Credit Card Fraud Detection using Deep and MachineLearning

Group 9

Group Members :

Nusrath Miya Shaik - 11763609
Aishwarya Linga - 11700535
Anuhya Gujjarlapudi - 11699842
Priyanka Bhadra - 11724168
Sushma Sri Alasyam - 11642736
Ranabothu Siri - 11700483

Abstract:

Credit cards remain a trend that worrying the lot of people that continuously results in massive losses to all the credit cardholders and also in other various institutions. The work which we did is the identification of fraud transactions by integrating the Deep learning algorithms and Machine algorithms for these fraud issues and which will include the class imbalance and the latest fraud schemes and false positives . Initially for detecting this fraud credit transactions which will use the traditional machine learning algorithms like Random Forest , SVM , Decision Trees . But for better performances we are implementing the Deep Learning Techniques and neural network algorithms like LSTM and for the data imbalance methods like we are using the Undersampling methods were implemented and when come to the performance of the algorithms it will additionally be boosted by the hidden layers and epochs

Introduction:

One type of known theft is credit card fraud (CCF), which entails using another person's credit card, account number, card number, or data to conduct fraudulent transactions. These days, card-not-present fraud has escalated and CCF has emerged as one of the major concerns for both individuals and businesses. As a result of the extensive growth of the e-commerce industry, online transactions, and e-banking, credit card fraud occurs every year. More than $246 million incidents of CCF were reported in 2023 alone; however, this number climbed progressively in comparison to previous years, and the financial damages linked to the scam also increased.

To tackle this problem the companies and the financial institutions are developing their own automated fraud detection system. And the main goal of credit card fraud detection is to create a machine learning model based on the existing transactions card data and the main aim of this model is to distinguish between the fraudulent and non fraudulent transactions with this type of detection system so they can know   whether the incoming transaction is fake or not . To implement this kind of the system technically the detection system has to overcome the quick response time of the system and the sensitivity and main was the feature pre-processing .

With the use of Machine learning and Deep learning algorithms have been implemented to solve the issues in the fields of banking , insurance and the health and surveillance and the detection. In the project we used machine learning algorithms and deep algorithms where we had to investigate the application to credit card fraud in the banking industry. The traditional machine learning algorithms like SVM which are tenable for the binary classification and which will be commonly used for the image recognition and the banking. Deep Learning methods were implemented for the large datasets and the neural networks were used for the fraud detction.

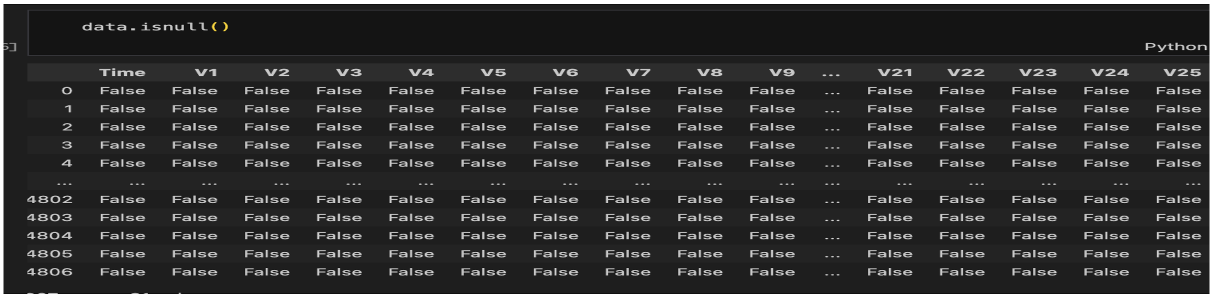
Deep Learning researchers are increasing their interests while using these CCF detection systems , the limited research has explored the use of the deep learning neural networks for the task . In this project we are using LSTM long short term memory model to distingush between the fraud and non fraud transactions . The class imbalance problem, which has an order magnitude of fewer fraudulent transactions compared to non fraud transactions, is addressed in the LSTM model with the additional layers to improve feature extraction and classification. The present approach that we have implemented is to use the LSTM ability to manage the sequential data to improve the accuracy of the credit card fraud detection.

Data Set :

While coming to the data set it has taken from kaggle , an open source platform . The data is on the 2013 transactions of the European credit card users . While coming to the shape of the data it has the 31 attributes and it has the rows of 284807 .In that it has the confidential data of the customers which are the 28 attributes. We have the amount attribute that displays the amount of each transaction that is present and when it comes to the last attribute we have class in this we have assigned that 1 is coming under the fraud transaction and 0 comes under the non fraud transactions.

