PROJECT REPORT

TOPIC: Credit Card Fraud Detection

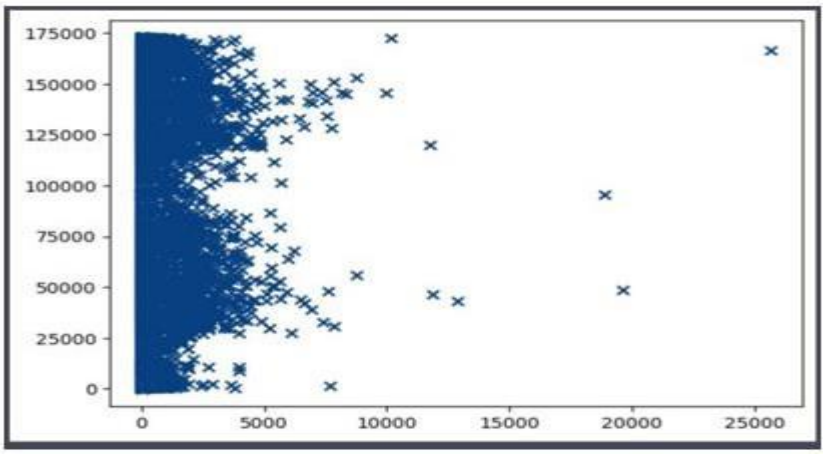
➢ PROBLEM STATEMENT:

Credit risk is associated with the possibility of a client failing to meet contractual obligations, such as mortgages, credit card debts, and other types of loans. The dataset given to us contains transactions made by credit cards in September 2013 by European cardholders. The dataset presents transactions that occurred in two days, with 492 frauds out of 284,807 transactions.
The dataset is highly unbalanced; the positive class (frauds) account for 0.172% of all transactions, making the dataset hard to handle. Given the details about the credit card, we have to predict whether the transaction that has occurred is Fraudulent or Non-Fraudulent.

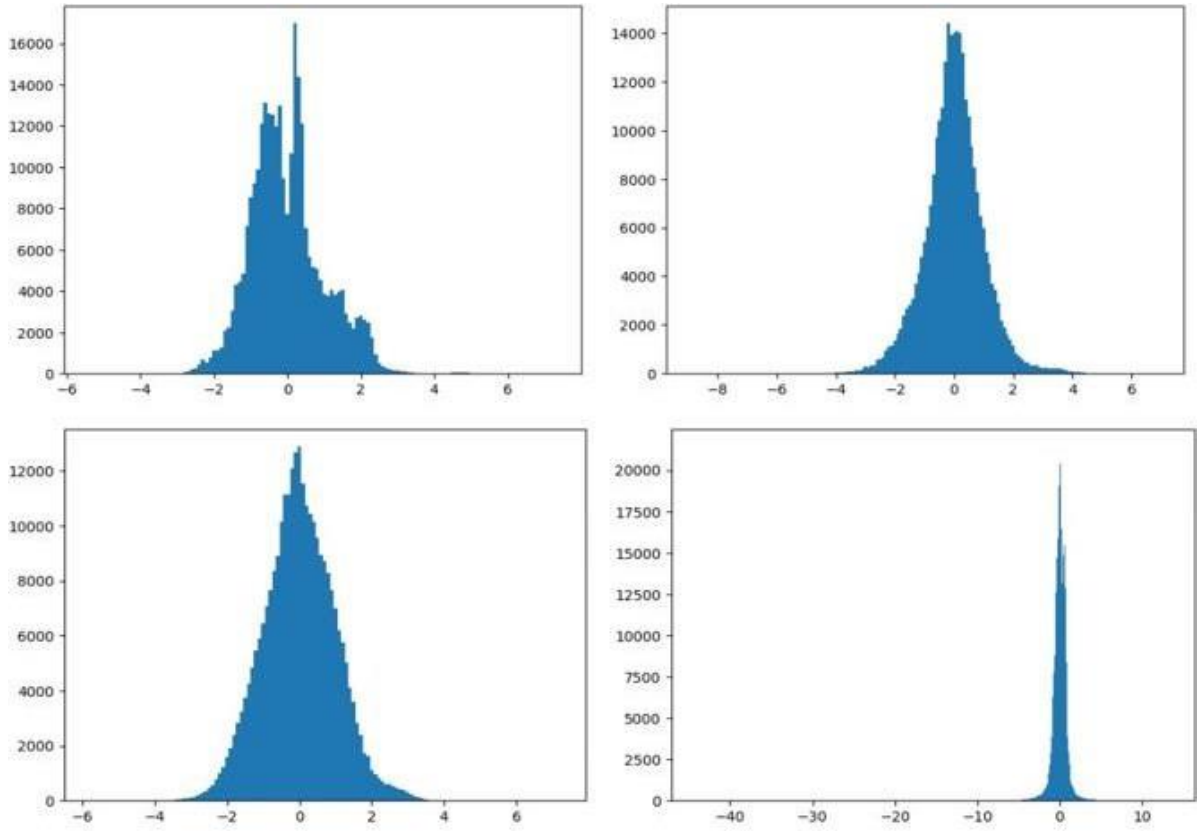
➢ DATA EXPLORATION:

