithms available. XGBoost uses a combination of techniques, such as parallelization, column block-ing, and cache-aware access, to achieve superior performance and reduced computation.

HYPERPARAMTER TUNING:

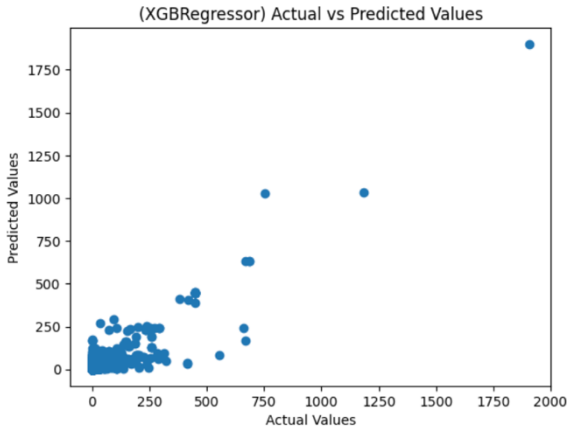
But when compared to decision tree and random forest, a higher r2 score was achieved in few steps of hyper-parameter tuning. Sklearn function XGBregressor was used. The execution time of this regressor was extremely less. Out of all, XGBoost gave the highest r2 score. Change of parameters does not have any significant change in the r2 score or MSE.

RESULTS:

In our model, XGBoost the function from sklearn is used with no parameters. When the model is fit against x_train and y_train data, the model returns a r2 score of 0.79 and MSE of 934.9612. An R2 score of 0.7888 indicates that the model explains about 79.02% of the variability in the "Existing Units", while the remaining 22.720% of the variability is still unexplained or attributed to other factors.

```
print("Mean Squared Error (XGBRegressor): {:.4f}".format(mse))
print("R-squared Score (XGBRegressor) : {:.4f}".format(r2))
```

```
Mean Squared Error (XGBRegressor): 934.9612
R-squared Score (XGBRegressor) : 0.7902
```



From the Actual vs Predicted graph we can infer that the model is learning in an increasing fashion, there are some noises in the starting. Since the accuracy is 79% the graph is in such a way that it is increasing but not the exact ideal one. On further preprocessing there can be perfect graph.

The relationship and how good our prediction can be inferred from the above graph, since our data has more features that are irrelevant and does not contribute to the modeling in anyway,

and almost 40 percent of the data has missing values. All these factors have a major drawback and can put down the model accuracy followed by not getting a ideal result graphs.

EFFECTIVENESS OF THE MODEL:

Out of all the 7 models, XGBoost gave the highest r2 prediction score, this is because the model can by default handle missing values.

Though our data has missing values, the model was able to capture, rectify and made the best use of them. This model is highly able to give good accuracy score with a high execution speed.

4. LGBM REGRESSOR:

Why LightGBM Regressor?

LGBM Regressor refers to the LightGBM Regressor, is used for our prediction, is a regression model implemented in the LightGBM library. LightGBM is gradient boosting framework that is known for its efficiency and speed, making it a popular choice for many machine learning tasks mainly for regression. This is similar to the XGBoost framework. It is used especially when dealing with large datasets and complex patterns in the data.

HYPERPARAMTER TUNING:

A higher r2 score was achieved in few steps of hyper-parameter tuning. Sklearn function lgbmregressor was used. The execution time of this regressor was extremely less. Out of all, XGBoost gave the highest r2 score. Change of parameters does not have any significant change in the r2 score or MSE but there was a mild change in the results. When the objective parameter was set to multiclass, there was a slight increase in the r2 score.

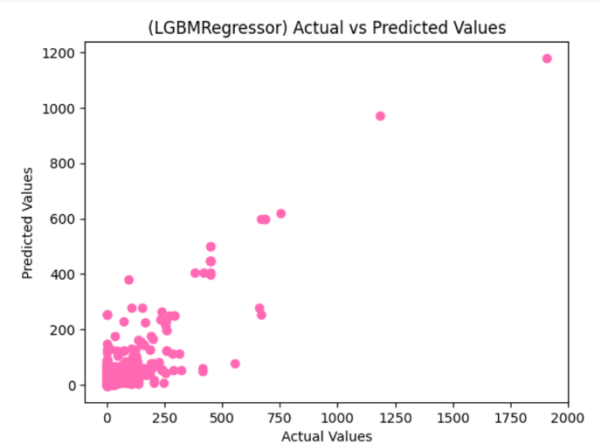
RESULTS:

In our model, LightGBM Regressor the function from sklearn is used with no parameters. When the model is fit against x_train and y_train data, the model returns a r2 score of 0.79 and MSE of 934.9612.

