Credit card fraud detection

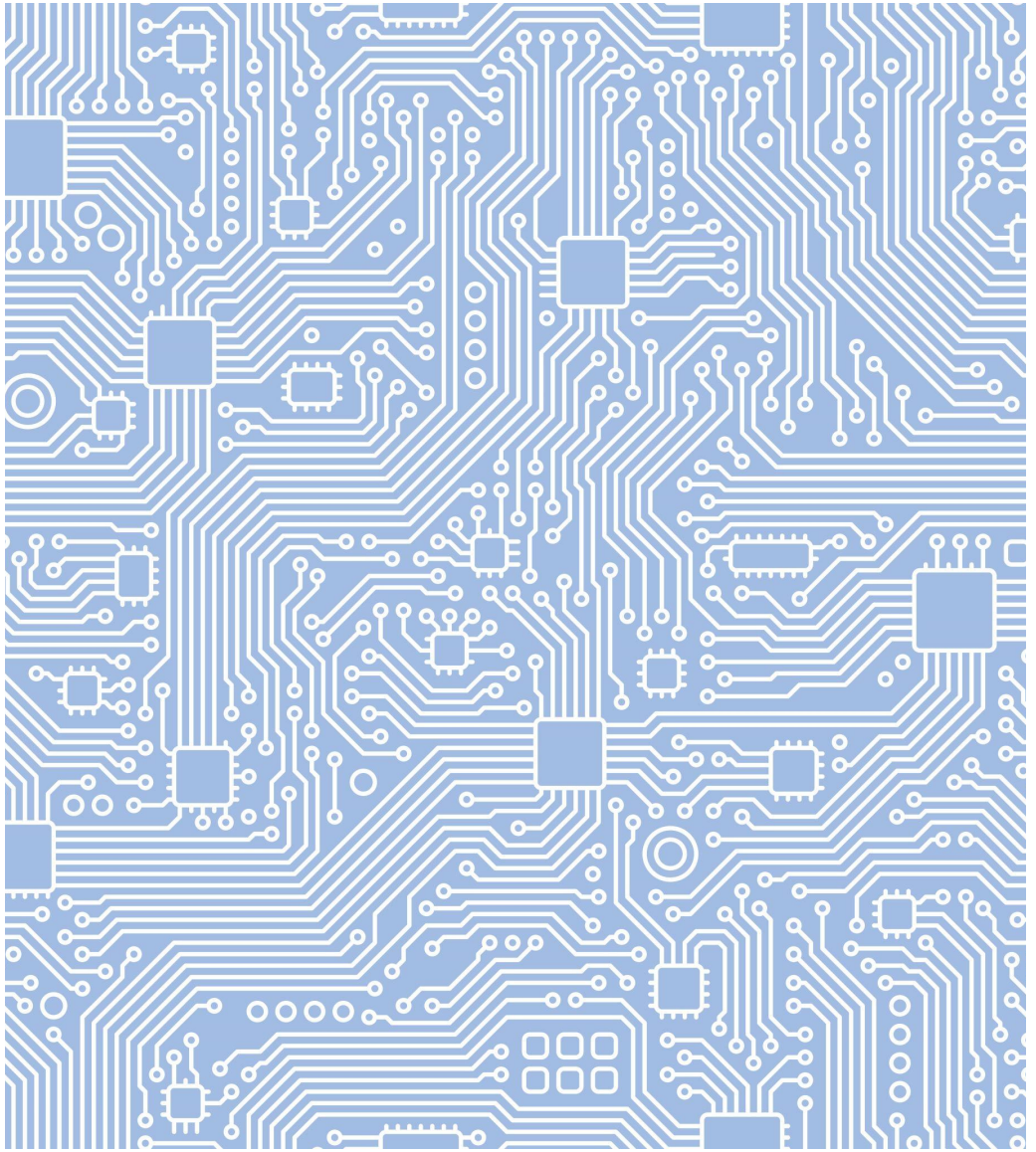
By:

Olu Olayeye

Daniel Saulsberry

Seth Boswell

Malli Montano



overview

The Annual Data Book compiled by the Federal Trade Commission reports that Credit card fraud accounted for 393,207 of the nearly 1.4 million reports of identity theft in 2020.

This makes credit card fraud the second most common type of identity theft reported, behind only government documents and benefits fraud for that year.

Some surveys suggest that a typical organization loses 5% of their yearly revenues to fraud. These numbers can only increase since the number of non-cash transactions increases, providing more opportunities for credit card fraud.

OVERVIEW

For retailers and banks to not lose money, procedures must be put in place to detect fraud prior to it occurring.

Credit card companies must identify fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

To combat this problem, financial institution traditionally uses rule-based approaches to identify fraudulent transactions.

These algorithms use strict rules to determine when a transaction is fraudulent.

Challenges of a strict ruled-based algorithm include:

Any new scenario that could lead to fraud needs to be manually coded into the algorithm.

Increases in customers and size of data leads to a corresponding increase in the human effort, time and cost required to track new scenarios and update the algorithm.

Since the algorithm cannot go beyond defined rules, it cannot dynamically recognize new scenarios that could result in fraudulent transaction.

How do we overcome these limitations?

Organizations are beginning to utilize machine learning and data science to build fraud detection systems.

Given the size of available data, computational resources, and powerful machine learning algorithm available today, data science and machine learning processes will be able to find patterns in data and detect fraud easily.

goal

The goal of this Credit Card Fraud Detection project is to classify a transaction as valid or fraudulent in a large dataset.

