Credit Card Fraud Detection Model

Machine Learning Models Comparison

Logistic Regression

Long Short Term Memory(LSTM)

Random Forest

Group-6

Scam Guards

Team Members:

Ayon Kumar Das, *Computer Science*, adas010@citymail.cuny.edu

Myesha Mahazabeen, *Computer Science*, mmahaza000@citymail.cuny.edu

Najia Jahan, *Computer Science*, njahan010@citymail.cuny.edu

Professor Yunhua Zhao

CSC 44800

Final Project

12/19/2023

Table of Contents

Abstract

This project introduces a practical approach to detect credit card fraud in real-time electronic transactions. We employ Logistic Regression, Long Short Term Memory (LSTM), and Random Forest models, emphasizing precision and recall. The study involves a detailed comparison of these models, assessing their performance using key metrics such as Accuracy, Precision, Recall, F1-Score, and Matthews Correlation Coefficient. By leveraging these techniques, our aim is to provide insights for improving fraud detection in the ever-evolving landscape of electronic transactions

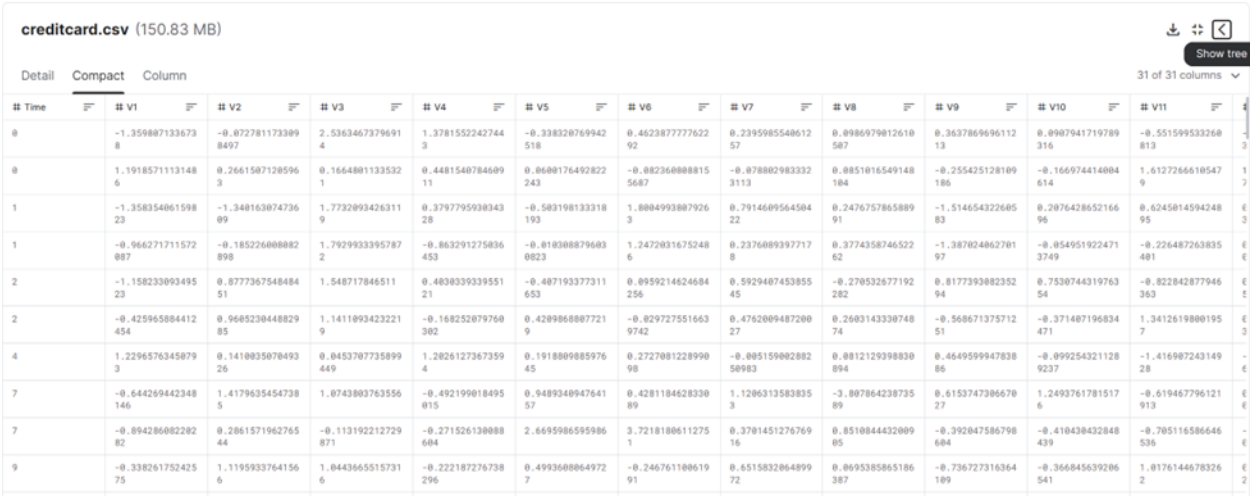
Introduction

In the age of digital transactions, the rise of credit card fraud urges us to find strong ways to stop it. Banks and financial companies face a big challenge in protecting people from tricky fraud tactics. This study goes a step further by checking not just one, but three computer methods—logistic regression, long short term memory(LSTM), and random forest—to see how good they are at telling apart real and fake credit card transactions. Our main goal is to share smart ideas that can help make even better systems to catch fraud. For this project, we're using a dataset with a substantial volume of transactions, ensuring our study covers a wide range of situations in the real world of electronic transactions. This extensive dataset allows us to comprehensively assess the performance of the models and draw meaningful insights to contribute to the ongoing efforts in enhancing credit card fraud detection.

Dataset Overview

The dataset utilized in this project provides a snapshot of credit card transactions conducted by European users in September 2013, spanning a concise two-day period. A total of 284,807 transactions are included, with only 492 identified as fraudulent—comprising a mere 0.172% of the dataset. To address confidentiality concerns, the dataset primarily includes features resulting from Principal Component Analysis (PCA) transformation, with 'Time' (representing seconds since the first transaction) and 'Amount' (indicating the transaction sum) being the exceptions. The crucial 'Class' feature serves as the label, denoting whether a transaction is fraudulent (Class 1) or non-fraudulent (Class 0) **[1]**.

Below are two small snippets of the dataset used.