An R2 score of 0.7506 indicates that the model explains about 75.06% of the variability in the "Existing Units", while the remaining 24.920% of the variability is still unexplained or attributed to other factors.

```
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (LGBMRegressor):{:.4f} ".format(mse))
r2 = r2_score(y_test, y_pred)
print("R-squared Score(LGBMRegressor) : {:.4f}".format(r2))
```

```
Mean Squared Error (LGBMRegressor):1111.5430
R-squared Score(LGBMRegressor) : 0.7506
```



From the Actual vs Predicted graph we can infer that the model is learning in an increasing fashion, there are some noises in the starting. Since the accuracy is 75% the graph is in such a way that it is increasing but not the exact ideal one. On further preprocessing there can be perfect graph. When compared to other algorithms since the accuracy is less, a mild negative difference in the graph can be witnessed.

EFFECTIVENESS OF THE MODEL:

Although this model does not give the highest accuracy when compared to the other models, LightGBM is designed to be efficient and has a low memory footprint, making it suitable for large datasets. It uses a histogram-based algorithm for splitting features, which can speed up training time and reduce memory usage compared to other gradient boosting algorithms.

It used a gradient based approach, which gives the ability to handle complex data and capture most of the insightful patterns in the dataset.

5. ADABOOST:

Why ADAboost?

ADABoost is also an ensemble method and we have used adaboost in our model because of its adaptive approach. It has the ability to adjust the weights of the misclassified samples during training. Like all above algorithms, adaboost is also known for its unique way of handling outliers and missing data. It is basically combining the predictions of multiple base models to create a strong overall model.

HYPERPARAMTER TUNING:

Out of our first five models, adaboost yielded the least accuracy after hyper parameter tuning. It was just 50 percent of the accuracy score which has high chances of being wrongly predicted. The execution time of this regressor was extremely less.

Change of parameters does not have any significant change in the r2 score or MSE but there was a mild change in the results.

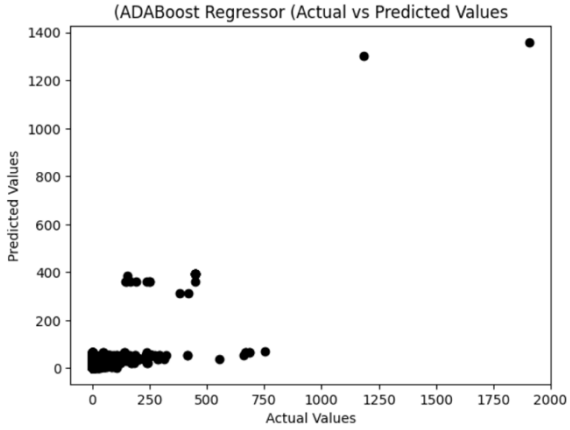
RESULTS:

In our model, adaboost Regressor the function from sklearn is used with no parameters. When the model is fit against x_train and y_train data, the model returns a r2 score of 0.52 and MSE of 2095.9612.

```
print("Mean Squared Error (AdaBoostRegressor): {:.4f}".format(mse))
print("R-squared Score (AdaBoostRegressor): {:.4f}".format(r2))
```

```
Mean Squared Error (AdaBoostRegressor): 2095.4223
R-squared Score (AdaBoostRegressor): 0.5299
```

An R2 score of 0.52 indicates that the model explains about 52.99% of the variability in the "Existing Units", while the remaining 47.920% of the variability is still unexplained or attributed to other factors.



From the Actual vs Predicted graph we can infer that the model is learning in an non-increasing fashion, there are some noises in the starting and also in the end. Since the accuracy is 53% the graph is in such a way that it is clumsy but not the exact ideal one. On further preprocessing there can be perfect graph. When compared to other algorithms since the accuracy is less, a negative difference in the graph can be witnessed.

Possible Reasons for Low Accuracy:

The imbalance in the data can be one primary reason for the low accuracy. Our dataset has highly imbalanced data, some models were able to capture despite the imbalance in the data but adaboost has failed.

There is possible chance that our feature selection is not up to the mark, more possible features can be checked and added to the input with a lower correlation factor.

If the distribution of the test data is significantly different from the training data, it can result in low accuracy. AdaBoost may not perform well on unseen data if the distribution of the test data differs from the training data.

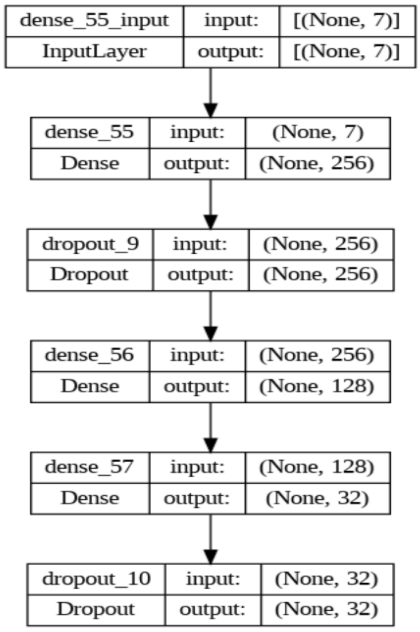
Though several hyper-parameters were checked and tuned, the best pair may no be identified that can also be a major reason for low accuracy.