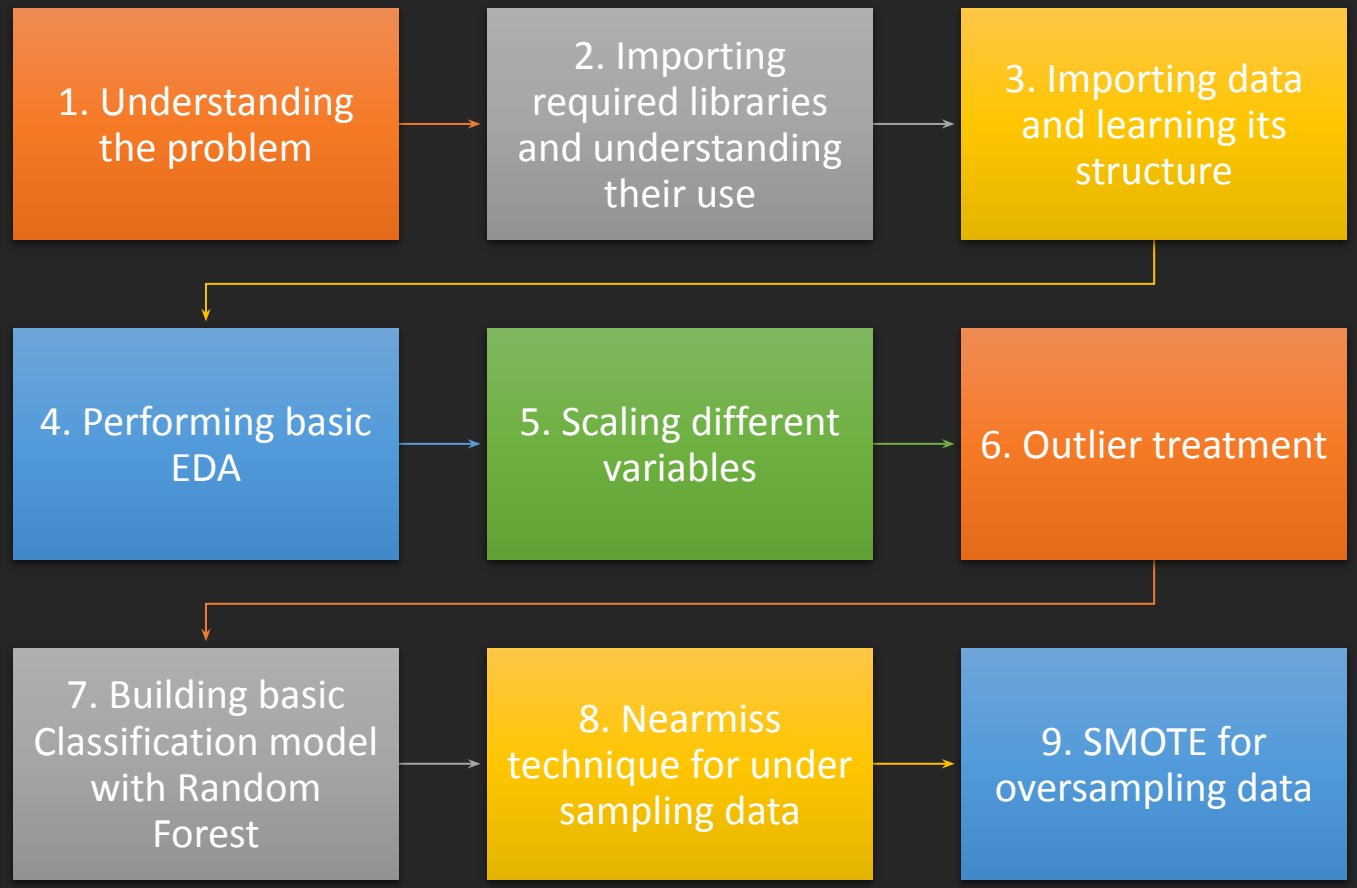
Since we are dealing with discrete values, this is a binary classification problem, and we would employ the use of a supervised machine learning algorithm.

Dataset used:

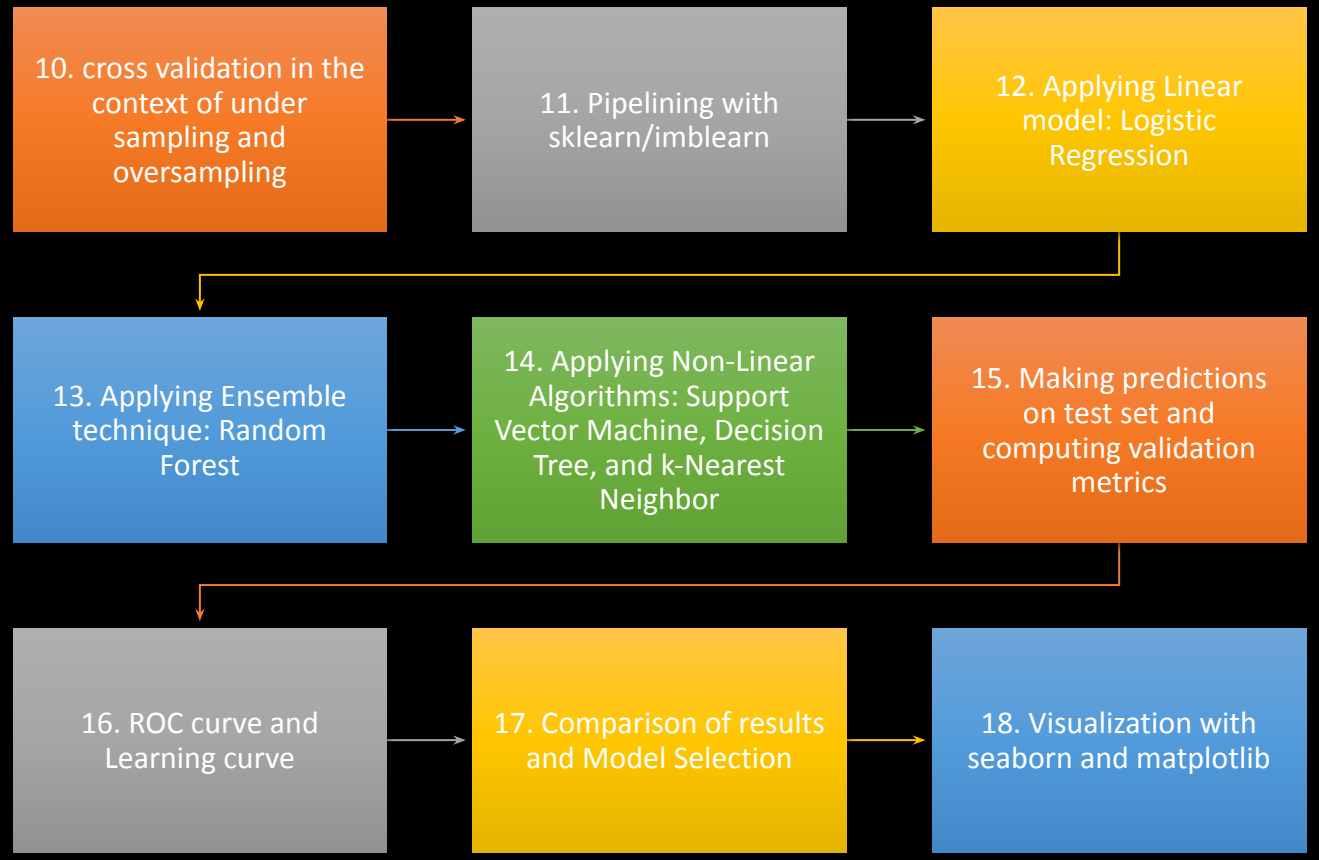
Transactions made by credit cards in September 2013 by European cardholders.

Transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions.