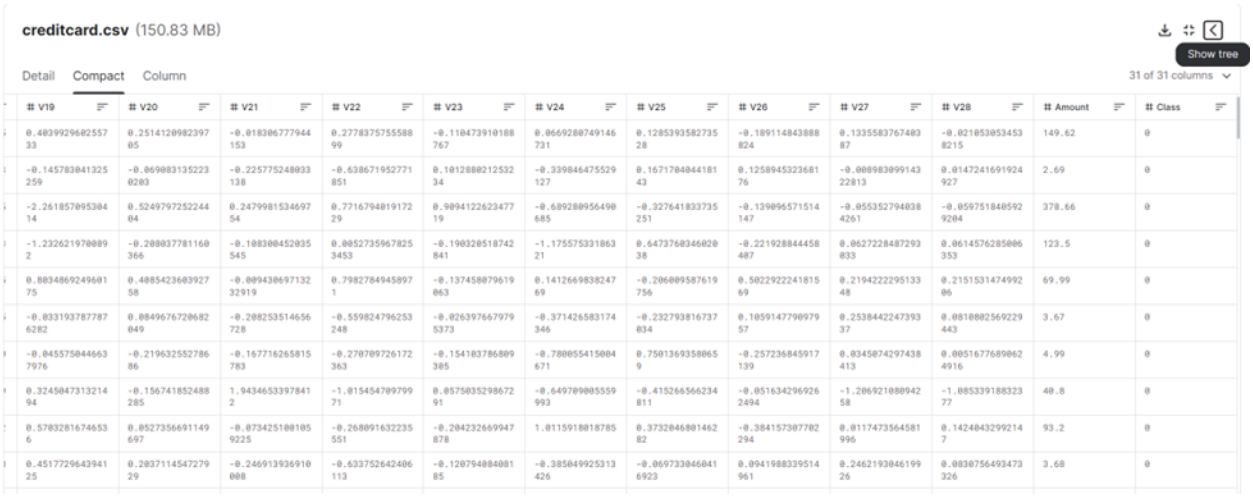
creditcard.csv (150.83 MB)

Detail | Compact | Column

31 of 31 columns

Show tree

# Time	# V1	# V2	# V3	# V4	# V5	# V6	# V7	# V8	# V9	# V10	# V11
0	-1.3598071336738	-0.07278117330898497	2.536346737969144	1.378155224274443	-0.3383207699425518	0.46238777776229292	0.2395985548061257	0.09869790126107587	0.3637869696112313	0.09079417197893316	-0.5515995332608813
0	1.19185711131486	0.26615071205963	0.16648011335321	0.44815407846091	0.06001764928222243	-0.08236080881556687	-0.07880298333231113	0.08510165491481184	-0.2554251281091186	-0.16697441400614	1.612726661085479
1	-1.3583540615982323	-1.3401630747360909	1.773209342631119	0.37977959303432828	-0.5031981333181193	1.80049938079263	0.7914609564504022	0.24767578658899191	-1.5146543226058383	0.20764286521669696	0.624581459424895
1	-0.9662717115720887	-0.1852260088828898	1.792993339578772	-0.8632912750364530	-0.01030887960830823	1.24720316752486	0.2376089397717878	0.37743587465226262	-1.3870248627019797	-0.05495192247137749	-0.2264872638354016
2	-1.1582330934952323	0.87773675484845151	1.548717846511	0.40303393395512121	-0.4071933773116533	0.09592146246842256	0.5929407453855545	-0.2705326771922822	0.8177393882352944	0.7530744319763544	-0.8228428779463633
2	-0.4259658044124544	0.96052304488298585	1.141109342322219	-0.1682520797603022	0.42098688077219	-0.02972755166397742	0.4762009487200027	0.26031433307487474	-0.5686713757125151	-0.3714071968344471	1.341261980019577
4	1.22965763450793	0.14100350704932626	0.04537077358994491	1.2026127367359425	0.19188098859764598	0.27270812289909898	-0.005159002882509830	0.08121293988308940	0.4649599947838686	-0.09925432112892237	-1.41690724314928
7	-0.6442694423481465	1.41796354547385	1.074380376355601	-0.4921990184950150	0.94893409476415757	0.42811846283308989	1.120631358383533	-3.807864238735899	0.6153747306670276	1.24937617815176	-0.6194677961219139
7	-0.8942860822028244	0.28615719627654471	-0.1131922127298718	-0.2715261300886040	2.66959865959863	3.72181806112751	0.37014512767691616	0.85108444320090505	-0.3920475867986804	-0.4104304328484390	-0.7051165866465365
9	-0.3382617524257556	1.119593337641566	1.044366551573166	-0.2221872767382960	0.49936088649727	-0.24676110061991	0.6515832064899726	0.06953858651863870	-0.7367273163641090	-0.3668456392065412	1.017614467832622



creditcard.csv (150.83 MB)