ADDITIONAL MODELS:

NEURAL NETWORKS:

Neural networks are used for this data and basically Neural nets are a means of doing machine learning, in which a computer learns to perform some task by analyzing training examples. We have used the same test and train split data to model the neural network. Neural networks was carried out by keras framework.

STRUCTURE OF THE NEURAL NETWORK:



This is flowchart representation of our neural network layers. Dropout is used in-order to reduce overfitting.

RESULTS:

```
Epoch 1/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0358 - accuracy: 0.5992
Epoch 2/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0343 - accuracy: 0.5997
Epoch 3/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0322 - accuracy: 0.5998
Epoch 4/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0380 - accuracy: 0.5989
Epoch 5/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0352 - accuracy: 0.5995
Epoch 6/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0358 - accuracy: 0.5991
Epoch 7/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0388 - accuracy: 0.5989
Epoch 8/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0332 - accuracy: 0.5998
Epoch 9/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0330 - accuracy: 0.5996
Epoch 10/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0350 - accuracy: 0.5991
Epoch 11/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0351 - accuracy: 0.5997
Epoch 12/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0345 - accuracy: 0.5999
Epoch 13/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0356 - accuracy: 0.5988
Epoch 14/50
221/221 [==============================] - 1s 5ms/step - loss: 0.0341 - accuracy: 0.5994
Epoch 15/50
221/221 [==============================] - 1s 4ms/step - loss: 0.0347 - accuracy: 0.6001
Epoch 16/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0325 - accuracy: 0.6000
Epoch 17/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0351 - accuracy: 0.5998
Epoch 18/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0361 - accuracy: 0.5993
Epoch 19/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0357 - accuracy: 0.5997
Epoch 20/50
221/221 [==============================] - 1s 3ms/step - loss: 0.0366 - accuracy: 0.5992
```

The model was trained using the loss=”sparse categorical crossentropy”, optimizer as sgd and ran 50 epochs, the loss and accuracy after every epoch is given in the image.

```
[241] test_loss, test_acc = model1.evaluate(nnx_test,nny_test)
      test_acc= test_acc*100
      print("The test accuracy :",test_acc)

      98/98 [==============================] - 1s 3ms/step - loss: 3.3379e-06 - accuracy: 0.6111
      The test accuracy : 61.10578179359436
```

After 50 epochs the test accuracy is 61 percent, which is not the best, relatively due to some short coming and inability of neural networks

SVM AND LINEAR REGRESSION:

Both SVM and linear regression was tried for our data but unfortunately both the algorithms gave very low accuracy when compared to all other models. The same set of train and test data was used for both the models but the models did not perform well due to some inabilities by default.

ISSUES ENCOUNTERED:

i. While using SVM regressor, the execution time was extremely high. While other models took seconds to execute, SVM took 20 mins to fit the model.

ii. Linear regression cannot handle outliers, the outliers have major impact on accuracy. Since this dataset is prone to outliers, there is low accuracy.

iii. Both the algorithm cannot handle missing values, if there is a single missing value, the compiler pops out error and was unable to execute.

iv. Both the algorithm are incapable of performing well on large datasets. Since our dataset is large and clumsy they require high pre-processing and data cleaning for better results.

OVERALL INSIGHTS FROM ALL THE 8 MODELS:

i. Out of all, XGBoost model attained the highest r2 score which shows the capability of the model to perform well, irrespective of the nature of the data.

ii. Both SVM and linear regression was tried for our data but unfortunately both the algorithms gave very low accuracy when compared to all other models.

iii. Adaboost was a similar model when compared to the ones that yielded high r2 score but unfortunately it attained comparatively less accuracy than the relative ones.

iv. Neural networks also did not turn out to our favor, it signifies that it requires more preprocessed data without any imbalances in it.

XGBoost < RandomForest < DecisionTree < LightGBM < Adaboost < NeuralNetworks

KEY ISSUES IN THE DATASET:

I. The dataset being highly inconsistent is major drawback for the models, the dataset has around 200k values which relatively higher in number which some models are unable to perform well.

II. The features in the dataset are multivariate and also contains time and date series data in it. Those features are processed although they do not contribute to the models used.

III. The missing values in the dataset are extremely high and filling all these data is impossible which led to dropping many features.

IV. The data spread over one feature is highly imbalanced for eg, in a categorical feature, one class has 60,000 values and other class has only one sample, this leads to imbalance in the data.

CONTRIBUTION:

TEAM MEMBER	CONTRIBUTION
AKSHEY RAM MURALI	50%
VAISAALI MURALI KRISHNAN	50%

REFERENCES:

https://numpy.org/

https://pandas.pydata.org/docs/

https://scikit-learn.org/stable/supervised_learning.html

https://matplotlib.org/

https://seaborn.pydata.org/

https://www.itl.nist.gov/div898/handbook/eda/section1/eda11.html

https://mathshistory.st-andrews.ac.uk/Biographies/Tukey/,

C. O'Neill and R. Schutt. Doing Data Science., O'Reilly. 2013.