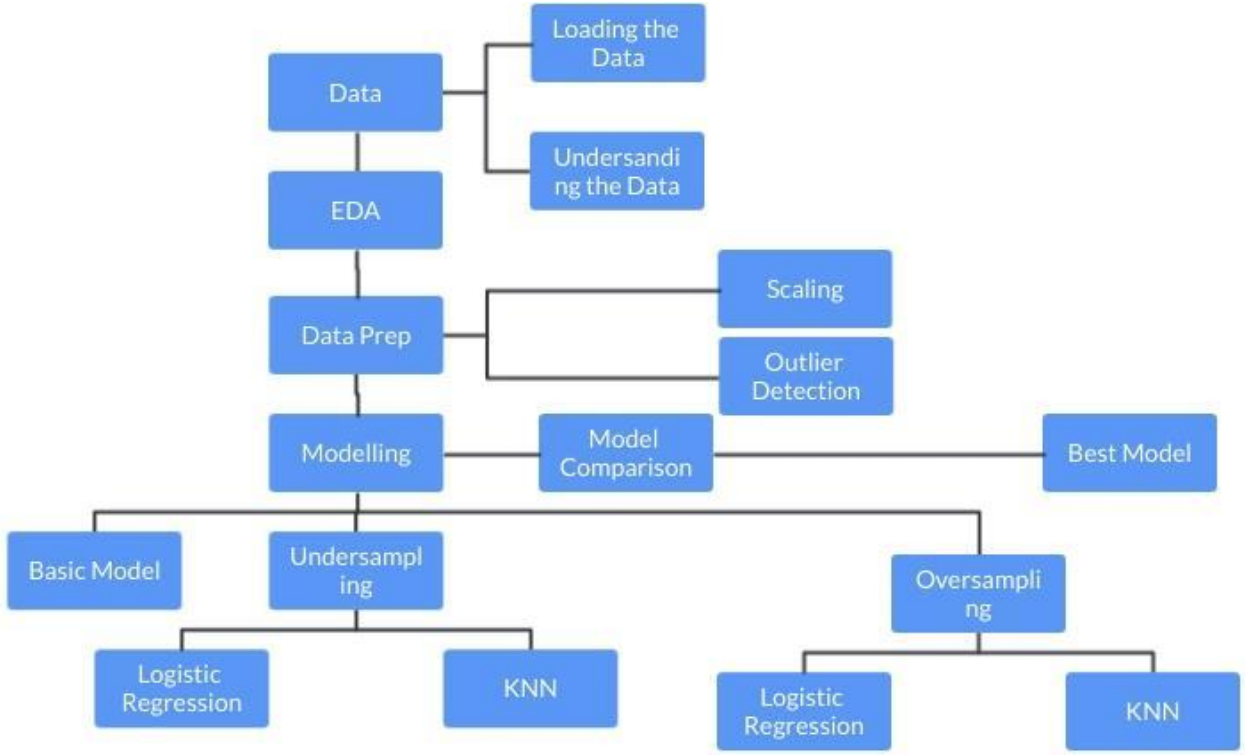
Control flow



Control flow (CONTINUED)



SOLUTION WORKFLOW




TECHNOLOGY

Logistic regression

Logistic regression is a classification algorithm used to find the probability of event success and event failure.

It is used when the dependent variable is binary (0/1, True/False, Yes/No) in nature.

It supports categorizing data into discrete classes by studying the relationship from a given set of labelled data.

It learns a linear relationship from the given dataset and then introduces a non-linearity in the form of the Sigmoid function.

Why logistic regression?

Logistic regression is easier to implement, interpret, and very efficient to train.

It makes no assumptions about distributions of classes in feature space.

It not only provides a measure of how appropriate a predictor (coefficient size) is, but also its direction of association (positive or negative).

Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.

RANDOM FOREST

Random forest is a technique used in modeling predictions and behavior analysis and is built on decision trees..

It contains many decision trees representing a distinct instance of the classification of data input into the random forest.

The random forest technique considers the instances individually, taking the one with most votes as the selected prediction.

It works well with both categorical and continuous values and automates missing values.

Why random forest?

- It reduces overfitting in decision trees and helps to improve the accuracy.
- It is flexible to both classification and regression problems.
- It works well with both categorical and continuous values.
- It automates missing values present in the data.
- Normalizing of data is not required as it uses a rule-based approach.

SVM

While SVMs do a good job recognizing speech, face, and images, they also do a good job at pattern recognition in categorical datasets.

SVM can be **used for the data that is not regularly distributed and have unknown distribution**.

SVM maps training examples to points in space to maximize the width of the gap between the two categories

New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

Why svm?

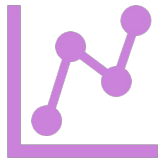
SVM works relatively well when there is a clear margin of separation between classes.

SVM is effective in high dimensional spaces.

SVM can be used for other types of machine learning problems, such as regression, outlier detection, and clustering.

SVM is very helpful method if we don’t have much idea about the data. It can be used for the data such as image, text, audio etc. It can be **used for** the data that is not regularly distributed and have unknown distribution

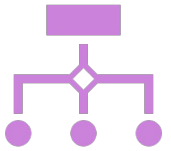
K-Means Clustering



Is a centroid-based algorithm, or a distance-based algorithm, where we calculate the distances to assign a point to a cluster.

It aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

The k-means algorithm searches for a pre-determined number of clusters within an unlabeled multidimensional dataset



The "cluster center" is the arithmetic mean of all the points belonging to the cluster, and each point is closer to its own cluster center than to other cluster centers.

Why k-means clustering?

It is easy to implement k-means and identify unknown groups of data from complex data sets. The results are presented in an easy and simple manner.

K-means algorithm can easily adjust to the changes. If there are any problems, adjusting the cluster segment will allow changes to easily occur on the algorithm.

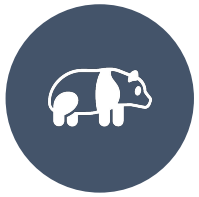
K-means is suitable for many datasets, and it’s computed much faster than the smaller dataset. It can also produce higher clusters.

The results are easy to interpret. It generates cluster descriptions in a form minimized to ease understanding of the data.

Compared to using other clustering methods, a k-means clustering technique is fast and efficient in terms of its computational cost

Database approach

Load raw dataset into AWS S3 bucket/PgAdmin.



Connect to AWS S3 bucket/PgAdmin and read data into Pandas.

Load the raw data into a PgAdmin Database Instance located in AWS.

Perform preprocessing steps and store cleaned data in a new table in AWS S3 bucket/PgAdmin.

Store some intermediate results (which can be used later for visualization) in AWS S3 bucket/PgAdmin.

The connection and S3 bucket details are in the Segment_One Jupyter Notebook.

A notebook that contains the code of the above steps is part of this repository.

Data cleaning and analysis

- This project will utilize Jupyter notebook and the panda's library to perform data cleaning and analysis.

Description of communication protocols

Communication for this project was through the Slack Group Chat.

Every team member will work in their individual branches.

