Credit Card Fraud Detection using Deep Learning and Machine learning

***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**

***Index Terms*—Deep Learning, Machine learning, LSTM,Under sampling , SVM , Random Forest**

I. 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, companies and financial institutions are developing their own automated fraud detection systems. The main goal of credit card fraud detection is to create a machine learning model based on existing transaction data, with the primary aim of distinguishing between fraudulent and non-fraudulent transactions. Such a detection system enables organizations to determine whether an incoming transaction is fraudulent. Implementing this type of system technically requires overcoming challenges related to the quick response time, sensitivity, and feature pre-processing.

The use of machine learning and deep learning algorithms has been implemented to solve issues in various fields, including banking, insurance, health, surveillance, and fraud detection. In this project, we employed machine learning algorithms and deep learning techniques to investigate the application of credit card fraud detection in the banking industry. Traditional machine learning algorithms, such as Support Vector Machines (SVM), are suitable for binary classification and commonly used for image recognition and banking. Deep learning methods were implemented for larger datasets, and neural networks were applied for fraud detection.

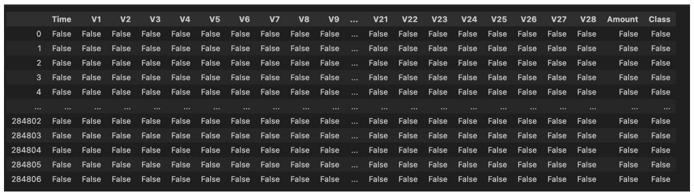
Deep learning researchers are increasingly interested in applying these techniques in CCF detection systems, although limited research has explored the use of deep learning neural networks specifically for this task. In this project, we are using a Long Short-Term Memory (LSTM) model to distinguish between fraudulent and non-fraudulent transactions. The class imbalance problem, characterized by a significantly smaller number of fraudulent transactions compared to non-fraudulent ones, is addressed in the LSTM model by adding additional layers to improve feature extraction and classification. Our approach leverages the LSTM's ability to handle sequential data to enhance the accuracy of credit card fraud detection.

II. Dataset

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.

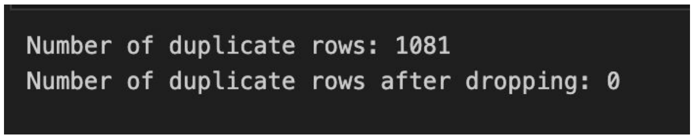
III. Data Visualization Data Preprocessing

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



```
         Time    V1    V2    V3    V4    V5    V6    V7    V8    V9  ...   V21   V22   V23   V24   V25   V26   V27   V28  Amount  Class
0       False False False False False 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 False False False   False  False
2       False False False False False 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 False False False   False  False
4       False False False False False False False False False False  ... False False False False False False False False   False  False
...       ...   ...   ...   ...   ...   ...   ...   ...   ...   ...  ...   ...   ...   ...   ...   ...   ...   ...   ...     ...    ...
284802  False False False False False False False False False False  ... False False False False False False False False   False  False
284803  False False False False False False False False False False  ... False False False False False False False False   False  False
284804  False False False False False False False False False False  ... False False False False False False False False   False  False
284805  False False False False False False False False False False  ... False False False False False False False False   False  False
284806  False False False False False False False False False False  ... False False False False False False False False   False  False
```