We began the project by exploring the dataset's structure and distribution. We manipulated data using Python and pandas and data visualisation using Matplotlib and Seaborn.
The various steps involved in this process are

1. Checked for the data type of each column.
2. Checked for missing data and found that there were no missing values in the given dataset.
3. Visualized the distribution of the amount and time features and found that the data is highly skewed.

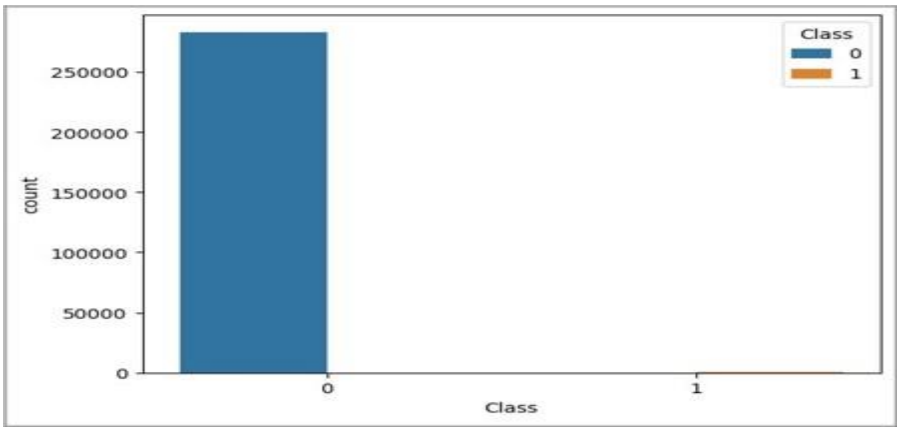


4. We used a scatter plot to examine how the hidden characteristics were spread out and found that they followed a normal distribution.

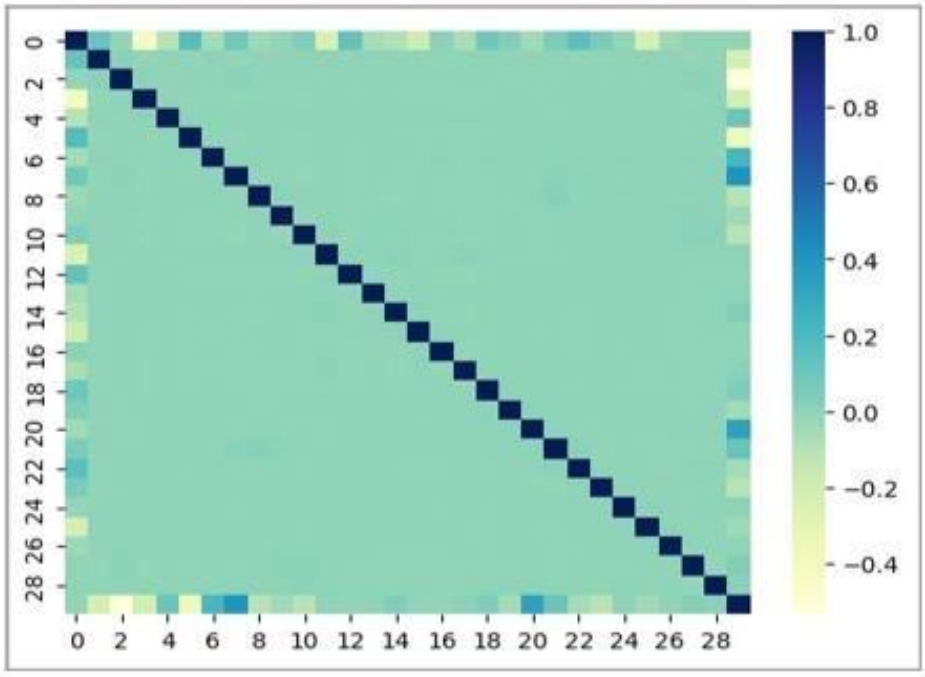


➢ DATA VISUALISATION:

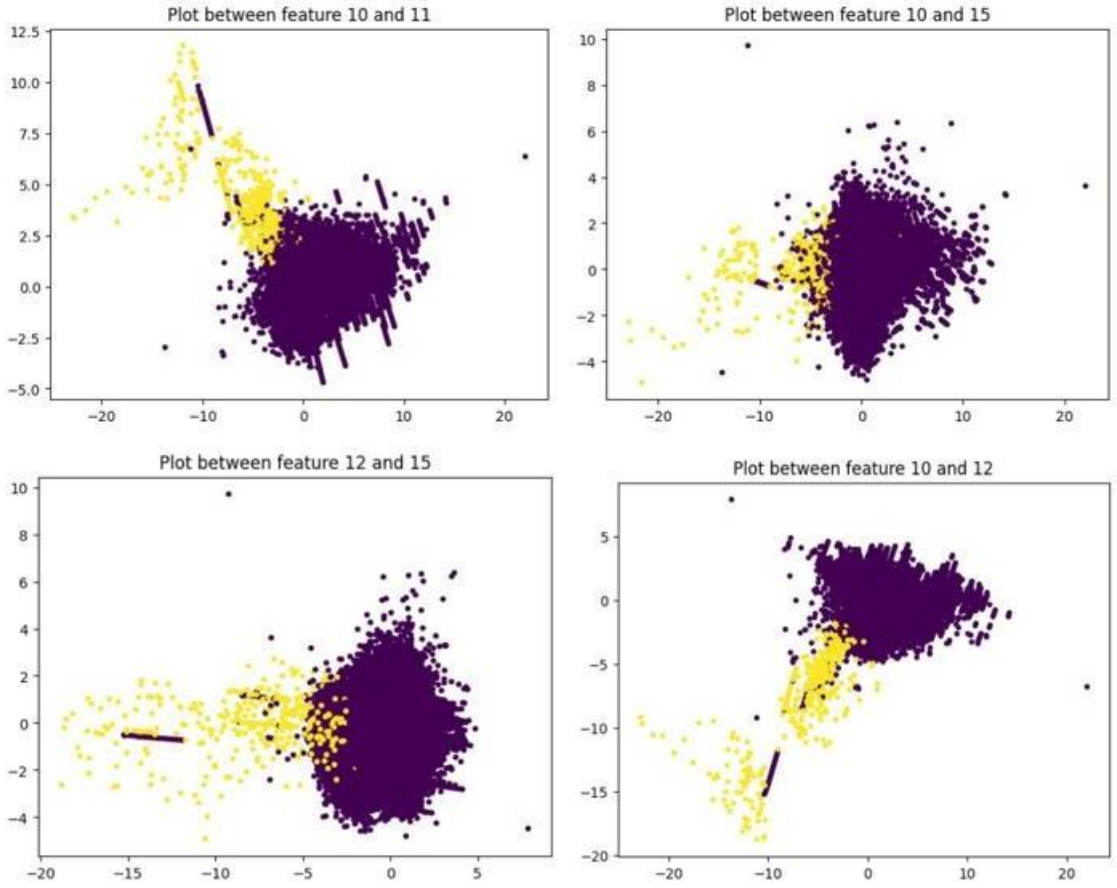
1. Plotted count chart using seaborn to count the number of classes labelled as zero and one. We found that the number of positive classes in the dataset was negligible in comparison to the negative classes.



2. Analyzed the correlation matrix created for all features.



3. Analyzed the scatter chart of all features against each other and displayed the ones that showed the maximum separability.



➢ MODEL OBSERVATIONS:

1. **Gaussian Naive Bayes**: It will perform well on the dataset as it was highly unbalanced. This can be because of its bias towards the majority class (negative) and its inability to capture differences accurately, leading to poor performance on minority classes.

2. **Logistic Regression and Decision Tree**: This will perform better than Gaussian Naive Bayes and could classify samples more accurately, leading to a better performance. In logistic regression, this happened due to its ability to assign greater weights to the minority class points and classify samples according to its logistic function, whereas in decision trees under a limited max_depth, it went through multiple splits making it predict the label more correctly.

3. **Bagging Classifier, Random Forest Classifier and K-Nearest Neighbours:** These algorithms did a very nice job in classifying the samples and were able to get higher values of recall and other metrics. In the Bagging Classifier, we used decision trees as estimators thus with bagging they made a better decision every time a sample was tested. Random Forest Classifier trained the estimators on different samples of the dataset along with a different set of features for each

estimator thus, it was able to make more accurate predictions. KNN classifiers worked well and from the patterns in the dataset, we can see that the maximum number of fraud cases and normal cases have their own neighbours, which made it easy to classify them.