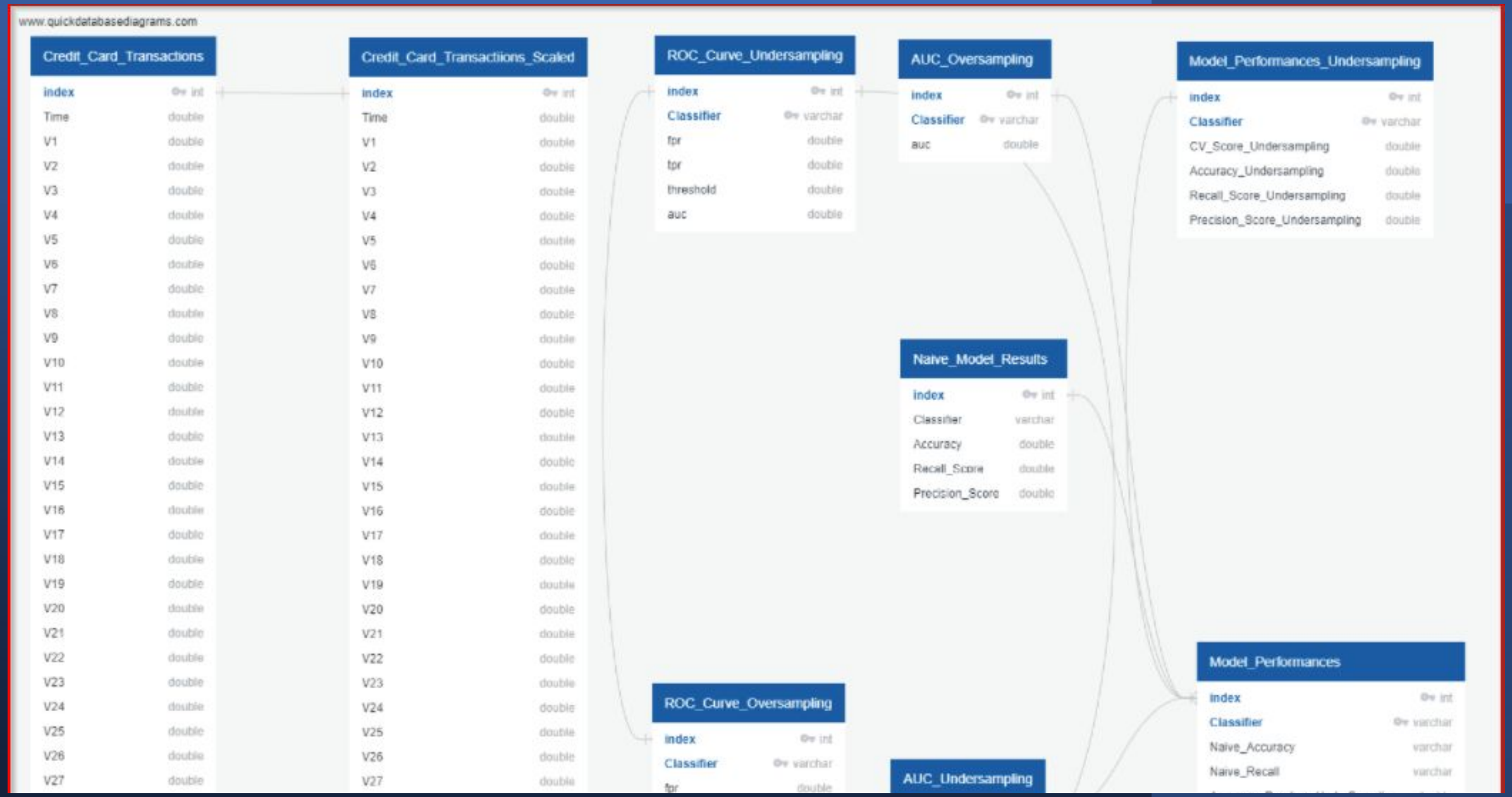
Team members will create pull requests which will be collectively approved in the slack channel.

A designated team member will validate the pull request and merge the request to the main branch.

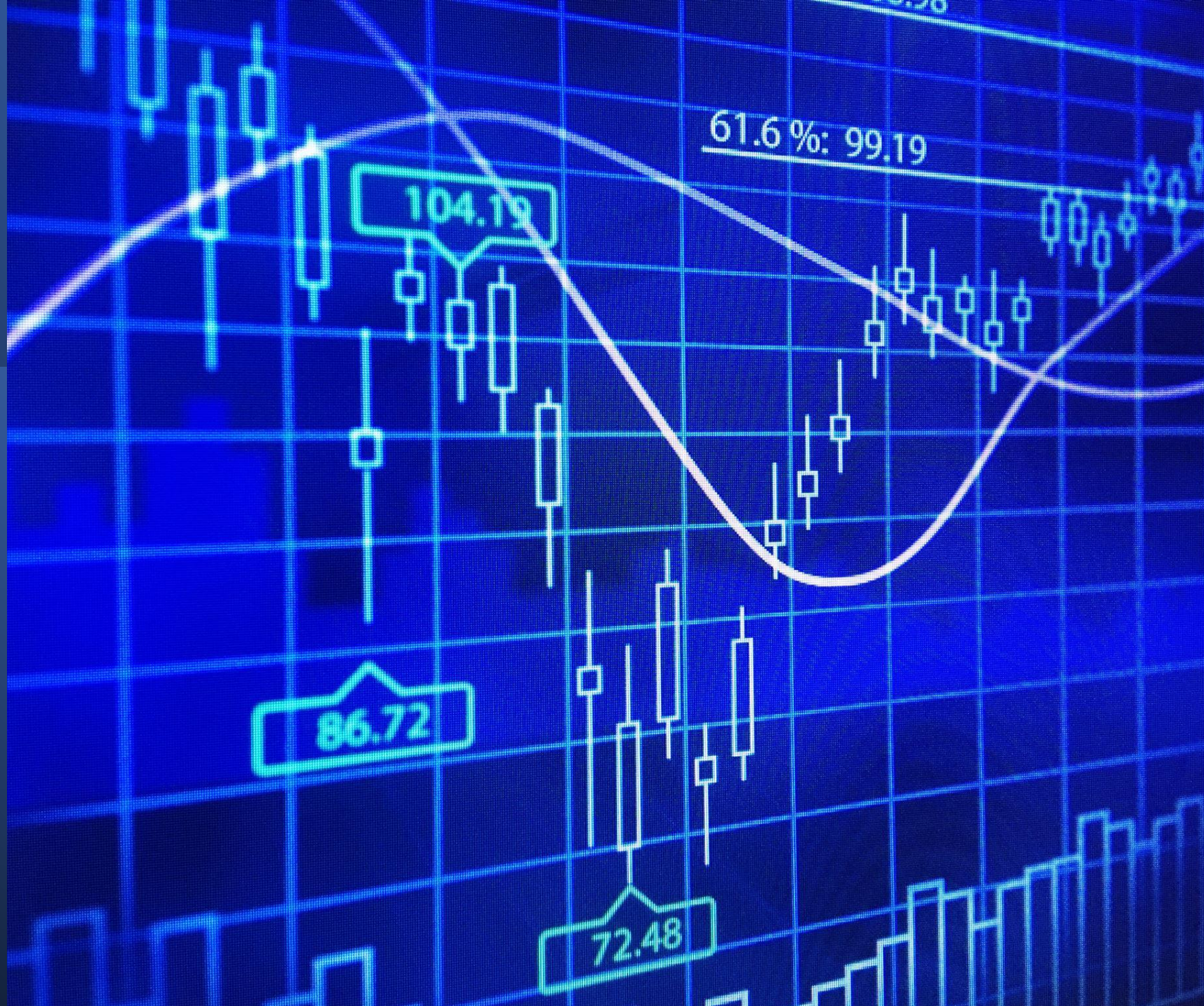
RESULTS

Database erd

- Database ERD shows all tables used to store intermediate exploratory data analysis results, and modelling results.


www.quickdatabasediagrams.com
Credit_Card_Transactions
index int
Time double
V1 double
V2 double
V3 double
V4 double
V5 double
V6 double
V7 double
V8 double
V9 double
V10 double
V11 double
V12 double
V13 double
V14 double
V15 double
V16 double
V17 double
V18 double
V19 double
V20 double
V21 double
V22 double
V23 double
V24 double
V25 double
V26 double
V27 double
Credit_Card_Transactiions_Scaled
index int
Time double
V1 double
V2 double
V3 double
V4 double
V5 double
V6 double
V7 double
V8 double
V9 double
V10 double
V11 double
V12 double
V13 double
V14 double
V15 double
V16 double
V17 double
V18 double
V19 double
V20 double
V21 double
V22 double
V23 double
V24 double
V25 double
V26 double
V27 double
ROC_Curve_Undersampling
index int
Classifier varchar
fpr double
tpr double
threshold double
auc double
AUC_Oversampling
index int
Classifier varchar
auc double
Model_Performances_Undersampling
index int
Classifier varchar
CV_Score_Undersampling double
Accuracy_Undersampling double
Recall_Score_Undersampling double
Precision_Score_Undersampling double
Naive_Model_Results
index int
Classifier varchar
Accuracy double
Recall_Score double
Precision_Score double
ROC_Curve_Oversampling
index int
Classifier varchar
fpr double
AUC_Undersampling
Model_Performances
index int
Classifier varchar
Naive_Accuracy varchar
Naive_Recall varchar

EXPLORATORY DATA ANALYSIS



Univariate analysis

Univariate plots show that the dataset is highly imbalanced.

