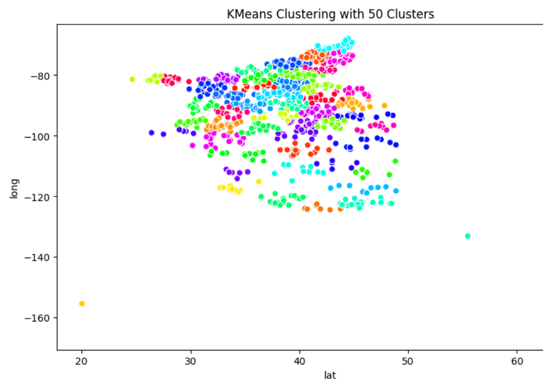
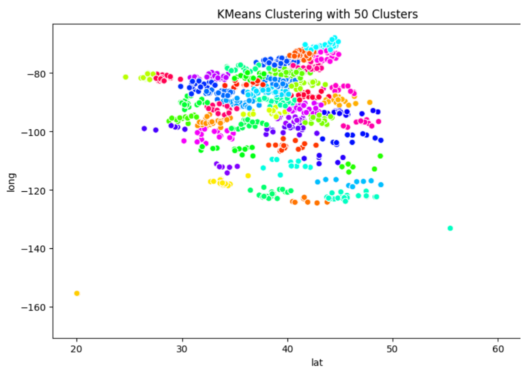
Wilbert Nathanael Limson
Kaggle Username: Wilb11

There are several steps that I took to solve this problem. The first step is **Input Data** through Google collab i pre process the data and load the trainingSet and testingSet. Moving on to the second step, **Preprocess Data** where I check if there are missing values, outliers or errors. Through this my finding is that all variables don't have any null values except “is_fraud” which we have to predict.

Afterwards the next crucial step is to **Explore and Analyze** the data, I have separated the data into Numerical and categorical data. In this I was able to find a finding on top 10 city with most fraud (Camden, Birmingham, Burton, Clarks Mills, Phoenix, Reynolds, Carlotta, Jay, Mound City, Bradley), top 10 job with most fraud, checking which category that is most affiliated with fraud transaction, which day on the week that is more likely to have fraud which is Sunday, while for the Time of the fraud happens most at 23:00 and 24:00. These data findings are useful to know which features are relevant in predicting the “is_fraud”. However, for me to use the categorical data, I have decided to perform one hot encoding and ordinal encoding.

The next step is to use Cluster Locations, because based on my understanding of data, simply measuring the distance from the “lat”,”long” till “merch_lat”, “merch_long” might create a noise because the distance of 50km can be towards a north direction, south, or other direction and might be a missed leading to predict where’s the fraud location. Therefore by creation a clustering based on the merch location and location of the card. I tried with different cluster size of 20, 30, 50. I have decided to move on with cluster of 50. Afterwards I created a new features which is the ‘cluster_owner’ and ‘cluster_merch’.





These increased my features to a total of 58 columns. With these If I decided to use all the features it will overfit therefore I decided to test the significance with p value less than 0.05 then it gives me the result which one that is significant and not to “is_fraud”. Another way to check this as well, I created a heatmap, however I find heatmap misses the distribution of the data and makes it too broad. Therefore the features that I used are amt, age, city_pop, cluster_owner, customer_avg_amt, customer_daily_freq, customer_fraud_rate, lat, merch_lat, merchant_fraud_history, and unix_time could be important predictors for detecting fraud.

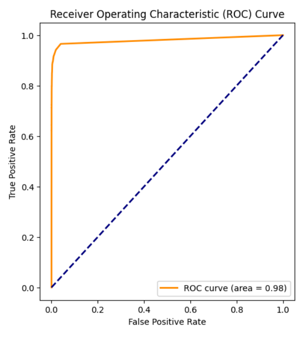
While the Not Significant Variables (P-value ≥ 0.05): Features are cc_num, cluster_merch, distance_to_merchant_km, long, merch_long, and zip do not show a statistically significant difference in means between the two groups. While the significant variables are Categorical features such as category, city, first, job, last, merchant, state, and street and the not Significant Variables: gender and trans_num. It reduces the dataset to 41 columns.