Fig. 1: checking for the null values



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

Fig. 2: Removing the duplicates

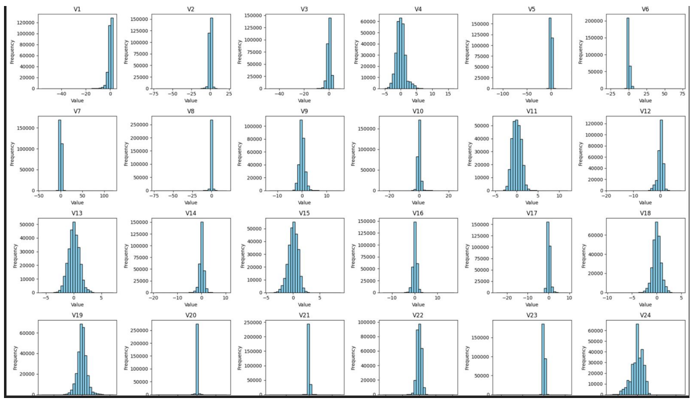


Fig. 3: Histogram visualization of all the features in data

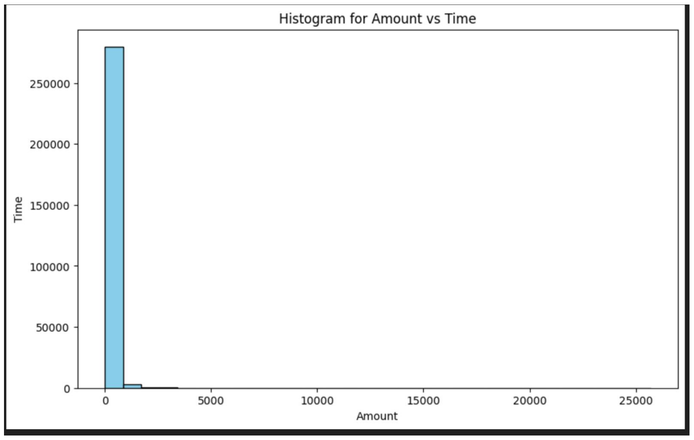


Fig. 4: Histogram visualization of Amount vs Time

IV. 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 . **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.

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 .

V. 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

A. *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

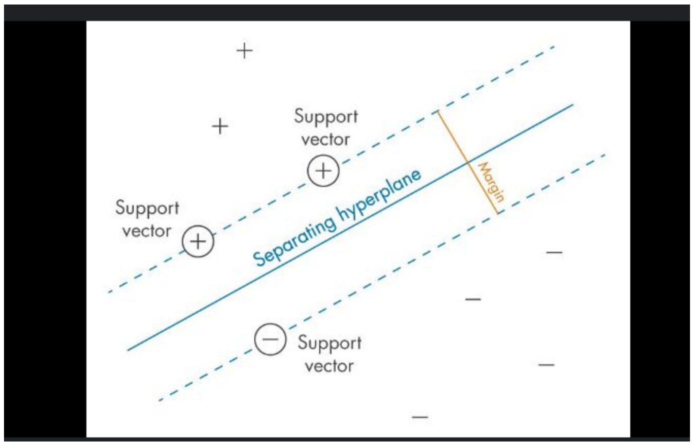


Fig. 5: svm

B. 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

C. Long short Term memory

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

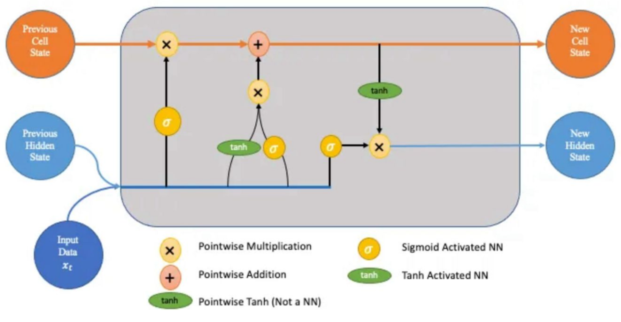


Fig. 6: overall LSTM architecture

D. Under Sampling

A technique to handle class imbalance (i.e. the minority class such as fraudulent transactions is very high compared to the majority class) is called undersampling. Here, we randomly remove a part of the majority class to balance this dataset. Undersampling is useful for LSTM model because it makes sure the model doesn't become biased towards the majority class when training, as they see more number of examples of each class. Since recall for the minority (fraud) class improves, the learning of the LSTM is better from a reduced number of non-fraudulent examples. Nevertheless, it can lead to losing important information from the majority class, and thus lower the model accuracy in case of non fraud cases.