The pie chart shows an imbalance in the data, with only 0.17% of the total cases being fraudulent.

The univariate distribution plot of the time and amount feature show we have a dataset with some large outlier values for amount.

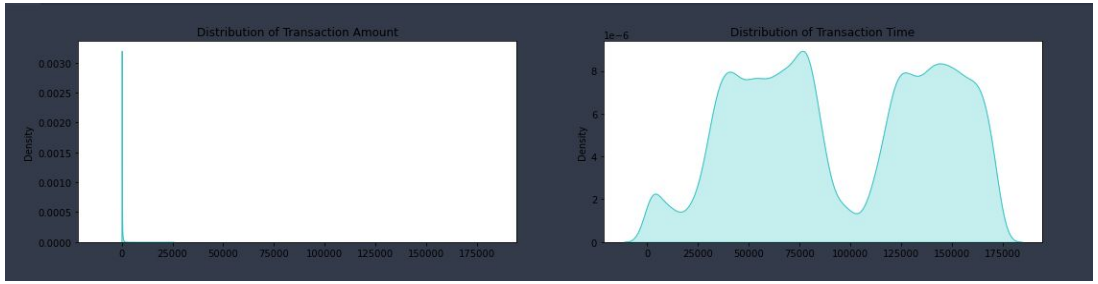
The time feature is distributed across two days.

Bivariate plots show that the valid transaction class has a normal distribution shape across most of the features.

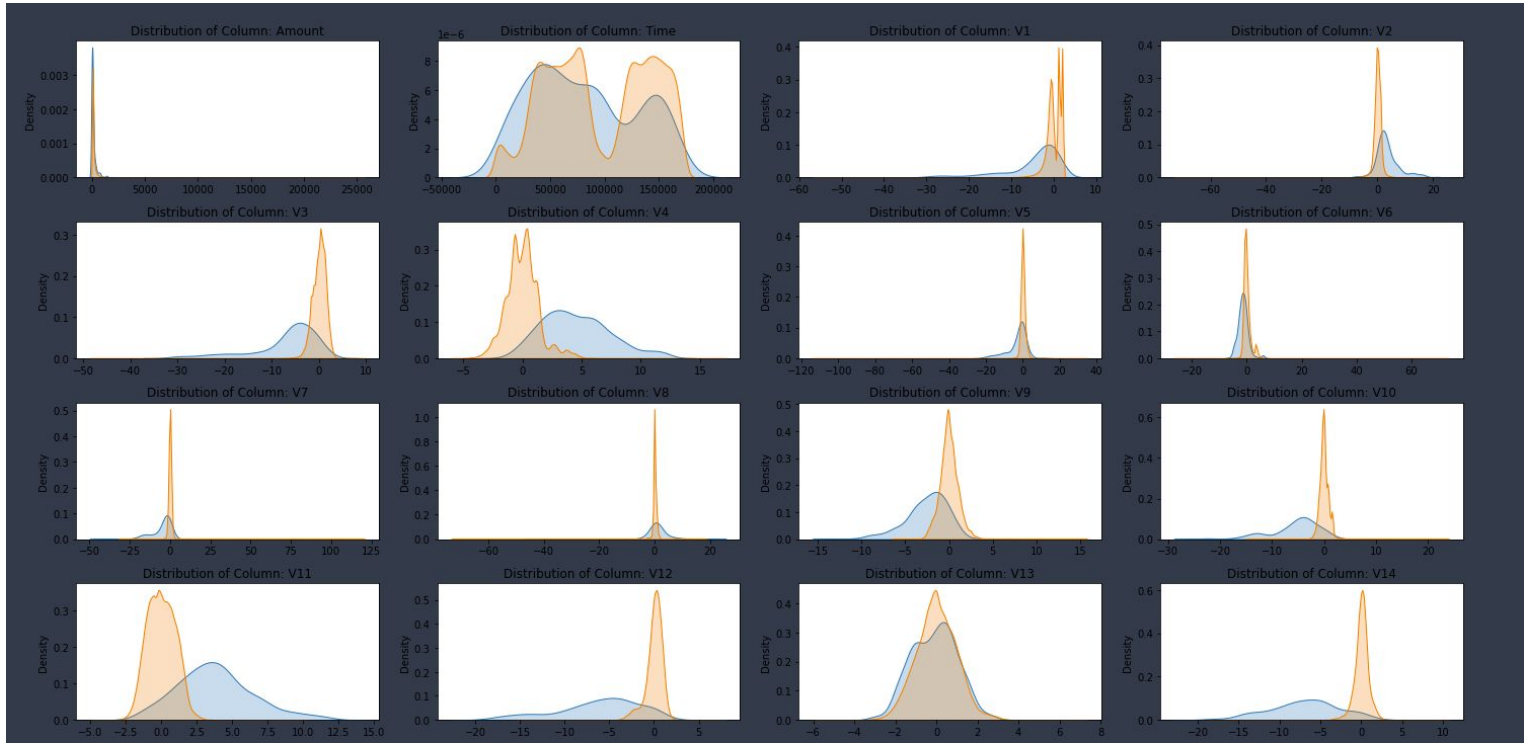
Conversely, the fraud class show long-tailed distribution across many of the features.

Univariate analysis

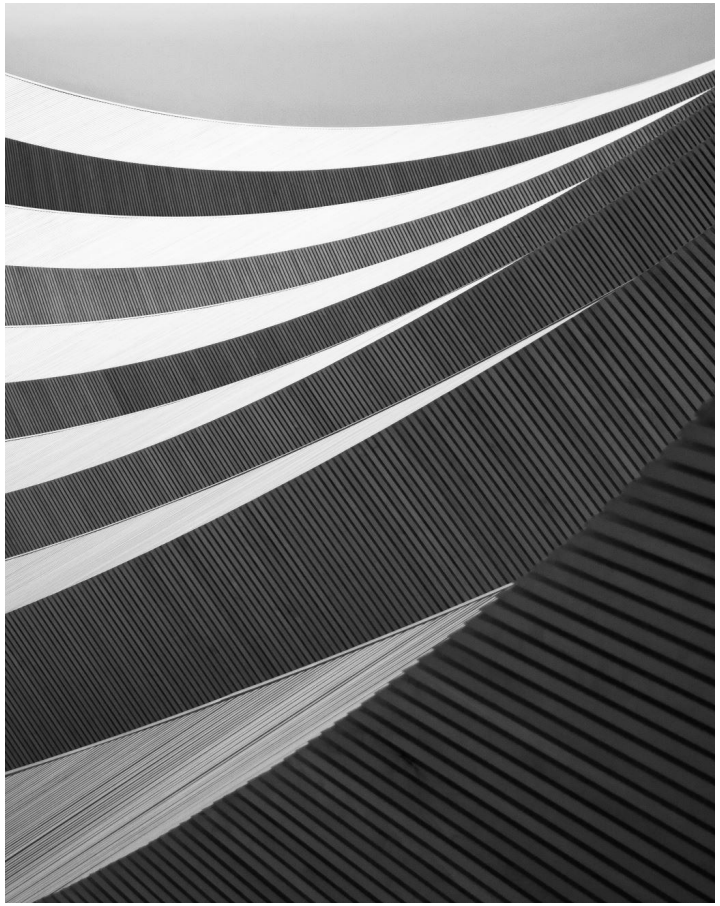
```
Fraudulent Transactions: 492
Valid Transactions: 284315
Proportion of Fraudulent Transactions: 0.001727485630620034

<AxesSubplot:ylabel=' '>
```