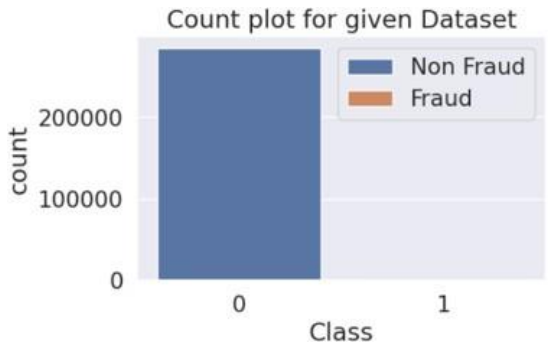
4. **Xtreme Gradient Boosting:** This classifier did the best job in classifying the samples as in an unbalanced dataset, mostly the minority class is suppressed, and thus using boosting (training the wrongly classified samples again with the next estimator) made the model work better.

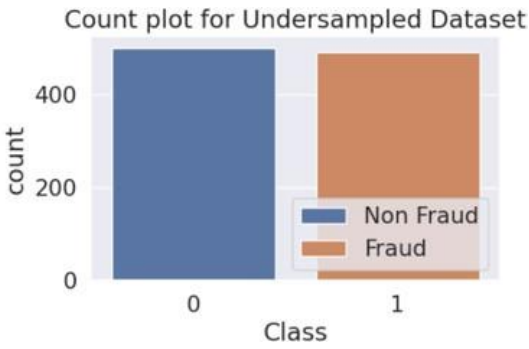
➢ OUR ATTEMPT:

Our Credit Card dataset consisted of **284315** Non-Fraud transactions and **492** Fraud transactions. As this is a highly unbalanced dataset, we applied Undersampling and Oversampling to get a balanced dataset.

● Undersampling

In this we randomly sampled **500 Non-Fraud transactions** out of the total dataset and then joined them with the **492 Fraud transactions** to get a dataset that was well-balanced with both the classes in almost equal proportions.





● Oversampling

In this we randomly sampled **4000 Non-Fraud transactions** out of the total dataset and then oversampled the already existing **492 Fraud transactions** to get a dataset that was well balanced with both the classes in almost equal proportions. We chose 4000 as the appropriate number because, as we are oversampling, the Fraud transactions would have been repeated in the new dataset around 8 times, giving us a good chance at including more fraud transactions while training while being computationally less expensive. Whereas if we had used a number greater

than 4000, we would have greater repetitions, thus creating a possibility of excessive repetition and overfitting and also, it would have been more computationally expensive.

- **Applying Classifiers on Undersampled and Oversampled Datasets**

We tried mainly three different classifiers for the task of classifying, which were **Decision Tree** and **Random Forest** and **K-Nearest Neighbours**. Based on the training and testing accuracy achieved and also the recall precision of the classifiers, especially for the Fraud class (class 1), Random Forest was the best choice to proceed with.

For fraud detection, Recall is given high importance as in fraud detection the cost of misclassifying a fraudulent transaction as a non-fraudulent transaction can be proven to be very costly. Here, as for Decision Tree and Random Forest, as the recall score for Fraud class (class 1) is almost the same, we then proceed to look at the precision scores(which are higher for RFC than in DT, thus indicating that are less non-fraud transactions being classified as fraudulent, hence being more precise) and the testing accuracies, which helped us in choosing Random Forest Classifier for further classification purposes.

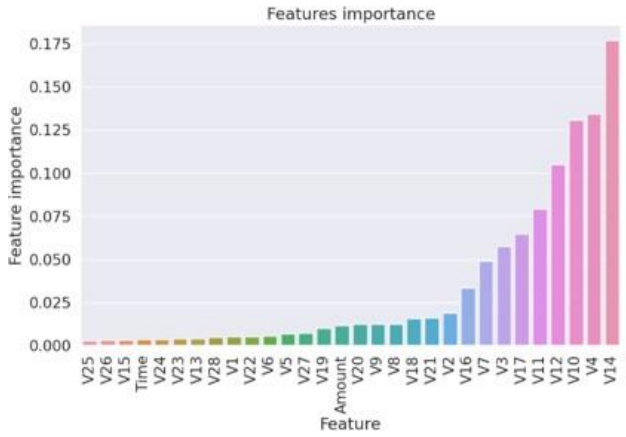
Model (on Undersampled data)	Training Accuracy	Testing Accuracy	Precision (class 0)	Precision (class 1)	Recall (class 0)	Recall (class 1)
Decision Tree	96.72	93.46	0.91	0.97	0.97	0.89
Random Forest	96.84	95.47	0.93	0.99	0.99	0.91
KNN	66.07	60.8				

Model (on Oversampled data)	Training Accuracy	Testing Accuracy	Precision (class 0)	Precision (class 1)	Recall (class 0)	Recall (class 1)
Decision Tree	96.5	95.5	0.94	0.97	0.97	**0.94**
Random Forest	97.07	96.25	0.93	0.99	0.99	**0.93**
KNN	75.68	70.75				