Show tree

Detail Compact Column

31 of 31 columns

# V19	# V20	# V21	# V22	# V23	# V24	# V25	# V26	# V27	# V28	# Amount	# Class
0.4039929602557 33	0.2514120982397 05	-0.018306777944 153	0.2778375755588 99	-0.110473910188 767	0.0669280749146 731	0.1285393582735 28	-0.189114843888 824	0.1335583767483 87	-0.021053053453 8215	149.62	0
-0.145783041325 259	-0.069083135223 0203	-0.225775248033 138	-0.638671952771 851	0.1012880212532 34	-0.339846475529 127	0.1671704044181 43	0.1258945323681 76	-0.008983099143 22813	0.0147241691924 927	2.69	0
-2.261857095304 14	0.5249797252244 04	0.2479981534697 54	0.7716794819172 29	0.9094122623477 19	-0.689280956490 685	-0.327641833735 251	-0.139096571514 147	-0.055352794038 4261	-0.059751840592 9204	378.66	0
-1.232621970089 2	-0.208037781160 366	-0.108300452035 545	0.0052735967825 3453	-0.190320518742 841	-1.175575331863 21	0.6473760346020 38	-0.221928844458 487	0.0627228487293 033	0.0614576285006 353	123.5	0
0.8034869249681 75	0.4885423683927 58	-0.009430697132 32919	0.7982784945897 1	-0.137458079619 063	0.1412669838247 69	-0.206009587619 756	0.5022922241815 69	0.2194222295133 48	0.2151531474992 06	69.99	0
-0.033193787787 6282	0.0849676720682 049	-0.208253514656 728	-0.559824796253 248	-0.026397667979 5373	-0.371426583174 346	-0.232793816737 034	0.1059147790979 57	0.2538442247393 37	0.0810802569229 443	3.67	0
-0.045575044663 7976	-0.219632552786 86	-0.167716265815 783	-0.270709726172 363	-0.154103786809 385	-0.780055415004 671	0.7501369358065 9	-0.257236845917 139	0.0345074297438 413	0.0051677689062 4916	4.99	0
0.3245047313214 94	-0.156741852488 285	1.9434653397841 2	-1.015454709799 71	0.0575035298672 91	-0.649709055559 993	-0.415266566234 811	-0.051634296926 2494	-1.206921080942 58	-1.085339188323 77	40.8	0
0.5703281674653 6	0.0527356691149 697	-0.073425180105 9225	-0.268091632235 551	-0.204232669947 878	1.0115918018785	0.3732046801462 82	-0.384157307782 294	0.0117473564581 996	0.1424043299214 7	93.2	0
0.4517729643941 25	0.2037114547279 29	-0.246913936910 008	-0.633752642406 113	-0.120794084081 85	-0.385049925313 426	-0.069733046041 6923	0.0941988339514 961	0.2462193046199 26	0.0830756493473 326	3.68	0

Machine Learning Models

Logistic Regression

Logistic Regression is a powerful statistical method commonly employed for binary classification tasks, such as predicting whether an email is spam or not, or, as in this project, determining whether a credit card transaction is fraudulent. Unlike linear regression, which predicts continuous outcomes, logistic regression models the probability that a given instance belongs to a particular category.

Mathematical Equation [2]:

The logistic regression model employs the logistic function (also known as the sigmoid function) to transform the linear combination of input features into a probability score between 0 and 1. The logistic function is represented by the equation:

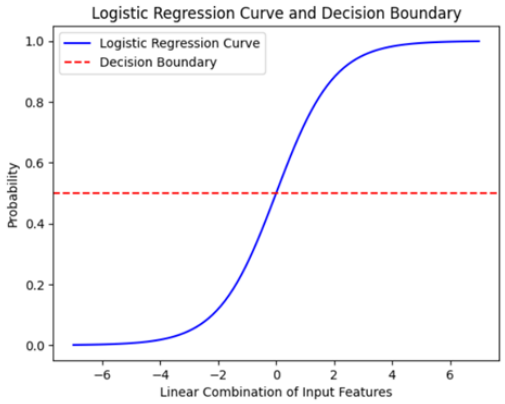
$$P(Y=1) = 1+ e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n)}$$

Here,

- P(Y=1) is the probability of the instance belonging to Class 1 (fraudulent in our case).
- e is the base of the natural logarithm.
- β_0 is the intercept.
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients associated with the input features $X_1, X_2, \ldots, X_n$.

Graphical Representation:

The sigmoid function generates an S-shaped curve, mapping any real-valued number to the range [0, 1]. This curve is essential for logistic regression, as it allows the model to output probabilities. The logistic regression decision boundary is set where the sigmoid function equals 0.5. Instances falling on one side of this boundary are predicted as Class 1, while those on the other side are predicted as Class 0.