Data Preprocessing & Data Visualization:

While coming to the data preprocessing techniques for cleaning and to check for the null values we have done checking for the null values data.null() and also checking the duplicate values and we drop the duplicate data using the data.duplicate.sum() also because it will impact on the model performance and the accuracy of the model . here are the below images   **fig1 checking for the null vlaues and the data describe**

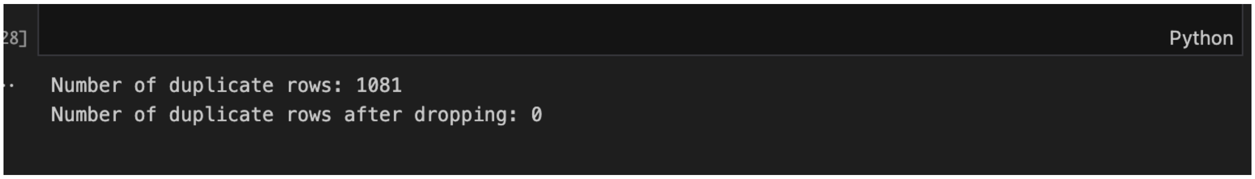


```
    data.isnull()
```

Python

```
      Time    V1    V2    V3    V4    V5    V6    V7    V8    V9  ...   V21   V22   V23   V24   V25
0    False False False False False False False False False False  ... False False False False False
1    False False False False False False False False False False  ... False False False False False
2    False False False False False False False False False False  ... False False False False False
3    False False False False False False False False False False  ... False False False False False
4    False False False False False False False False False False  ... False False False False False
...    ...   ...   ...   ...   ...   ...   ...   ...   ...   ...  ...   ...   ...   ...   ...   ...
4802 False False False False False False False False False False  ... False False False False False
4803 False False False False False False False False False False  ... False False False False False
4804 False False False False False False False False False False  ... False False False False False
4805 False False False False False False False False False False  ... False False False False False
4806 False False False False False False False False False False  ... False False False False False
```

Fig2 will describe the dropping null values :



Python

```
Number of duplicate rows: 1081
Number of duplicate rows after dropping: 0
```

Fig3 will be the data visualization technique that will give the plot of all features of their values and frequencies .

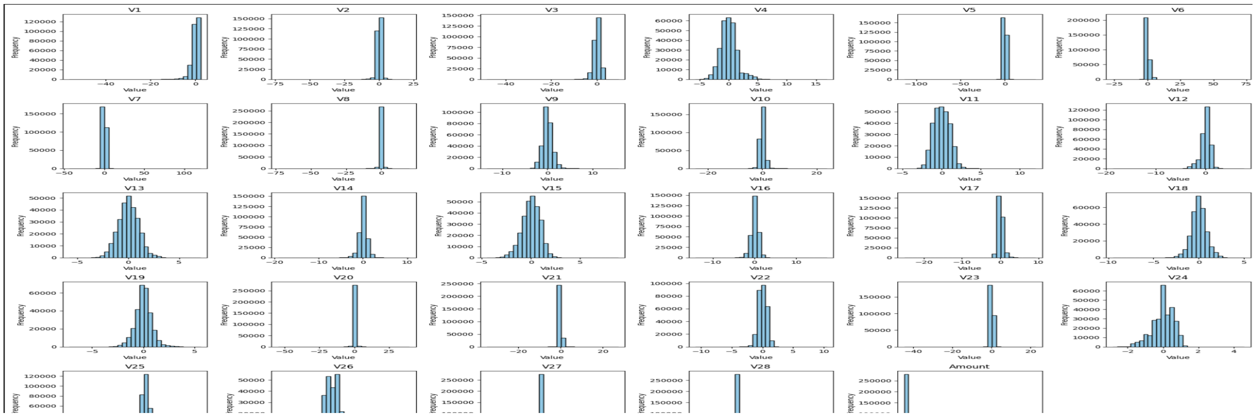
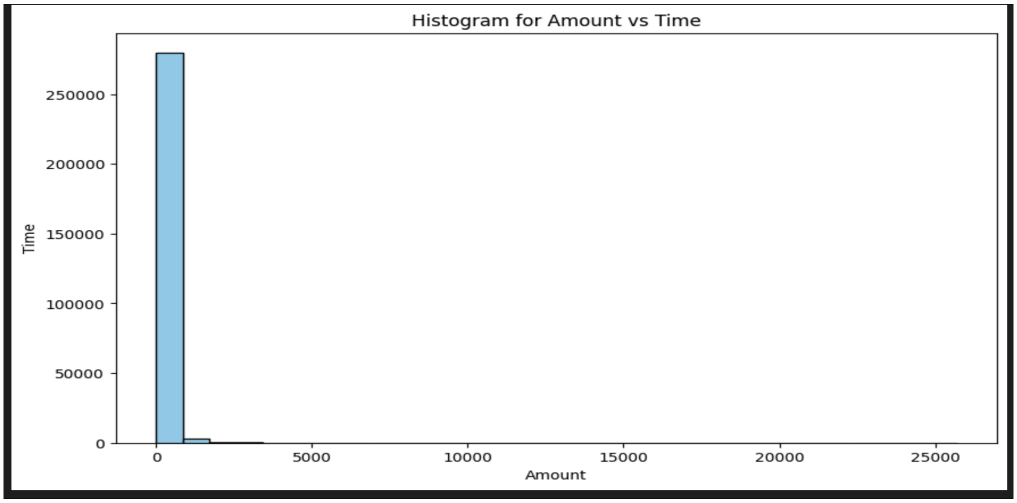


Fig 4 will be the visualization between Amount vs Time



Existing Methods : While coming to the existing methods while coming to both machine learning and deep learning approaches . Here are the few approaches that are applied for this project .