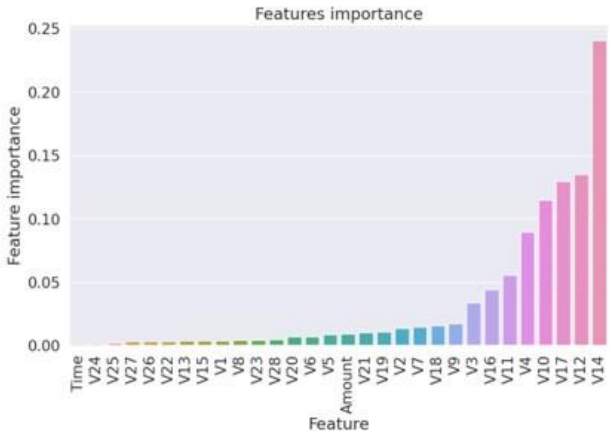
From this point forward the steps taken are identical for both Undersampled and Oversampled Datasets

As we are dealing with 30 features in the dataset, there are bound to be features that do not provide much useful information to the classifier. Hence, from the Random Forest model trained, we got the amount of importance for all the features of the dataset and then selected the ones that were the most important and formed an updated dataset with only these features.

Undersampled Dataset



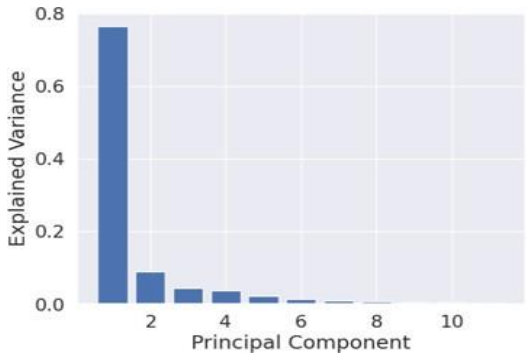
Oversampled Dataset



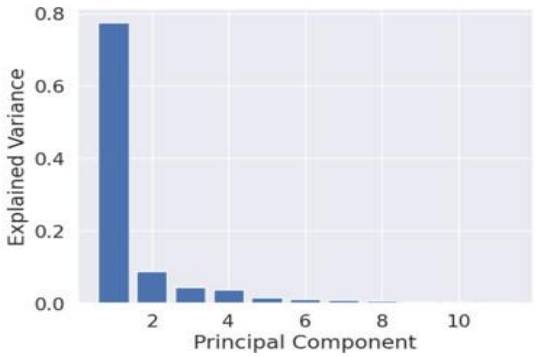
- **<u>Applying PCA</u>**

In the updated dataset, we can further apply PCA to reduce the dimensionality and use the new features that conserve the most amount of variance of the dataset.

Undersampled Dataset



Oversampled Dataset

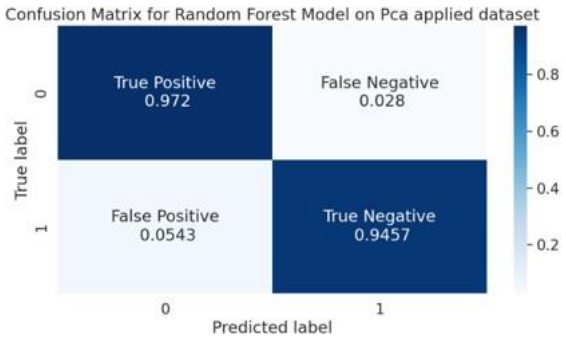


In this case we can clearly see that most of the variance conserved is amongst the first 4 features. Hence, we transform this dataset to get a new dataset with n_components = 4 while applying

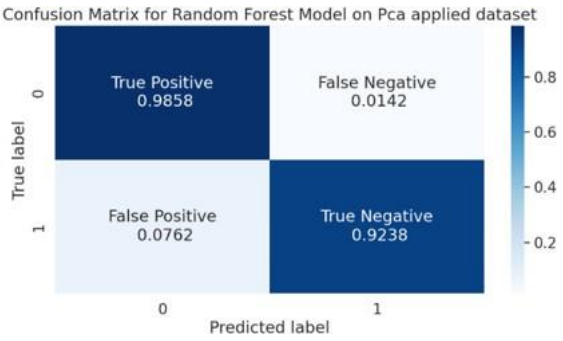
PCA. On this transformed dataset with 4 features, we then applied Random Forest Classifier and KNN and received the following results.