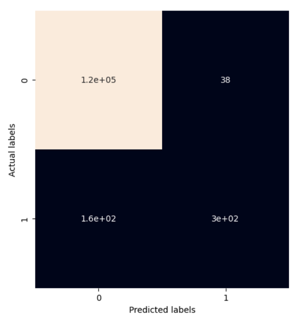
However I encountered a problem where since my laptop broke and I had to use online Collab, causing running a pipeline the categorical_transformer and numerical transformer data into a processed data became too long to continue but once it gave me Accuracy: 0.999, F1 Score: 0.999 AUC(ROC): 0.9321 however Time taken for model training and prediction: 7894.97 seconds. Therefore I decided to only process the data with numerical data only causing my training to be underfit from using 40 features to 29 features.

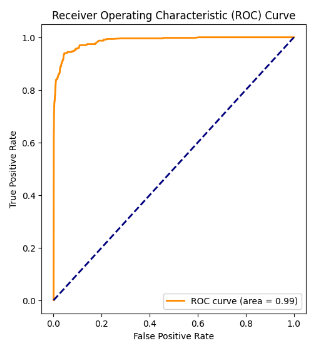
The next crucial step is to apply scaling and pca the data. However i have realized after PCA the data the features have reduced to a significant amount to 15 from 29 features. Therefore I decided to use scaled feature data instead. Then I defined my model, i have tried Logistic Regression, Gaussian NBm decision Tree, Random Forest and XGB boosting. These model then I compared the Accuracy, F1 score, and checking the area under the curve and Receiver Operating Characteristic. Here are the result,

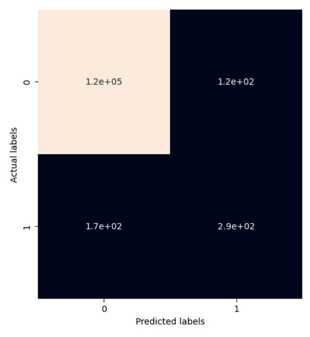
	Accuracies	F1 Scores	ROC AUC	Time Taken
LogisticReg	0.9962	0.9948	0.5365	1.9014
GaussianNB	0.9918	0.9932	0.7093	1.1180
DecisionTree	0.9952	0.9953	0.7088	23.8305
RandomForest	0.9978	0.9983	0.8285	92.4513
XGB	0.9976	0.9975	0.8163	11.8449

By this it is quite obvious that the best model is RandomForest and followed by XGB, in the graph bellow it shows that the legend indicates that the area under the ROC curve (AUC) is 0.98. suggesting the model has an excellent measure of separability.









The next step I attempted was to do a param_grid search with n_estimators that shows the number of trees in the forest. Two values I wanted to test are 100 and 200. And the next parameter is max_depth which is the maximum depth of the trees. I have set up three values to be tested - None (which means unlimited depth), 10, and 20. Afterwards I wanted to implement Set Up the Grid Search, I initialized GridSearchCV with the Random Forest model (model_rf) and the parameter grid (param_grid). That used 5-fold cross-validation (cv=5), which means the training data will be split into five parts, training the model five times on four parts and validating it on the fifth. And the scoring parameter is set to 'accuracy', which means that the grid search will evaluate the models using accuracy as the metric. The verbose parameter is set to 1, which means that the process will output detailed information during the fitting process. Afterwards fitting the Grid Search to the Data where the grid search would be fitted to the training data (X_train_pca and Y_train), allowing it to evaluate all possible combinations of parameters in the grid. And then fitting the grid search to the data, the best performing model configuration is accessed through grid_search.best_estimator_. However, since I broke my laptop, it caused the runtime to be too long and I had reached the maximum ram on the device. I decided to not pursue it but I believe it will increase my accuracy.

Afterwards, I decided to stack to improve my model, by combining two different models: a RandomForestClassifier and an XGBoost classifier (XGBClassifier). The XGBoost model is adjusted to optimize its performance by changing its settings to not use the label encoder and to evaluate its performance using 'log loss'. Afterwards I applied a Logistic Regression model to combine the predictions from these two base models into a final prediction. After timing the duration to train the combined model it shows that it’s quite practical and efficient in terms of the computation resources and time.

```
Accuracy:  0.9993254581948603
F1 Score:  0.9993056883341156
AUC(ROC):  0.9266006131963825

Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00    121114
         1.0       0.96      0.85      0.90       450

    accuracy                           1.00    121564
   macro avg       0.98      0.93      0.95    121564
weighted avg       1.00      1.00      1.00    121564
```

From the result of the stacking above we can see the improvement of Accuracy, F1 Score, and the AUC (ROC) which made the prediction accuracy go higher. Then we validate the model then submit our prediction to the Kaggle. There is a lot of room for improvement such as implementing the grid search before running the model, Using pipeline so we could use all the categorical dataset and find a better result.