Machine Learning Models :
Logistic Regression: The most common tech or easy method of logistic regression is to classify the fraudulent or non fraudulent transactions. The main purpose of using the algorithm is that it performs well for the binary classification methods and it gives the probability scores but when it comes to the large complex datasets it will not work properly and the results are also not very effective.

SVM: It is the main use for the classification problem which will include fraud detection the main function of the SVM is it will draw a hyperplane whenever you are classifying between two features or multiple features for example like fraud or non fraud transactions and it performs well in the high or complex dataset which have the more features . But when it comes to the non linear data whenever the huge data it leads to high computational problems . We have implemented this method in our project

K-Nearest Neighbors(KNN): K - Nearest neighbor algorithms will work based on the distance that can classify the fraud and non fraud transactions based on the closeness of the distance of other transactions . It may be be effective for the CCF but it needs more hyperparameter tuning and when it comes for the large high dimensional data it may not perform well in large datasets.

Deep Learning methods :

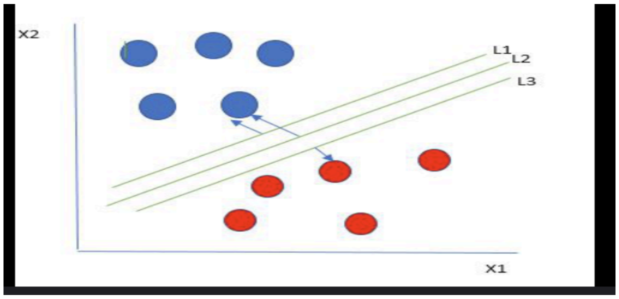
Artificial Neural Network (ANN): ANN will work efficiently in between the neurons to look for the patterns that were happening in between the transaction data and which are able to find the difference between the actual data and the fraud data .

Autoencoders:
Autoencoders are the unsupervised model which will learn the data to reconstruct the input data and it will identifies the anomalies as the transactions are with the high reconstruction erros which will help for the model to detect the outliers or misleading values in the fraud detection particularly when the data of the fraud is less.

Reinforcement Learning :
This is the best way to learn from the rewards using the Q learning algorithm which will optimize the model by learning from the framework.It will adapt to evaluate the patterns which are fraud but it might be difficult to implement to CCF .

Algorithms: In this project Credit Card Fraud transaction i have implemented using Machine algorithms and Deep Learning algorithms . In Machine Learning we have implemented the Random Forest Regression and SVM classification . In Deep Learning we have implemented the LSTM and to data imbalance we have done Under Sampling methods

Support Vector Machine (SVM) :
SVM algorithms are the classifiers and the general and main point of SVM algorithms are as follows: linear classifiers It will work in high dimensional data, in that non-linear task becomes a linear one and will be used for detecting the fraud transaction.ns . One of the most important and key attribute is that the kernel function which will define the classification function which is dot product between the data point and fact This is in that the Actual Support Vector Machine which will find a hyperplane to greatest possible degree of separation between the fraudulent and non fraudulent transactions and which will reduce or minimize overfitting of training data.Given below fig is an example how the hyperplane classifies between two features



Random Forest Algorithm : Random forest algorithm is an ensembling tech which will completely work on the building the multiple decision trees from the data samples which are random and the features .how the multiple decision trees will work is each decision tree will independently predicts the outcome and the final output or the final outcomes predicts will depends on the majority vote . In this approach it will reduce the overfitting and it will improve the accuracy of the model . while come to this tasks like credit card fraud transactions it will work better and in this kind of data we have the data imbalance but random forest algorithm will perform quite good

LSTM: LSTM are the long short term memory cells and these are well used for the sequential data such as the stock prices and transactions history .LSTM consists of three gates which are Forget gate , Input Gate and Output Gate . Forget gates will decide which information has to stay in the cell state . It has the outputs which will be 0 and 1 . 0 means it completely loses the information if it is 1 it will retain the information completely. Input gate has to control how much it will store the new information; it also gives the outputs between 0 to 1 . Output gate will determines the what information has to sent to the hidden state for the current step this gate filters the cell state and the outputs between 0 and 1

Methodology :

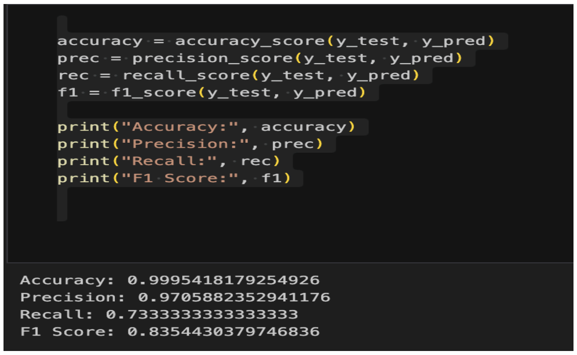
Credit card fraud detection procedure was based on the enhancement and cleansing of the dataset, as well as standardizing the feature amount. The dataset was divided into a training set at 80% and testing set at 20%. The Random Forest Classifier was built achieving an accuracy of 99.95%, the precision of 97.06%, the recall of 73.33% and the F1 score of 83.54%. These performance indices were complemented by the ROC curve and the AUC, pointing to acceptable charge discrimination. A Confusion Matrix was used for true/false positive/negative evaluations. The Support Vector Machine model was also developed to identify a benchmark and achieved 99.94% accuracy; precision of 98.33%; and a recall score of 65.56%. The results showed that the specialty of the Random Forest for higher recall, which means this model is more appropriate for fraud detection. Thus, it is determined that in this case, the Random Forest method gives a higher accuracy in comparison with the SVM model.