In the graph, the x-axis represents the linear combination of input features, and the y-axis represents the predicted probability. As the linear combination increases, the probability

approaches 1, and as it decreases, the probability approaches 0. The decision boundary (shown as a vertical line) determines the threshold for classification.

Model Training

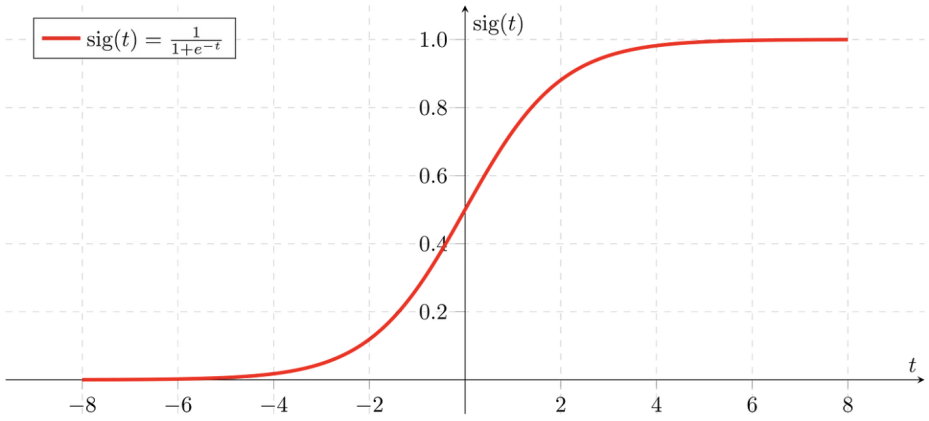
Trained on a dataset with fraudulent and non-fraudulent transactions, Logistic Regression learns patterns using a sigmoid function. Coefficients and the function's output help set a decision boundary, making it effective for binary outcomes like credit card fraud detection. The model's interpretability and suitability for this scenario contribute to its valuable role in the project.

LSTM(Long Short Term Memory)

LSTM is a recurrent neural network (RNN) design primarily used for tasks involving sequence prediction problems. The single hidden state of a traditional RNN can make it challenging for the network to learn long-term dependencies. Long short term memory networks (LSTMs) solve this issue by adding a memory cell, which is a container that can store information for a long time. They can learn long-term dependencies in sequential data, which makes them a good fit for tasks like price prediction, speech recognition, language translation, and time series forecasting.

The three gates, the input, forget, and output gates, control the memory cell. This memory cell's input, output, and addition of data is determined by these gates. The data added to the memory cell is managed by the input gate. What information is taken out of the memory cell is managed by the forget gate. Information that is output from the memory cell is also controlled by the output gate. As a result, long-term dependencies can be learned by LSTM networks, which are able to selectively store or discard information as it passes across the network**[5].**

Sigmoid Activation Function with Graph



LSTM gates are the sigmoid activation functions; that is, they output a value between 0 and 1, which is often one of the two. Because we want a gate to only produce positive values and to be able to clearly indicate whether we should maintain a certain feature or discard it, we use the sigmoid function for gates. "0" indicates that nothing can pass through the gates. A gate that reads "1" indicates that everything can pass through it **[6]**.

LSTM Model Architecture [6]

The chain structure of the LSTM architecture is made up of neural networks and various memory building blocks known as cells. The gates perform memory manipulations, while the cells store information. Three gates are present in LSTM:

Forget Gate

The forget gate is used to remove information that is no longer useful in the cell state. It takes two inputs, x_t (input at that specific time) and h_{t-1} (previous cell output), multiplies them with weight matrices, and adds bias. The resulting product is then passed through an activation function that produces a binary output: if the output is 0, the piece of information is forgotten for that specific cell state, if the output is 1, it is kept for future use. The equation for forget gate is:

$$\mathbf{f_t = s\ (w_f\ [h_{t-1},\ x_t] + b_f)}$$

Input Gate

The input gate modifies the cell state by adding pertinent information. Using inputs h_{t-1} and x_t, the sigmoid function is first used to regulate the information before filtering the values to be remembered in a manner akin to a forget gate. Next, a vector containing all possible values from h_{t-1} and x_t is created using the tanh function, which produces an output ranging from -1 to +1. Finally, the useful information is obtained by multiplying the vector values by the regulated values. The equation for the input gate is:

$$\mathbf{i_t = s\ (w_i\ [h_t\text{-}1,\ x_t] + b_i)}$$

Then the previous state is multiplied by f_t, ignoring the data that was previously decided to ignore. Then $i_t * C_t$ is included. This is a representation of the updated candidate values, with each state value's amount of update considered.

$$\mathbf{c_t = tanh(w_c\ [h_t\text{-}1,\ x_t] + b_c}$$

$$\mathbf{c_t = f_t\ *\ c_{t-1} + i_t\ *\ c_t}$$

$$\mathbf{h_t = o_t\ *\ tanh(c_t)}$$

Here, c_t denotes cell state(memory) and c_t refers to candidate for cell state.