BIVARIATE ANALYSIS



Naïve model results



While the naive logistic classifier accuracy is 100%, our classifier did not do an excellent job at predicting fraudulent transactions.

With precision and recall of 0.84 and 0.62, we would need a better understanding of the dataset to determine the best way to improve the recall metric.

While the naive random forest classifier accuracy is 100%, and precision is 95%, our random forest classifier only achieved a 77% recall.

We would need a better understanding of the dataset to determine the best way to improve the recall metric.

	Classifier	Accuracy	Recall Score	Precision Score
0	Logistic Regression	0.999143	0.617886	0.844444
1	Random Forest	0.999551	0.772358	0.959596

Naïve model results

The roc-auc curve

ROC is a probability curve that plots True Positives and False Positives at different classification thresholds.

AUC - ROC curve is a performance measurement for a classifier at various classification thresholds.

Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.

Increasing the classification threshold classifies more items as negative, thus increasing both False Negatives and True Negatives.

The AUC lets us find the optimal classification threshold that minimizes False Positives and False Negatives.

For our credit card classification problem, we would want a classification threshold that increases True Positives.

AUC measures the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1).

AUC tells how much the model is capable of distinguishing between classes.

Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1.

The learning curve

The learning curve is the plot of the training/cross-validation error versus the sample size.

Learning curves show the relationship between training set size and the recall metric on the training and validation sets.

The learning curve detects whether the model has the high bias or high variance.

If the model suffers from high bias problem, as the sample size increases, training error will increase and the cross-validation error will decrease.

Training error and cross-validation error will end up close to each other but still at a high error rate.

If the model suffers from high variance, as the sample size increases, the training error will keep increasing and cross-validation error will keep decreasing.

Training error and cross-validation error will end up at a low training and cross-validation error rate.

Undersampling model results

By Undersampling the majority class in our dataset, all classifiers achieved recall scores greater than 85% except for the Support vector classifier.

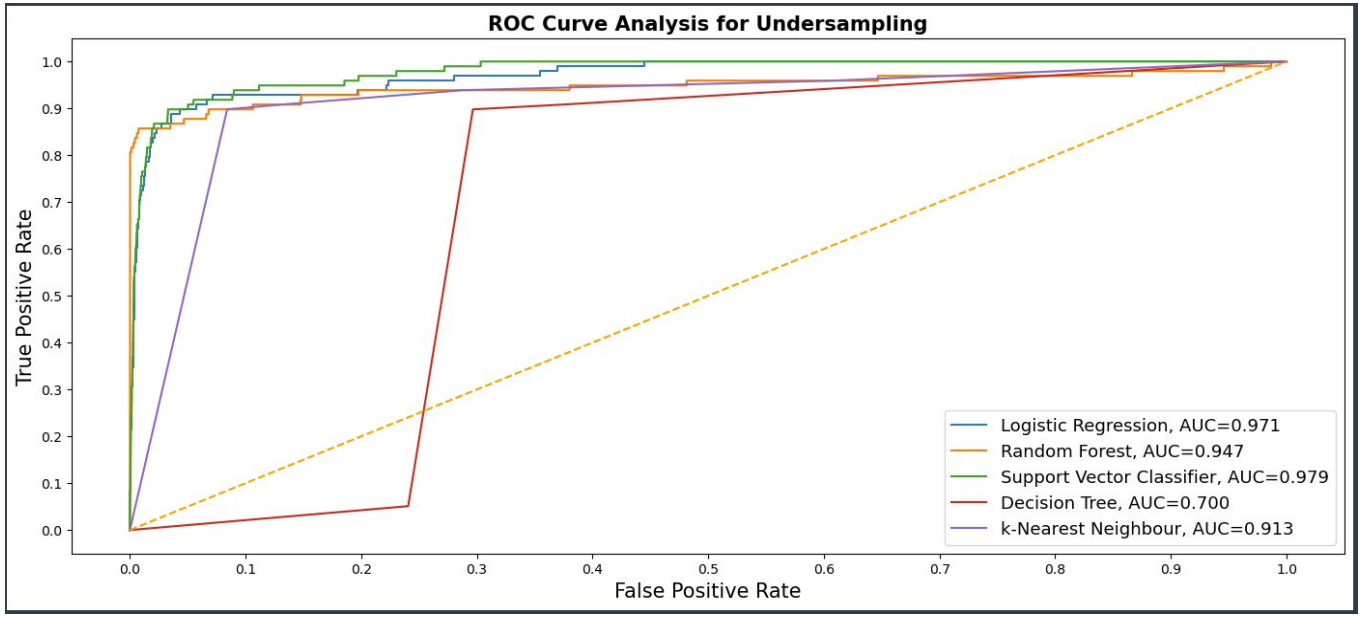
The ROC Curve show that the Support Vector Classifier has the largest AUC at 0.979, while the decision tree classifier has the smallest AUC at 0.700.

All undersampling curves show a fairly ideal learning curve.