After implementing of the traditional machine learning we have implemented the Deep learning neural networks using LSTM and data imbalance technique Under Sampling first it will preprocess the credit card fraud data by splitting the features of X and y labels after selecting the dependent and independent features we are splitting the data into training and test and the input features are reshaped according to the input of the LSTM models and the labels are one hot encode. A sequential LSTM has built in three layers and each LSTM layer has the specified units of 50,80,120 with the regularization technique of l2 and to prevent the overfitting we are introducing the dropout layers to reduce the overfitting and to improve the accuracy and performance and efficiency of the model we are going to use the hyperparameter tuning like using the Adam optimizer and the loss function which is cross entropy loss and then it was trained for 20 epochs after the training of the model the model predictions on the test set are evaluating using the accuracy , precision , recall and F1 score and we got the pretty good results but due to the imbalance of the data we are going to use the under sampling technique .

After implementing LSTM we are using undersampling tech to balance the imbalance data we are using the under sampling technique to match with minority classes; it will reduce the majority classes . To create the balance in the dataset we are selecting the randomly of true transaction to fraud transctions.

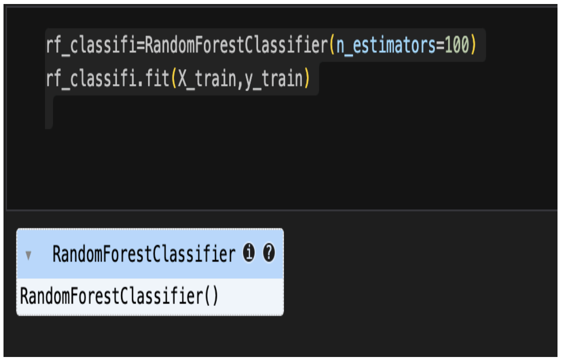
Machine Learning Results :

While compared to the results using traditional machine algorithms accuracy and precision for two algorithms is varies in small percentages but while coming to the confusion matrices in random forest 66 transactions of fraud were detected while remaining of 56654 samples of non fraud are ignored as fraud and 24 fraud transactions are failed to detect the false negatives and while coming to the 2 non fraud trans were predicted and in SVM we have 59 fraud transactions and 56655 non fraud transactions were missed 31 frauds incident triggered in one instant .RF classifies better TP and reduce FN compare to SVM



```
accuracy = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", prec)
print("Recall:", rec)
print("F1 Score:", f1)
```

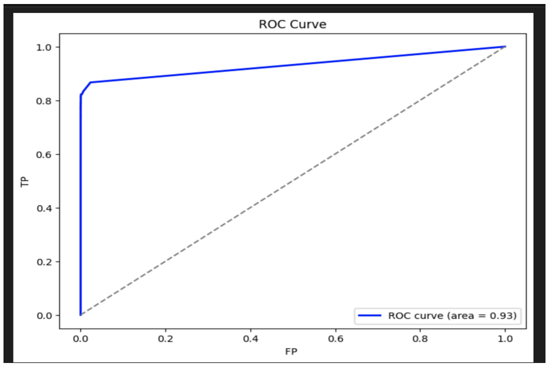
```
Accuracy: 0.9995418179254926
Precision: 0.9705882352941176
Recall: 0.7333333333333333
F1 Score: 0.8354430379746836
```



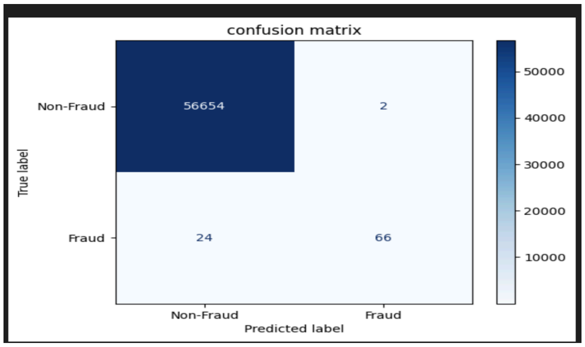
```
rf_classifi=RandomForestClassifier(n_estimators=100)
rf_classifi.fit(X_train,y_train)
```

```
RandomForestClassifier()
```

ROC Curve of Random Forest :