- **<u>Random Forest on PCA applied Dataset</u>**

Undersampled Dataset



Oversampled Dataset



Undersampled Dataset

Accuracy	Original Dataset	PCA Applied Dataset
Training Dataset	96.84	96.34
Test Dataset	95.47	94.47

Oversampled Dataset

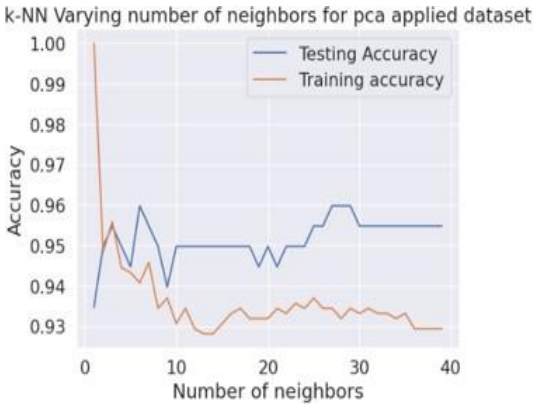
Accuracy	Original Dataset	PCA Applied Dataset
Training Dataset	97.07	95.73
Test Dataset	96.25	95.0

The accuracy of the model on PCA applied dataset is quite close to the accuracy on original data therefore the new model is better than previous one as it takes less time and resources to give similar results.

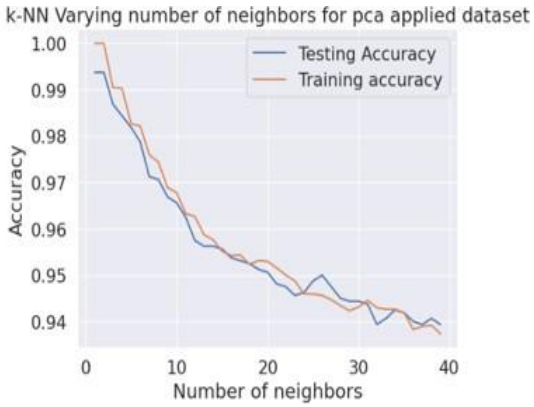
- **<u>KNN on PCA applied Dataset</u>**

For the KNN classifier, first, we find the training and testing accuracy of the model on the set of k values to find the most suitable value of k at which training and testing are nearly identical.

Undersampled Dataset



Oversampled Dataset



Undersampled Dataset

Accuracy	Original Dataset	PCA Applied Dataset
Training Dataset	66.07	94.82
Test Dataset	60.8	94.47

Oversampled Dataset

Accuracy	Original Dataset	PCA Applied Dataset
Training Dataset	75.68	100
Test Dataset	70.75	99.0

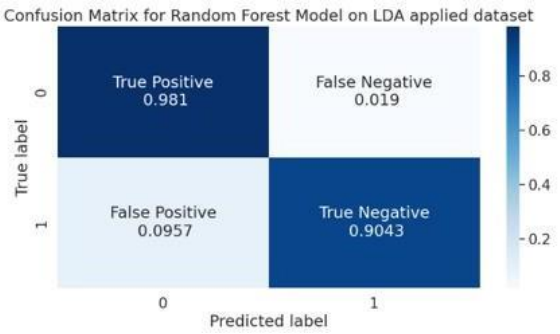
From this we conclude that KNN does not give better results when the number of features is large but when we reduce the number of features in the dataset accuracy of the KNN model increases significantly.

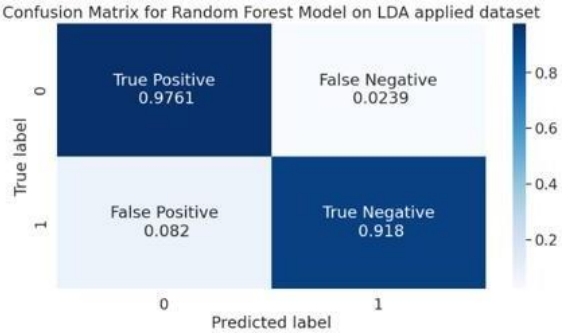
- **Applying LDA**

In the updated dataset that we got from using the important features of the original dataset, we can also apply LDA as LDA is used to maximize the separability between the classes present in the dataset. Hence, we transform this dataset to get a new dataset by applying LDA. On this transformed dataset, we then applied Random Forest Classifier and KNN and received the following results.