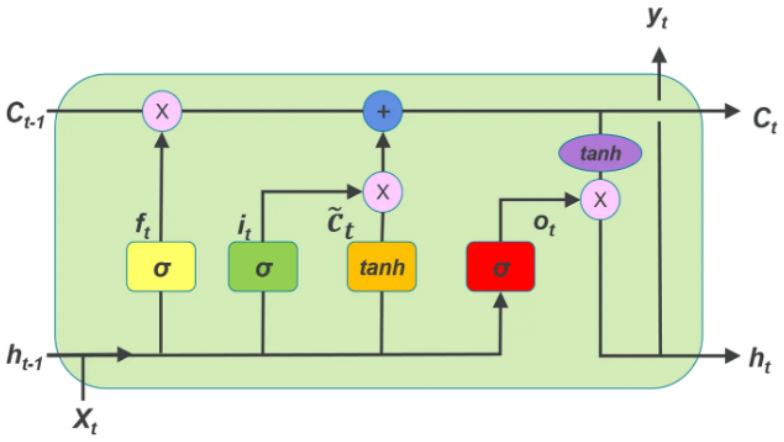
Output Gate

The output gate performs the function of extracting valuable information to be displayed as output from the current cell state. Using the tanh function on the cell, a vector is first created. Next, the data is filtered by the values that need to be remembered using inputs h_{t-1} and x_t, and the information is regulated using the sigmoid function. The vector's values and the regulated values are finally multiplied before being sent as an output and input to the following cell. The equation for output gate is:

$$\mathbf{o_t = s\ (w_o\ [h_t\text{-}1,\ x_t] + b_o)}$$

Here, h_{t-1} denotes the output of the previous lstm block at t-1 time. x_t refers to input at current timestamp, b_x refers to biases for all the gates and w_x refers to the weight for all the gates. And s is the sigmoid function.

Below is the figure of a single LSTM block:



To control the information flow between 0 and 1, each gate has sigmoid activation functions that help it decide what to keep and what to discard. In addition, the hyperbolic tangent (tanh) activation function is used to modify the cell state. This function guarantees that values fall between -1 and 1, enabling more stable updates to the cell state. With this architecture, information can be updated or selectively retained over long sequences, allowing LSTMs to capture long-range dependencies in sequential data efficiently.

An LSTM Neural Network is constructed using TensorFlow's Keras API. The model consists of an LSTM layer with 100 units, followed by a Dropout layer with a dropout rate of 0.5 to prevent overfitting. The output layer is a dense layer with a sigmoid activation function, suitable for binary classification problems. The model is compiled using binary cross-entropy as the loss function and the Adam optimizer with a learning rate of 0.001. The accuracy metric is used for model evaluation

Training the Model

The LSTM model is trained on the preprocessed data using a training set, and the performance is evaluated on a validation set. The training process involves 10 epochs with a batch size of 64.

Random Forest [3]

Overview: In our credit card fraud detection project, we have employed the Random Forest model as one of the key machine learning algorithms. Random Forest is an ensemble learning method known for its effectiveness in classification tasks. Here's a concise overview of its role and characteristics in our project.

Ensemble Learning: Random Forest operates by creating multiple decision trees during the training phase. Each decision tree is constructed using a different subset of the dataset and a random subset of features. The final prediction is then determined by aggregating the individual predictions of these trees.

Decision Trees in Random Forest: The fundamental building block of Random Forest is the decision tree. These trees are simple models that make decisions based on a series of rules. In our project, decision trees are trained to discern patterns in credit card transactions, distinguishing between legitimate and fraudulent activities.

Bagging Technique: Random Forest employs a technique called bagging (Bootstrap Aggregating). This involves training each decision tree on a random subset of the dataset, allowing for diversity among the trees. This diversity helps improve the model's robustness and generalization to unseen data.

Feature Importance: One of the notable features of Random Forest is its ability to provide a measure of feature importance. This indicates the relevance of each feature in making predictions. For credit card fraud detection, understanding which features contribute most to the model's decision-making process is crucial.

Model Training: During the training phase, the Random Forest model is fed with our credit card transaction dataset. This dataset includes features derived from Principal Component Analysis (PCA) transformation, along with 'Time' and 'Amount' attributes. The model is trained to recognize patterns and intricacies associated with fraudulent transactions.

Comparison of the Models

Comparing the Accuracy

When comparing our models, we evaluated their performance using these key metrics: Accuracy, Precision, Recall, F1 Score, and Matthews Correlation Coefficient (MCC). Accuracy gives an overall measure, Precision gauges the accuracy of positive predictions, Recall assesses sensitivity to detecting true positives, F1 Score balances Precision and Recall, and MCC offers a comprehensive correlation measure. Together, these metrics guide our assessment, ensuring a thorough evaluation of the models in the context of credit card fraud detection.