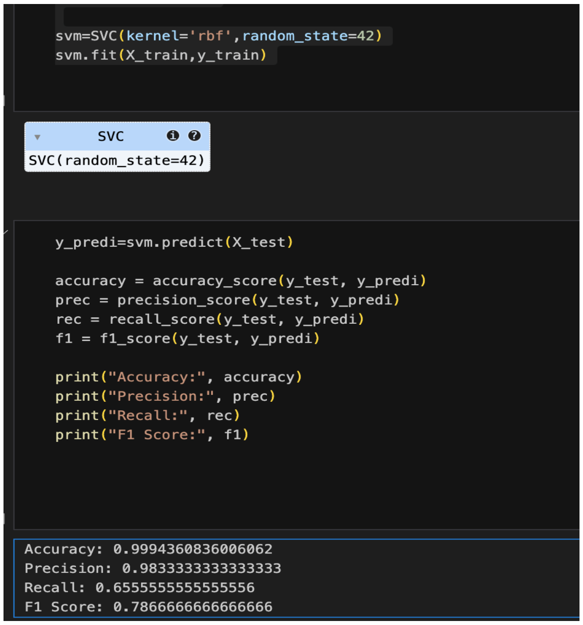
Confusion Matrix of Random Forest :



	Non-Fraud	Fraud
Non-Fraud	56654	2
Fraud	24	66

SVM RESULTS

Accuracy and confusion matrix for SVM

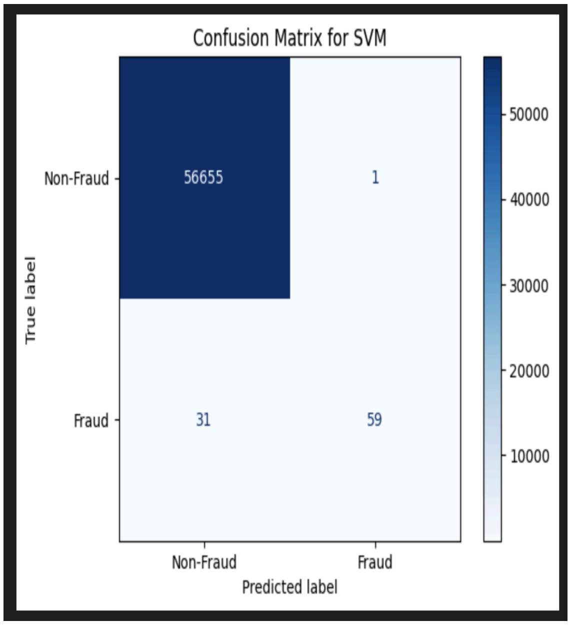


```
svm=SVC(kernel='rbf',random_state=42)
svm.fit(X_train,y_train)
```

```
SVC(random_state=42)
```

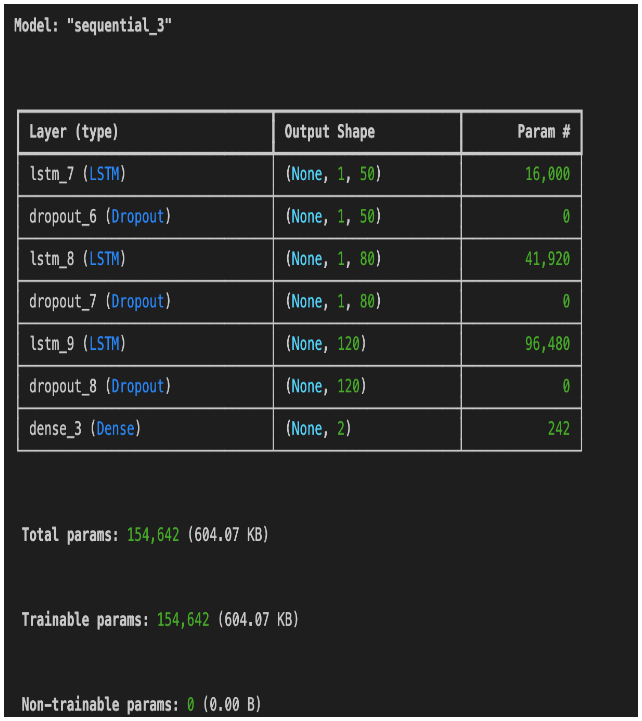
```
y_predi=svm.predict(X_test)

accuracy = accuracy_score(y_test, y_predi)
prec = precision_score(y_test, y_predi)
rec = recall_score(y_test, y_predi)
f1 = f1_score(y_test, y_predi)

print("Accuracy:", accuracy)
print("Precision:", prec)
print("Recall:", rec)
print("F1 Score:", f1)
```

```
Accuracy: 0.9994360836006062
Precision: 0.9833333333333333
Recall: 0.6555555555555556
F1 Score: 0.7866666666666666
```



	Non-Fraud	Fraud
Non-Fraud	56655	1
Fraud	31	59

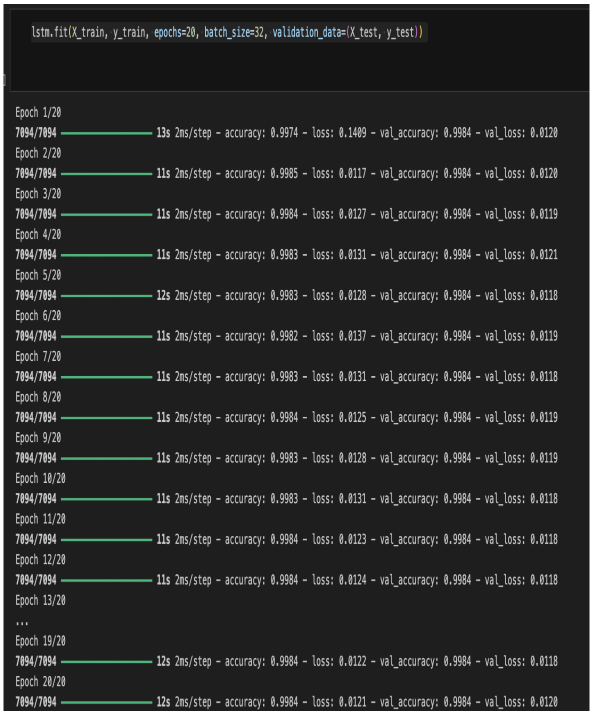
Deep Learning (LSTM) results :