- **Random Forest on LDA applied Dataset**

Undersampled Dataset **Oversampled Dataset**





Undersampled Dataset

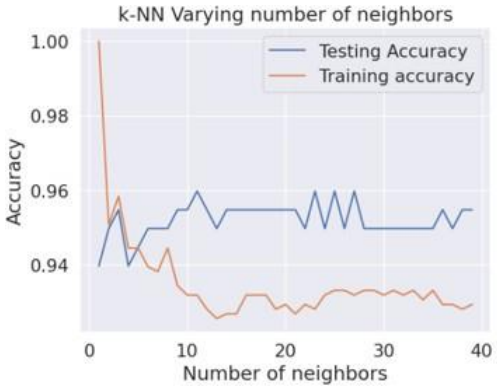
Accuracy	Original Dataset	PCA Applied Dataset
Training Dataset	96.84	93.82
Test Dataset	95.47	94.47

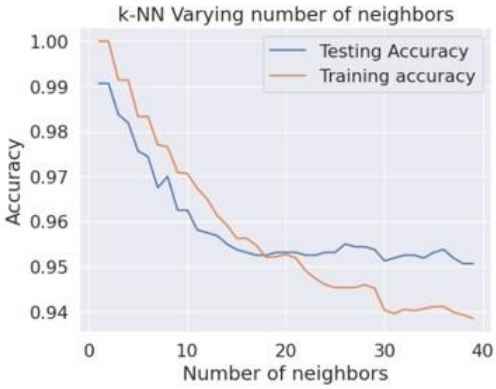
Oversampled Dataset

Accuracy	Original Dataset	PCA Applied Dataset
Training Dataset	97.07	94.28
Test Dataset	96.25	93.0

- **KNN on LDA Applied Dataset**

Undersampled Dataset **Oversampled Dataset**





Undersampled Dataset

Accuracy	Original Dataset	PCA Applied Dataset
Training Dataset	66.07	92.18
Test Dataset	60.8	95.57

Oversampled Dataset

Accuracy	Original Dataset	PCA Applied Dataset
Training Dataset	75.68	**94.98**
Test Dataset	70.75	**93.43**

Compiled Accuracy Table

Model	Under sampled Dataset		Oversampled Dataset	
	Training	**Testing**	**Training**	**Testing**
Random Forest	**96.84**	**95.47**	**97.07**	**96.25**
Decision Tree	**96.72**	**93.46**	**96.5**	**95.5**
KNN	**66.07**	**60.8**	**75.68**	**70.75**
PCA Applied Dataset				
Random Forest	**96.34**	**94.47**	**95.73**	**95.0**
KNN	**94.82**	**94.47**	**100**	**99.0**
LDA Applied Dataset				
Random Forest	**93.82**	**94.47**	**94.28**	**93.0**
KNN	**92.18**	**95.57**	**94.98**	**93.43**

➢ OBSERVATIONS AND CONCLUSIONS:

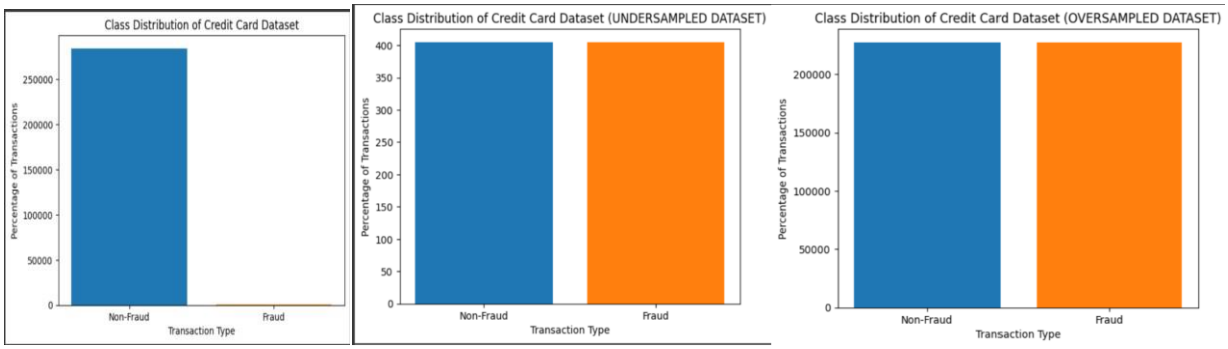
1. We observed that the KNN classifier improved its performance drastically after selecting important features and the new dataset containing only these important features was used. This could be because earlier features that were not so important were also present in the dataset; they added noise, which may have led to poor classification. Further PCA and LDA being applied to this important feature dataset further refined the features by conserving as much variance as possible and ensuring maximum separability amongst the classes respectively, hence, improving accuracy.

2. After training different types of models we find that Random Forest Classifier gives us the best overall result when compared to KNN and that too in an Oversampled Dataset. An oversampled dataset gives us better accuracy in comparison to an under-sampled dataset because, in an under-sampled dataset, we have only 1000 samples in which there are only 492 cases of fraud. In contrast, in the oversampled dataset, we have 4,000 fraud cases, so the model gets trained on many more instances of fraud transactions thus the model can identify the fraud cases in the test dataset with a higher accuracy.