VI. Methodology

The methodology for this project involves the following steps:

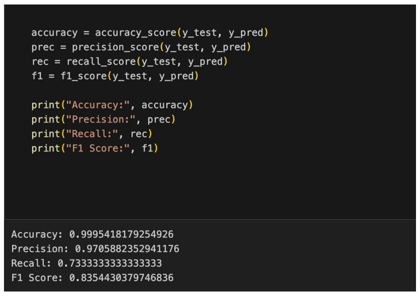
- Credit card fraud detection procedure was based on the enhancement and cleansing of the dataset, as well as standardizing the feature amount. The dataset was divided into a training set at 80% and a testing set at 20%. The Random Forest Classifier was built, achieving an accuracy of 99.95%, a precision of 97.06%, a recall of 73.33%, and an 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, achieving 99.94% accuracy, a precision of 98.33%, and a recall score of 65.56%. The results highlighted the strength of the Random Forest in achieving higher recall, indicating that this model is more appropriate for fraud detection. Thus, it is determined that in this case, the Random Forest method provides higher accuracy in comparison with the SVM model.
- After implementing traditional machine learning, we proceeded to implement deep learning neural networks using LSTM and an undersampling technique to address data imbalance. First, the credit card fraud data was preprocessed by splitting the features X and y labels. After selecting the dependent and independent features, the data was divided into training and test sets, and the input features were reshaped according to the requirements of the LSTM model, with labels one-hot encoded. A sequential LSTM model was constructed with three layers, each having specified units of 50, 80, and 120, along with $L2$ regularization to prevent overfitting. Dropout layers were also introduced to improve the model's accuracy, performance, and efficiency. Hyperparameter tuning included the use of the Adam optimizer and cross-entropy loss function, and the model was trained for 20 epochs. After training, model predictions on the test set were evaluated using metrics such as accuracy, precision, recall, and F1 score. Despite achieving promising results, due to data

imbalance, we employed an undersampling technique to further enhance model performance.
- 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.

VII. Machine Learning results

A. *Random Forest Results :*

While comparing the results obtained using traditional machine learning algorithms, the accuracy and precision for the two algorithms vary slightly. However, examining the confusion matrices provides more insight into the models' performance. In the Random Forest model, 66 fraudulent transactions were correctly detected, while 56654 samples of non-fraud transactions were incorrectly classified as fraud. Additionally, 24 fraudulent transactions were not detected (false negatives), and 2 non-fraudulent transactions were misclassified.and we got an accuarcy precision , recall , and f1 score around in the given below figure



```
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
```

Fig. 7: given below is the classification report of Random forest results

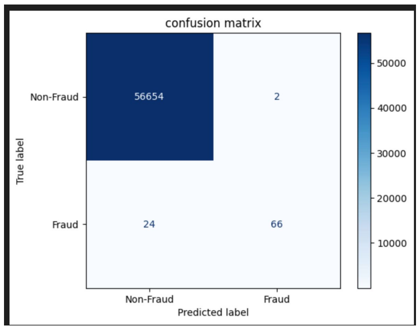


Fig. 8: confusion matrix for the Random forest algorithm

In the image you see the ROC curve, which is a plot of true positive rate versus the false positive rate at different decision thresholds. The shape of the curve indicates the model is working well — most points lie on the top left of the corner, which means high true positive rate with low false positive rate. The final area of 0.93 indicates the excellent performance of the model. The presence of this high AUC value suggests that the model performs well in classifying the instances as positive and negative; a good condition that makes the model trusted to be used in credit card fraud detection, for example, since we need to distinguish between the classes (fraud, non-fraud).