Model: "sequential_3"

Layer (type)	Output Shape	Param #
lstm_7 (LSTM)	(None, 1, 50)	16,000
dropout_6 (Dropout)	(None, 1, 50)	0
lstm_8 (LSTM)	(None, 1, 80)	41,920
dropout_7 (Dropout)	(None, 1, 80)	0
lstm_9 (LSTM)	(None, 120)	96,480
dropout_8 (Dropout)	(None, 120)	0
dense_3 (Dense)	(None, 2)	242

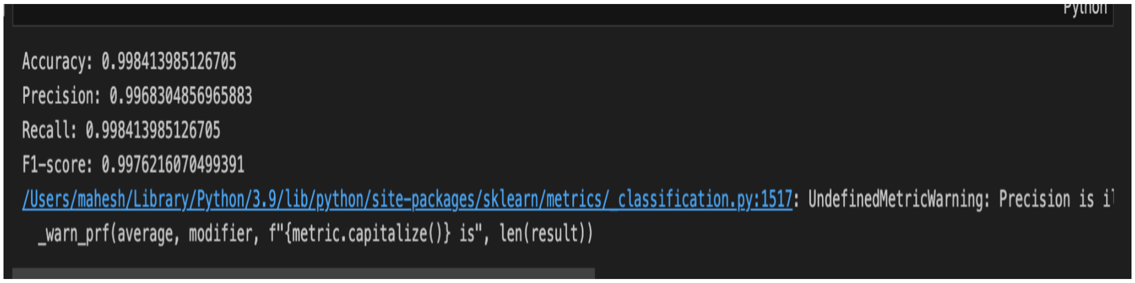
```
Total params: 154,642 (604.07 KB)

Trainable params: 154,642 (604.07 KB)

Non-trainable params: 0 (0.00 B)
```



```
lstm.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))
```

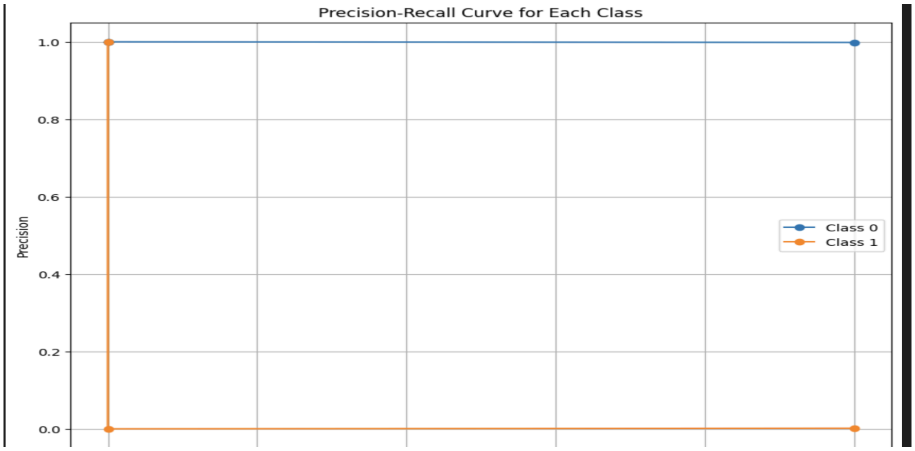
```
Epoch 1/20
7094/7094 ━━━━━━━━━━━━━━━━━━━━ 13s 2ms/step - accuracy: 0.9974 - loss: 0.1409 - val_accuracy: 0.9984 - val_loss: 0.0120
Epoch 2/20
7094/7094 ━━━━━━━━━━━━━━━━━━━━ 11s 2ms/step - accuracy: 0.9985 - loss: 0.0117 - val_accuracy: 0.9984 - val_loss: 0.0120
Epoch 3/20
7094/7094 ━━━━━━━━━━━━━━━━━━━━ 11s 2ms/step - accuracy: 0.9984 - loss: 0.0127 - val_accuracy: 0.9984 - val_loss: 0.0119
Epoch 4/20
7094/7094 ━━━━━━━━━━━━━━━━━━━━ 11s 2ms/step - accuracy: 0.9983 - loss: 0.0131 - val_accuracy: 0.9984 - val_loss: 0.0121
Epoch 5/20
7094/7094 ━━━━━━━━━━━━━━━━━━━━ 12s 2ms/step - accuracy: 0.9983 - loss: 0.0128 - val_accuracy: 0.9984 - val_loss: 0.0118
Epoch 6/20
7094/7094 ━━━━━━━━━━━━━━━━━━━━ 11s 2ms/step - accuracy: 0.9982 - loss: 0.0137 - val_accuracy: 0.9984 - val_loss: 0.0119
Epoch 7/20
7094/7094 ━━━━━━━━━━━━━━━━━━━━ 11s 2ms/step - accuracy: 0.9983 - loss: 0.0131 - val_accuracy: 0.9984 - val_loss: 0.0118
Epoch 8/20
7094/7094 ━━━━━━━━━━━━━━━━━━━━ 11s 2ms/step - accuracy: 0.9984 - loss: 0.0125 - val_accuracy: 0.9984 - val_loss: 0.0119
Epoch 9/20
7094/7094 ━━━━━━━━━━━━━━━━━━━━ 11s 2ms/step - accuracy: 0.9983 - loss: 0.0128 - val_accuracy: 0.9984 - val_loss: 0.0119
Epoch 10/20
7094/7094 ━━━━━━━━━━━━━━━━━━━━ 11s 2ms/step - accuracy: 0.9983 - loss: 0.0131 - val_accuracy: 0.9984 - val_loss: 0.0118
Epoch 11/20
7094/7094 ━━━━━━━━━━━━━━━━━━━━ 11s 2ms/step - accuracy: 0.9984 - loss: 0.0123 - val_accuracy: 0.9984 - val_loss: 0.0118
Epoch 12/20
7094/7094 ━━━━━━━━━━━━━━━━━━━━ 11s 2ms/step - accuracy: 0.9984 - loss: 0.0124 - val_accuracy: 0.9984 - val_loss: 0.0118
Epoch 13/20
...
Epoch 19/20
7094/7094 ━━━━━━━━━━━━━━━━━━━━ 12s 2ms/step - accuracy: 0.9984 - loss: 0.0122 - val_accuracy: 0.9984 - val_loss: 0.0118
Epoch 20/20
7094/7094 ━━━━━━━━━━━━━━━━━━━━ 12s 2ms/step - accuracy: 0.9984 - loss: 0.0121 - val_accuracy: 0.9984 - val_loss: 0.0120
```