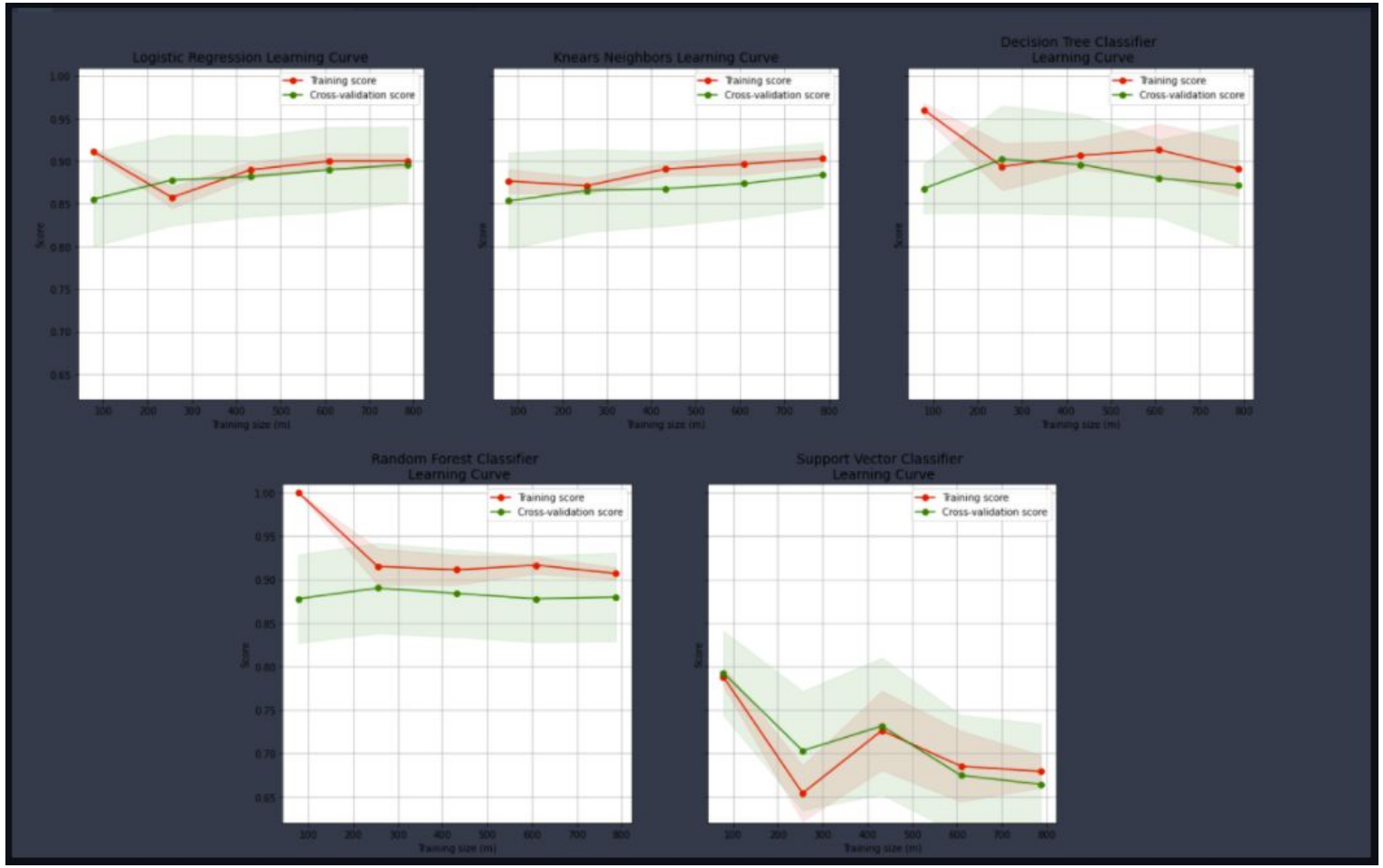
As the training size increases, training error and validation error generally reduces and end up at a low rate.

Undersampling results

	Classifier	CV Score - Undersampling	Accuracy - Undersampling	Recall Score - Undersampling	Precision Score - Undersampling
0	Logistic Regression	0.863031	0.957463	0.887755	0.034814
1	Random Forest	0.939241	0.505372	0.959184	0.003326
2	Support Vector	0.649984	0.991907	0.663265	0.131846
3	Decision Tree	0.903733	0.703750	0.897959	0.005191
4	k-Nearest Neighbour	0.903603	0.869141	0.908163	0.011813



UNDERSAMPLING RESULTS



Oversampling model results

By Oversampling the dataset, we achieved recall scores greater than 85% for all classifiers.

The Random Forest classifier had the best accuracy of 99%.

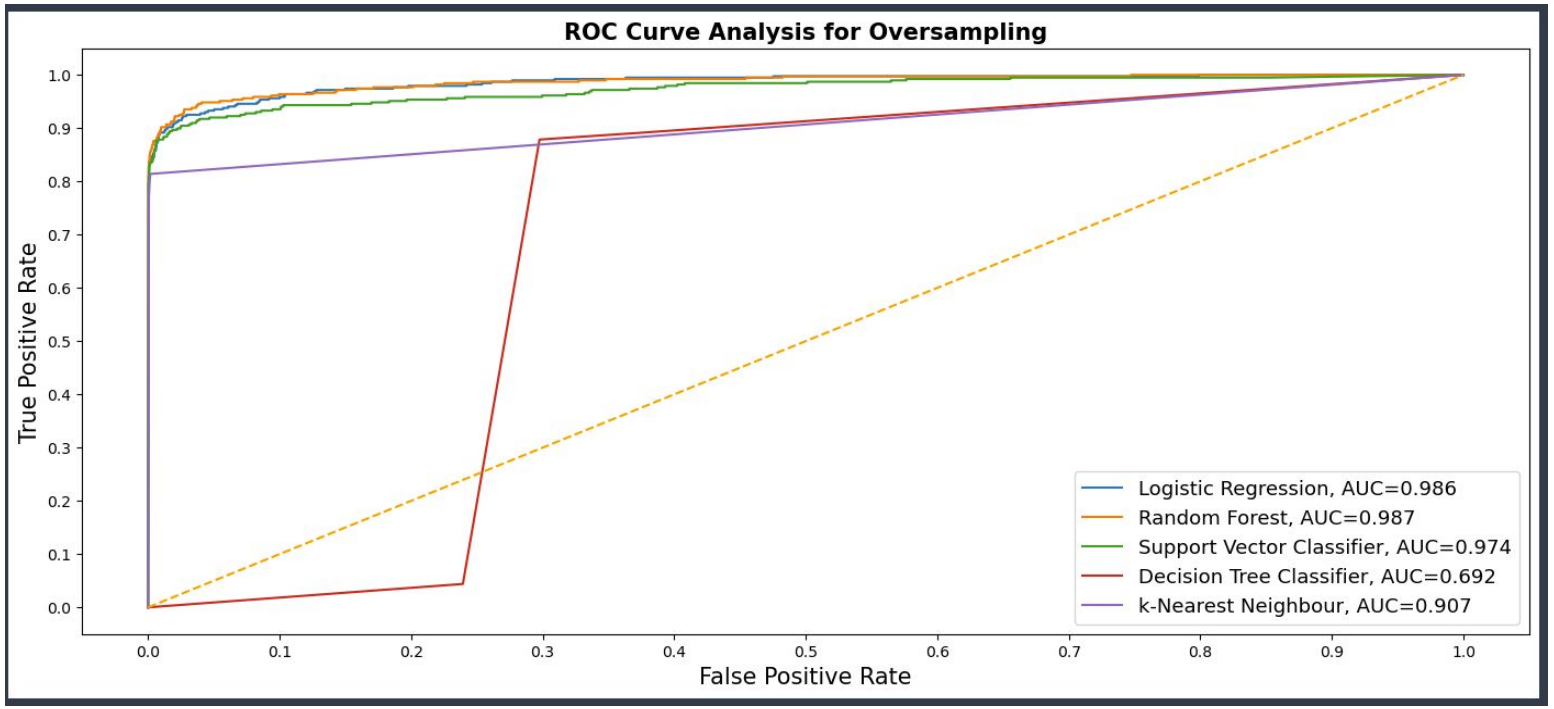
The ROC Curve show that the random forest classifier has the largest AUC at 0.987 while the decision tree classifier has the smallest AUC at 0.692.

All oversampling learning curves show a good fit.