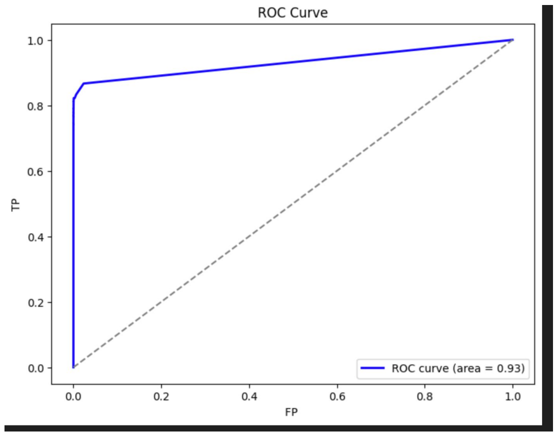
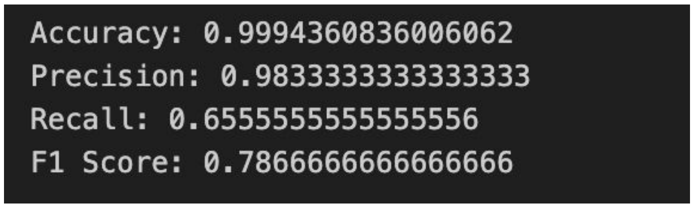


Fig. 9: ROC curve for random forest algorithm

B. *Support Vector Machine Results :*

For the SVM model, 59 fraudulent transactions were detected, while 56655 non-fraud transactions were misclassified, resulting in 31 false negatives in a single instance. The Random Forest classifier demonstrates better performance in classifying true positives (TP) and reducing false negatives (FN) compared to the SVM model.Given below will give me the quality analysis of the SVM model and here are the results and given figure will show the classification report.



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

Fig. 10: classfication report of SVM model

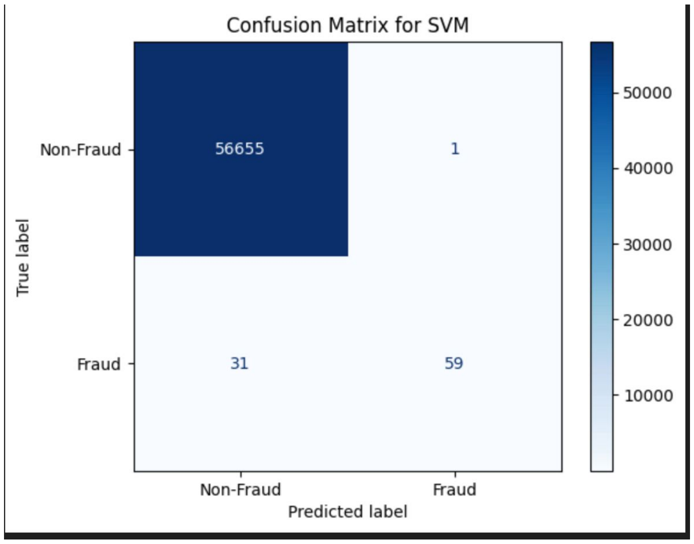


Fig. 11: confusion matrix of SVM model

VIII. Long Short Term Memory Results

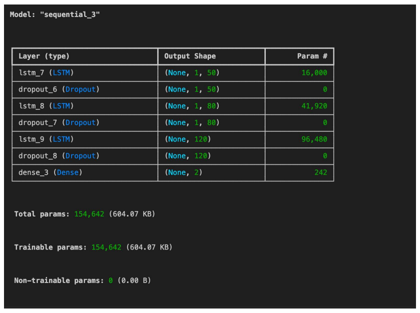
A. *LSTM Results*

Now we are presenting the results of the LSTM model, and here is the information about the architecture. The neural network consists of three LSTM layers with increasing output shapes:

- LSTM layer 1: Output shape = $(\text{None}, 1, 50)$, Parameters = 16,000
- LSTM layer 2: Output shape = $(\text{None}, 1, 80)$, Parameters = 41,920
- LSTM layer 3: Output shape = $(\text{None}, 120)$, Parameters = 96,480

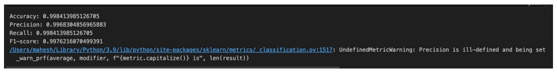
Dropout layers are used following each LSTM layer to prevent overfitting, and they do not contain any trainable parameters. The final Dense layer has 242 parameters and outputs two classes, indicating a classification task with two output classes.