Scores	Logistic Regression	Long Short Term Memory	Random Forest
Accuracy	0.99892	0.99956	0.99959
Precision	0.81356	0.96202	0.97468
Recall	0.48979	0.77551	0.78571
F1 Score	0.61166	0.85875	0.87005
Matthew Correlation Coefficient	0.63078	0.86354	0.87492

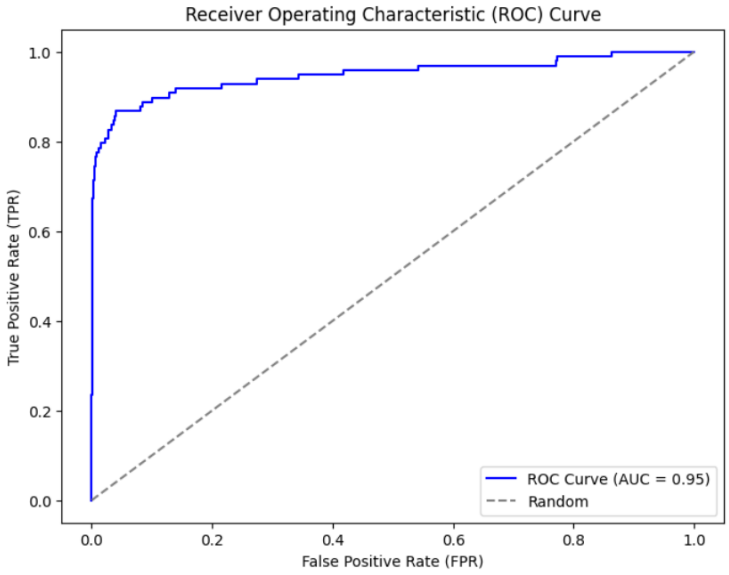
In comparing the accuracy of the three models based on the output values on the table above, it is evident that Random Forest outperforms both Logistic Regression and Long Short Term Memory (LSTM) across various metrics. Random Forest exhibits the highest accuracy, precision, recall, F1 Score, and Matthews Correlation Coefficient among the models, with values of 0.99959, 0.97468, 0.78571, 0.87005, and 0.87492, respectively. This indicates a robust performance in correctly classifying both fraudulent and non-fraudulent transactions. While LSTM follows

closely with competitive scores, Logistic Regression lags behind, particularly in recall, where it demonstrates the lowest sensitivity to fraudulent transactions. The superiority of Random Forest in accuracy metrics suggests its effectiveness in credit card fraud detection, making it a promising choice for real-time electronic transaction security.

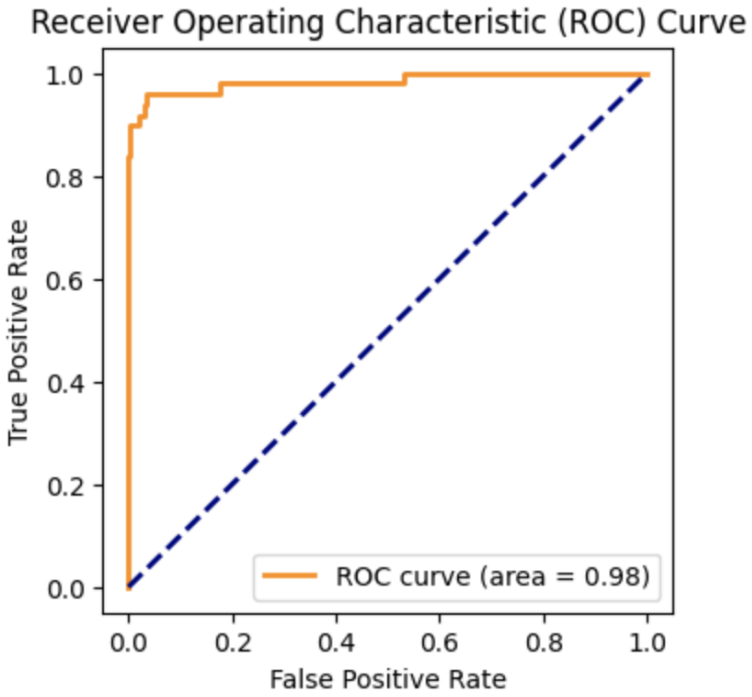
Comparing the Matrices & Graphs

Receiver Operating Characteristics (ROC Curve)