Accuracy of the LSTM model with data imbalance :

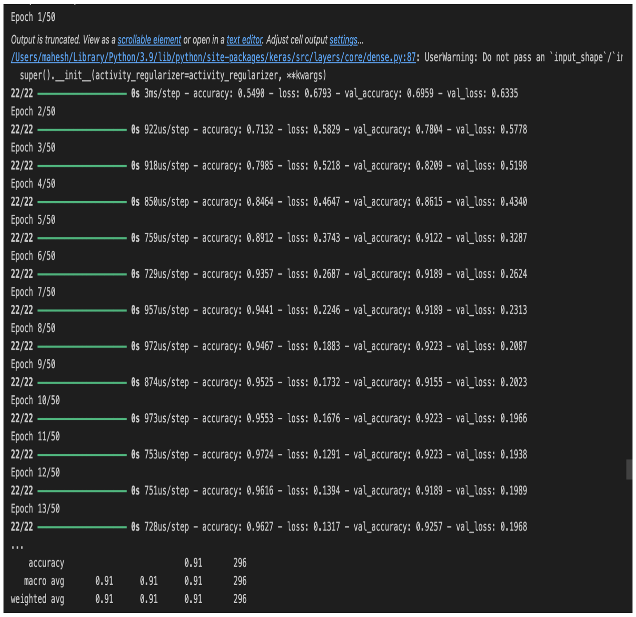


```
Accuracy: 0.998413985126705
Precision: 0.9968304856965883
Recall: 0.998413985126705
F1-score: 0.9976216070499391
/Users/mahesh/Library/Python/3.9/lib/python/site-packages/sklearn/metrics/_classification.py:1517: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

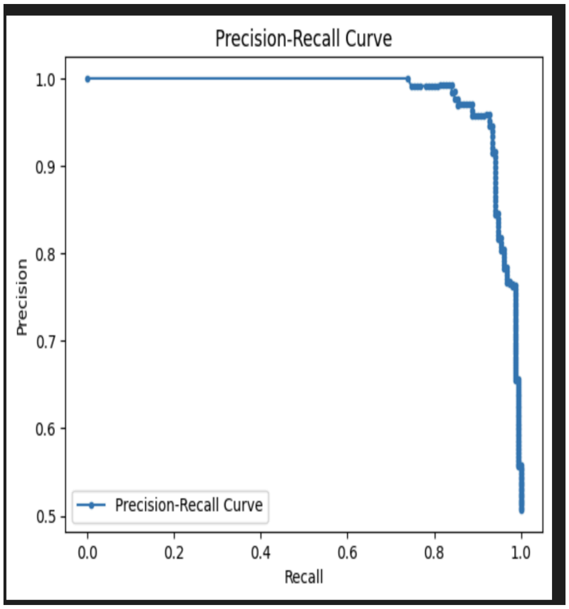
ROC Curve :

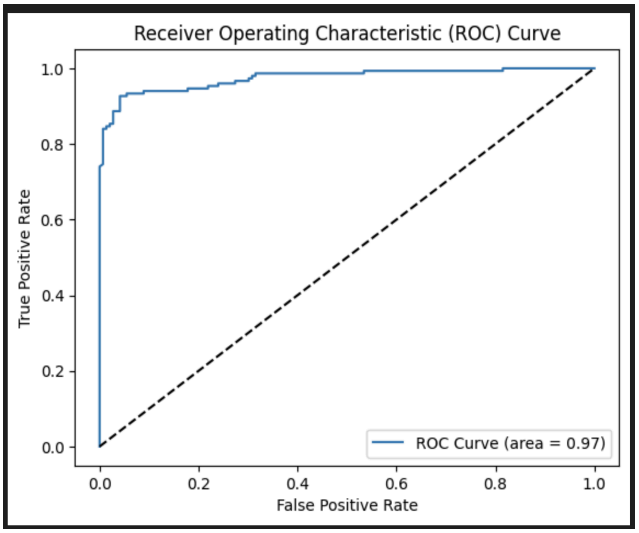


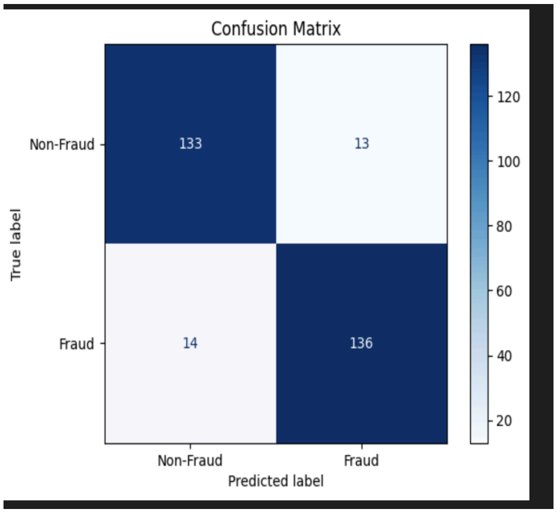
UNDERSAMPLING RESULTS :