Out of a total of 154,642 parameters, all are trainable. This model is designed for sequential data tasks.



```
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)
```

Fig. 12: model summary of the lstm architecure



```
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 ill-defined and being set
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Fig. 13: accuarcy report of the lstm model

The below image shows a Precision-Recall Curve for two classes (Class 0 and Class 1). Class 0 has a perfect precision of 1.0 across all recall values, while Class 1 has a precision of 0.0, indicating a highly imbalanced or poorly performing model for Class 1.

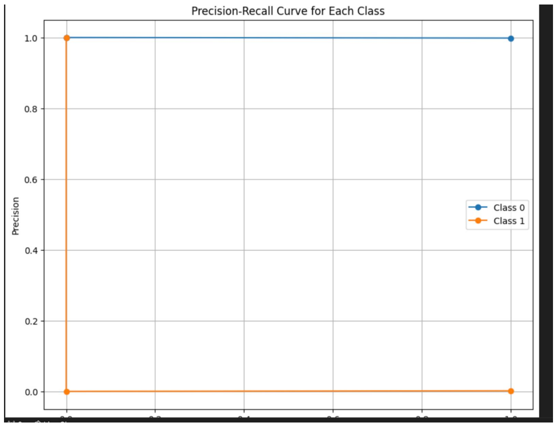
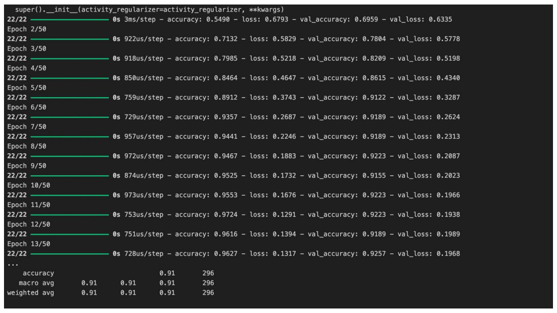


Fig. 14: ROC curve of the lstm model

IX. Undersampling Results

A. *undersampling results*

The first example showcases how the training updates of a neural network model goes over multiple epochs. We then display the performance metrics for each epoch, with accuracy, loss, validation accuracy and validation loss. We train our model for 50 epochs, and only the first 13 epochs are shown. The accuracy is increased from 0.5490 in epoch 1 to 0.9627 epoch 13 and changes in loss goes from 0.6793 to 0.1317. Like wise the validation accuracy rises to 0.9257 and the validation loss is decreased from 0.6335 to 0.1968, the better performance on generalization at the expense of training as this happens. We also see efficient processing times (in microseconds per step) for training steps per epoch.