3. Random Forest is better in comparison to decision tree in this case because in our dataset with only the important features, as all our features are important, we can't decide on any one feature, whereas random forest incorporates all features while training by training weak classifiers in the form of Decision Trees and then using them together to become a strong classifier, therefore it gives us better results.

SMOTE (Synthetic Minority Oversampling Technique)

SMOTE is a technique that involves the generation of random samples for the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling. The working of SMOTE involves, the selection of a random minority class instance says 'a', and then using the KNN algorithm to find its K nearest neighbors then synthetic instances are created by choosing one of the K neighbors and connecting 'a' and 'b' to form a line segment in the feature space. And then drawing random data points in between this line and thus generating a synthetic dataset.



We performed a comparative analysis study of different methods used for training of models for imbalanced datasets. We can conclude from the study that almost all different kinds of machine learning classifiers like Ensemble-KNN and Cost sensitive Adaboost and Cost sensitive ANN

provide quite promising results over the provided dataset. **Ensemble-KNN method** is giving **best results** for the recall and f1-score since more weights are given to classifiers with considerably high g-mean and f1-score. So, finally the predictions based on the majority vote are more influenced by models predicting the minority class better. Cost-sensitive Neural network deals with the imbalanced dataset on algorithmic level. The CS-ANN model is the next best classifier for the imbalanced dataset. In addition, the Cost sensitive ADABoost also performs well on the dataset in which we have tuned four parameters intuitively.

RESULTS BEFORE AND AFTER SMOTING:

	Precision_1	Recall_1	f1-score_1	Precision_0	Recall_0	f1-score_0
ensemble-KNN	0.787234	0.961039	0.865497	0.999947	0.999648	0.999798
Undersampled_Bagging	0.905983	0.042967	0.082043	0.958466	0.999798	0.978696
Oversampled_Bagging	0.726496	0.833333	0.776256	0.999701	0.999437	0.999569
SMOTE_Bagging	0.743590	0.737288	0.740426	0.999455	0.999472	0.999463
Combined_oversampling_undersampling_Bagging	0.743590	0.756522	0.750000	0.999507	0.999472	0.999490
Logistic_imbalanced	0.547009	0.927536	0.688172	0.999912	0.999068	0.999490
Logistic_Random_Undersampled	0.905983	0.046923	0.089226	0.962125	0.999799	0.980600
Logistic_Oversampled	0.897436	0.076253	0.140562	0.977623	0.999784	0.988580
Logistic_combined	0.897436	0.089438	0.162665	0.981194	0.999785	0.990402
GNB	0.837607	0.079160	0.144649	0.979945	0.999659	0.989704
GNB_Random_Undersampled	0.037010	0.863248	0.070977	0.999705	0.953769	0.976197
GNB_Oversampled	0.069815	0.871795	0.129278	0.999730	0.976093	0.987770
CS_ADABoost	0.709091	0.829787	0.764706	0.999719	0.999719	0.999719
CS_ANN	0.797872	0.892857	0.842697	0.999842	0.999666	0.999754

Precision, Recall, F1-Score values for classifiers trained on the credit card dataset.

From the above table, we can see that Ensemble-KNN, Bagging Classifier on Oversampled data and Cost-sensitive ANN are giving high F1-score when trained over the imbalance credit card dataset.

Recall for the minority class is the most important factor for analysing the prediction of the classifiers. F1-Score for the minority class(f1-score_1) is the next important since we are focusing on the correct prediction of the minority (fraud transaction) class.

● The recall for the Gaussian Naive Bayes classifier on under-sampled data is 95% which is significantly lower than the any other classifier. The loss of information due to under-sampling results in the decrease of recall for the majority class.

● In the Logistic Regression Classifier on under-sampled data, the majority class data is reduced and hence the false positive (the number of negative samples classified as positive) gets reduced. As a result, precision for the minority class (Precision_1) improves.

● Ensemble KNN gives exceptionally high recall and comparatively better F1-Score for the minority class.

● Cost sensitive ANN is performing well on the imbalanced dataset and giving a good recall and F1-Score.

● SMOTE is giving better results than under-sampling and oversampling because it does not duplicate minority class like oversampling. SMOTE makes new points based on KNN and hence, provides average precision and recall for classifier models.

➢ LEARNINGS:

This project gave us a very good opportunity to learn how to handle extremely skewed and imbalanced data. This helped us learn various techniques to handle such types of data, such as Oversampling, Under-sampling and SMOTING. We also learned the importance of using only the desired features while training classifiers. We learnt that using all the features may not always be beneficial and how selecting only the important features and using other feature reduction and selection techniques such as PCA and LDA has a profound impact on the accuracy of the classifier.