```
Epoch 1/50
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
/Users/mahesh/Library/Python/3.9/lib/python/site-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`ir
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5490 - loss: 0.6793 - val_accuracy: 0.6959 - val_loss: 0.6335
Epoch 2/50
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 922us/step - accuracy: 0.7132 - loss: 0.5829 - val_accuracy: 0.7804 - val_loss: 0.5778
Epoch 3/50
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 918us/step - accuracy: 0.7985 - loss: 0.5218 - val_accuracy: 0.8209 - val_loss: 0.5198
Epoch 4/50
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 850us/step - accuracy: 0.8464 - loss: 0.4647 - val_accuracy: 0.8615 - val_loss: 0.4340
Epoch 5/50
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 759us/step - accuracy: 0.8912 - loss: 0.3743 - val_accuracy: 0.9122 - val_loss: 0.3287
Epoch 6/50
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 729us/step - accuracy: 0.9357 - loss: 0.2687 - val_accuracy: 0.9189 - val_loss: 0.2624
Epoch 7/50
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 957us/step - accuracy: 0.9441 - loss: 0.2246 - val_accuracy: 0.9189 - val_loss: 0.2313
Epoch 8/50
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 972us/step - accuracy: 0.9467 - loss: 0.1883 - val_accuracy: 0.9223 - val_loss: 0.2087
Epoch 9/50
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 874us/step - accuracy: 0.9525 - loss: 0.1732 - val_accuracy: 0.9155 - val_loss: 0.2023
Epoch 10/50
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 973us/step - accuracy: 0.9553 - loss: 0.1676 - val_accuracy: 0.9223 - val_loss: 0.1966
Epoch 11/50
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 753us/step - accuracy: 0.9724 - loss: 0.1291 - val_accuracy: 0.9223 - val_loss: 0.1938
Epoch 12/50
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 751us/step - accuracy: 0.9616 - loss: 0.1394 - val_accuracy: 0.9189 - val_loss: 0.1989
Epoch 13/50
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 728us/step - accuracy: 0.9627 - loss: 0.1317 - val_accuracy: 0.9257 - val_loss: 0.1968
...
    accuracy                         0.91       296
   macro avg       0.91      0.91      0.91       296
weighted avg       0.91      0.91      0.91       296
```







After applying the undersampling techniques LSTM model overcomes the overfitting issues and it gets the better accuracy ad precision and recall and the outcome showing the perfect This outcome suggests almost perfect balance between precision and recall: the higher the recall level, the higher is the precision, but as soon as the recall approaches 1 as in our case the precision is still consistently high measure of performance of the classifier is as follows, where the ROC curve for the classifier gives an AUC of 0.97, which denotes that the classifier has very good discriminant power between the positive and negative classes.

Conclusion : while coming to the conclusion for the CCF detection systems using the transition ML algorithms will give the results but sometimes due to the higher dimensional data this will go overfitting issues and it will effect the performance of the model using DL methods using LSTM we faces the overfitting and lot of disturbances in the ROC curve this is due to the imbalance of the data but using the UnderSampling Methods we got the better accuracy and better results .

References :

https://matplotlib.org/

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

https://scikit-learn.org/stable/modules/preprocessing.html

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

https://en.wikipedia.org/wiki/Long_short-term_memory

https://www.tensorflow.org/tutorials/keras/classification

https://pypi.org/project/tensorflow/