```
  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
```

Fig. 15: model summary of the lstm architecure

The graph for the Precision Curve shows that precision keeps stable on the higher side (close to 1.0) almost on all recall values, which means good performance in terms of precision.n. Let's look at this, however, recall increasingly approaches 1.0, which shows a tradeoff between precision and recall above certain recall levels.

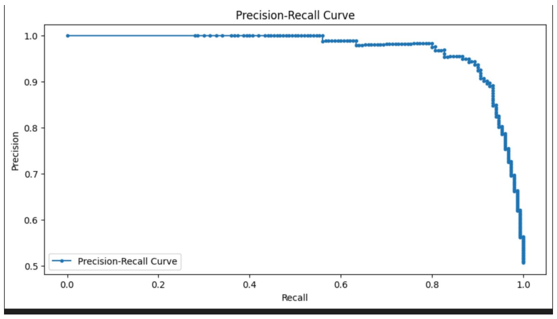


Fig. 16: undersampling training precision curve

The ROC Curve in the second image shows a strong classifier performance with a high true (sensitivity) and low false (1 - specificity) positive rate in the curve.erms of precision. However, as recall approaches 1.0, precision starts to drop, suggesting a trade-off between precision and recall at higher recall levels. This model has a high AUC value of 0.96 it well tell us that the model is performing very well.

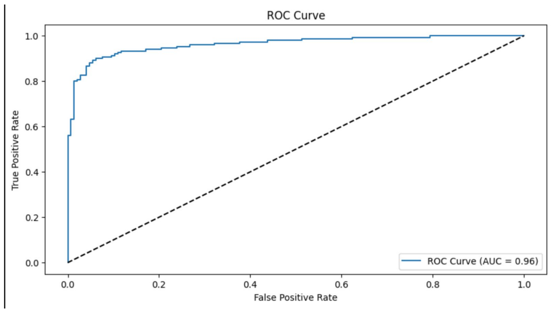


Fig. 17: undersampling training recall curve

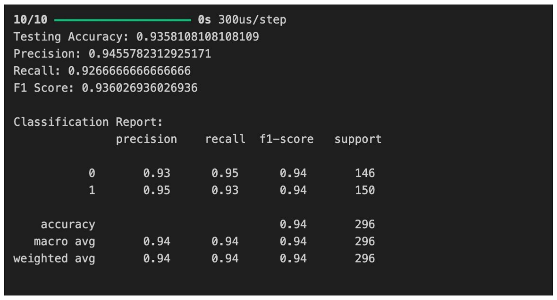
X. Undersampling Testing results

A. *Classification Report*

Here is the classification report with the accuracy on testing data of 93.58

$$\text{F1} = 0.94 = 2 \times \frac{(0.93 \times 0.95)}{(0.93 + 0.95)}$$

All of the above accuracy, macro average, and weighted average are also 0.94.



```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 300us/step
Testing Accuracy: 0.9358108108108109
Precision: 0.9455782312925171
Recall: 0.9266666666666666
F1 Score: 0.936026936026936

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.95      0.94       146
           1       0.95      0.93      0.94       150