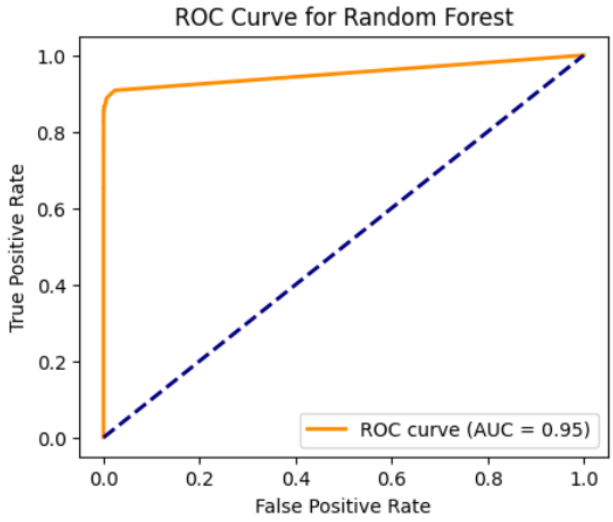
Logistic Regression



Long Short Term Memory



Random Forest



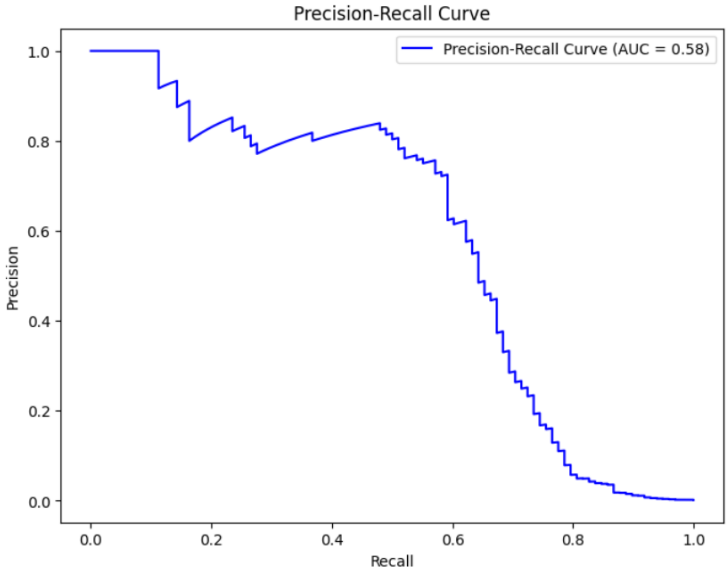
The Receiver Operating Characteristic (ROC) curve assesses how well a binary classification model distinguishes between positive and negative instances across different threshold levels. The Area Under the Curve (AUC) summarizes this performance. From our output:

- **Logistic Regression:** AUC = 0.95 - Good ability to distinguish with a balanced true positive rate and false positive rate.
- **LSTM:** AUC = 0.98 - Excellent discrimination, superior to Logistic Regression, indicating strong performance.
- **Random Forest:** AUC = 0.95 - Robust discrimination, similar to Logistic Regression.

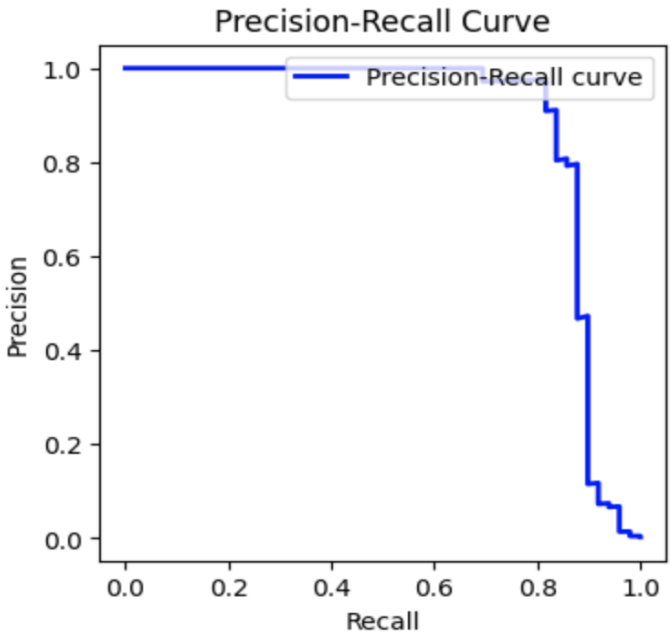
All models exhibit reasonably strong performance, with LSTM standing out for superior discrimination.