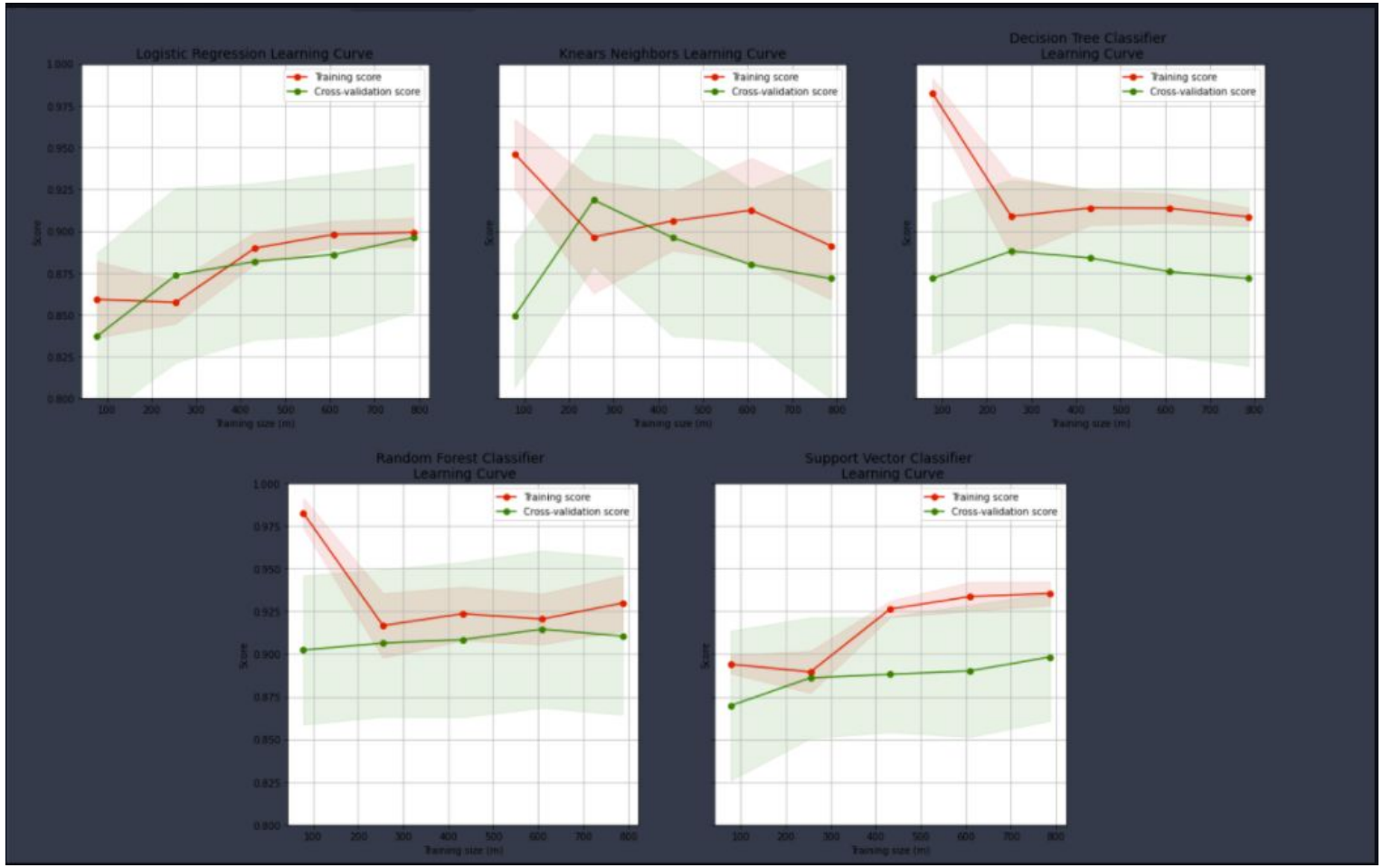
As the training size increases, training error and validation error generally reduces and end up at a low error rate.

COMPARING MODEL RESULTS

	Classifier	Naive - Accuracy	Naive - Recall	Accuracy - Random UnderSampling	Accuracy - Oversampling (SMOTE)	Recall - Random UnderSampling	Recall - Oversampling (SMOTE)
0	Logistic Regression	0.999143	0.617886	0.957463	0.975773	0.887755	0.918367
1	Random Forest	0.999537	0.772358	0.505372	0.995558	0.959184	0.887755
2	Support Vector	Not Applicable	Not Applicable	0.991907	0.977805	0.663265	0.899225
3	Decision Tree	Not Applicable	Not Applicable	0.703750	0.703750	0.897959	0.897959
4	k-Nearest Neighbour	Not Applicable	Not Applicable	0.869141	0.998508	0.908163	0.798450



OVERSAMPLING LEARNNG CURVE



SUMMARY

The dataset

The dataset used for this project has 284807 rows of credit card transactions.

Exploratory data analysis reveal as expected that we have a highly imbalanced dataset.

Only 0.17% of all transaction are fraudulent.

While a large portion of the features have been anonymized with PCA, univariate and bivariate distribution plots show that the genuine transaction class has an approximately normal distribution across all features, and the fraud class was had a left skewed distribution for many of the features.

Naïve models

While naive logistic regression and random forest had an accuracy of 100% and a precisions of 84% and 96% respectively, both classifiers only managed recall scores of 62% and 77% respectively.

This means that, the classifiers would miss fraud transaction almost 25% of the time.

This type of metric would cost an organization alot of money.

Performance metric

Classifying transactions as fraud or genuine is an anomaly detection problem where only a small fraction are the anomalies, measuring model performance with the accuracy metric will not be ideal.

To capture fraud transactions, we would require a classifier that has a high recall metric.

Recall is the ratio of of True Positives to the total of True Positives and False Positives

Oversampling, undersampling, roc, and learning curve

To improve the recall score of the naive models, we employ oversampling and undersampling.

With these methods, we achieved recall scores greater than 90%, and 85% for the undersampling and oversampling methods, respectively.

While recall for random forest was highest at 95.9%, the classifier had a lower AUC value (91.5) than the logistic regression classifier with AUC of 92.1.

Analysis of the learning curve show that the logistic regression had a generally good fit.

Best model

To choose the best model, we may consider the following factors:

Characteristics of the dataset

Since we are dealing with an imbalanced dataset, our first intuition is applying techniques such as undersampling and oversampling.

The Random Forest Classifier works well with resampling.

The Random Forest classifier lets us bootstrap samples, so we take a resample of our training data set.

And then we rebuild classification or regression trees on each of those bootstrap samples.

Evaluation metric: recall score

Since imbalanced datasets will generally have high accuracy scores, we need a different metric to evaluate model performance.

The choice of model may depend on the recall metric which measures the ratio of of True Positives to the total of True Positives and False Positives

Auc-roc curve

The learning curve helps us evaluate whether our model is overfitting, underfitting, or has a good fit.

The learning curve can be used in addition to the recall metric, and the AUC-ROC curve to select the best model for this classification problem.

Challenges and recommendations

One challenge with this project was computation resources required to run the RandomizedGridSearchCV and the model Cross-Validation scores.

One way to to mitigate this challenge in the future may be to use the HalvingGridSearchCV which may in some cases may be 30% faster than the RandomizedGridSearchCV.

We may also explore using an online environment that has unlimited computation resources that can handle the resource requirements for memory and CPU intensive models and processes.

Since feature extraction had been done on the dataset, visualizing potentially interesting relationships was not possible with this dataset.