    accuracy                           0.94       296
   macro avg       0.94      0.94      0.94       296
weighted avg       0.94      0.94      0.94       296
```

Fig. 18: undersampling test classification report

B. *Precision-Recall Curve*

The Precision vs Recall curve for different thresholds is shown in the graph below. Initially, the model achieves perfect precision and recall, but then it begins to approximate both precision and recall, indicating good performance.

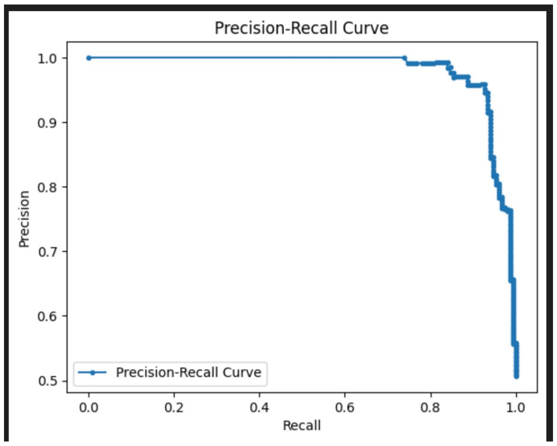


Fig. 19

C. *ROC Curve*

A Receiver Operating Characteristic (ROC) curve with an AUC (Area Under the Curve) of 0.97 is displayed below. The AUC measures the model's performance against the false positive rate and true positive rate. A large AUC value indicates that the model performs well.

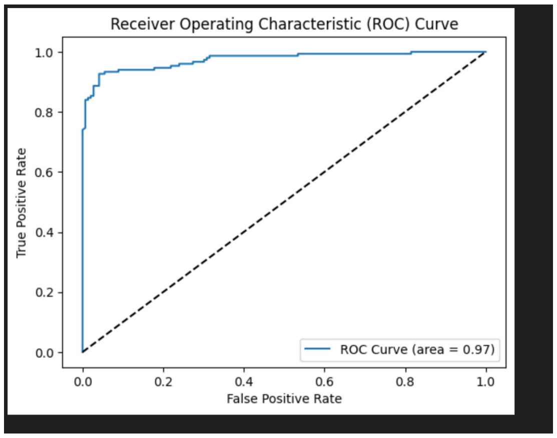


Fig. 20: ROC Curve with AUC of 0.97

D. *Confusion Matrix*

The confusion matrix shows how well the model predicted the two classes. There were 479 false positives and 14 false negatives for Class "0", and 470 false positives and 22 false negatives for Class "1". This suggests strong classification with relatively few errors.

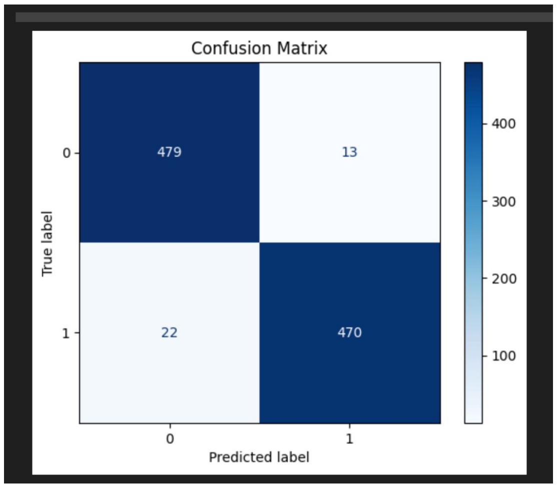


Fig. 21: Confusion Matrix

XI. 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

Below are the references for the papers you mentioned:

References

[1] John Richard D. Kho, Larry A. Vea, *Credit Card Fraud Detection Based on Transaction Behaviour*, Proc. of the 2017 IEEE Region 10 Conference (TENCON), Malaysia, November 5-8, 2017.

[2] Clifton Phua, Vincent Lee, Kate Smith, Ross Gayler, *A Comprehensive Survey of Data Mining-based Fraud Detection Research*, School of Business Systems, Faculty of Information Technology, Monash University, Wellington Road, Clayton, Victoria 3800, Australia.

[3] Suman, *Survey Paper on Credit Card Fraud Detection*, Research Scholar, GJUST Hisar HCE, Sonepat, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 3 Issue 3, March 2014.

[4] Wen-Fang Yu, Na Wang, *Research on Credit Card Fraud Detection Model Based on Distance Sum*, 2009 International Joint Conference on Artificial Intelligence.

[5] B. Lebichot, G. M. Paldino, W. Siblini, L. He-Guelton, F. Oblé, G. Bontempi, *Incremental learning strategies for credit cards fraud detection*, Int. J. Data Sci. Anal., vol. 12, no. 2, pp. 165-174, Aug. 2021.

[6] X. Zhang, Y. Han, W. Xu, Q. Wang, *HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture*, Inf. Sci., vol. 557, pp. 302-316, May 2021.

[7] S. Bakhtiari, Z. Nasiri, J. Vahidi, *Credit card fraud detection using ensemble data mining methods*, Multimedia Tools Appl., vol. 82, no. 19, pp. 29057-29075, Aug. 2023.

[8] M.-H. Yang, J.-N. Luo, M. Vijayalakshmi, S. M. Shalinie, *Contactless credit cards payment fraud protection by ambient authentication*, Sensors, vol. 22, no. 5, pp. 1989, Mar. 2022.

[9] J. Wang, W. Liu, Y. Kou, D. Xiao, X. Wang, X. Tang, *Approx-SMOTE federated learning credit card fraud detection system*, Proc. IEEE 47th Annu. Comput. Softw. Appl. Conf. (COMPSAC), pp. 1370-1375, Jun. 2023.

[10] A. A. El-Naby, E. E.-D. Hemdan, A. El-Sayed, *An efficient fraud detection framework with credit card imbalanced data in financial services*, Multimedia Tools Appl., vol. 82, no. 3, pp. 4139-4160, Jan. 2023.