Precision-Recall Curve:

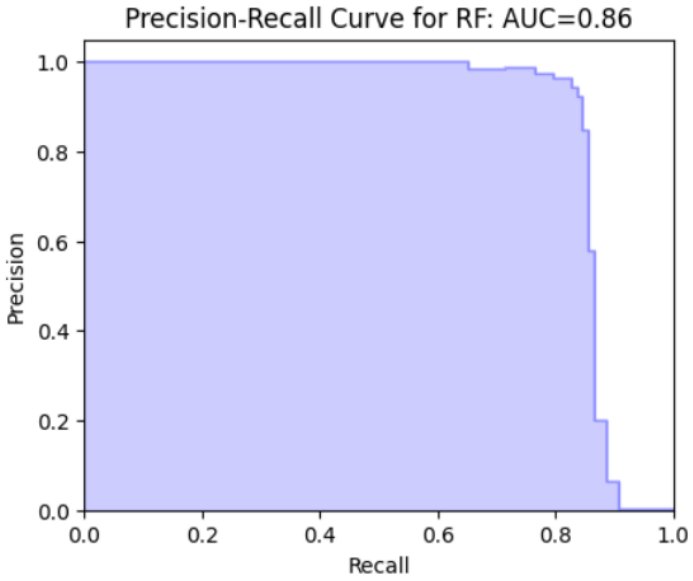
Logistic Regression



Long Short Term Memory



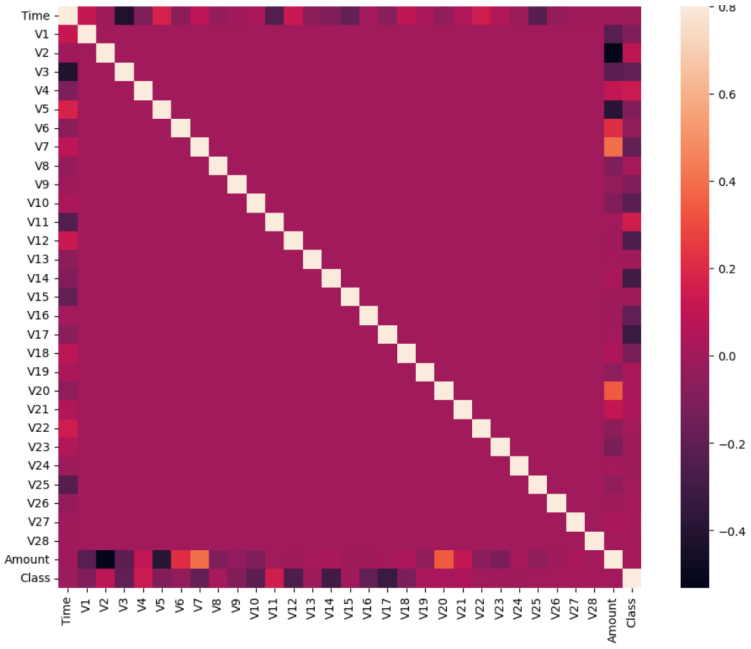
Random Forest



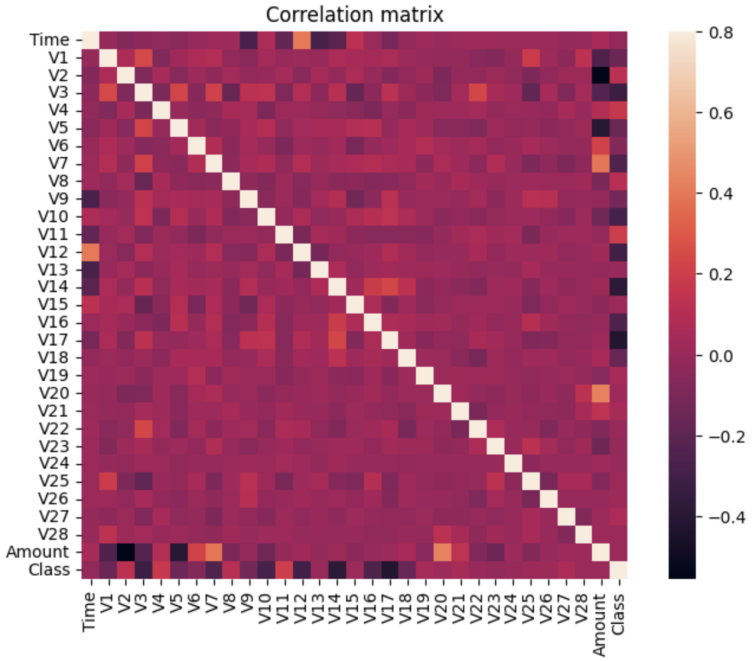
The Precision-Recall (PR) curve evaluates a binary classification model's precision and recall across various threshold levels, providing insights into its ability to correctly identify positive instances while minimizing false positives. The Area Under the Curve (AUC) summarizes the PR curve's performance. From the outputs above, we can see how LSTM and Random Forest show strong and robust precision-recall characteristics while Logistic Regression lags behind.

Correlation Matrix (Heatmap):

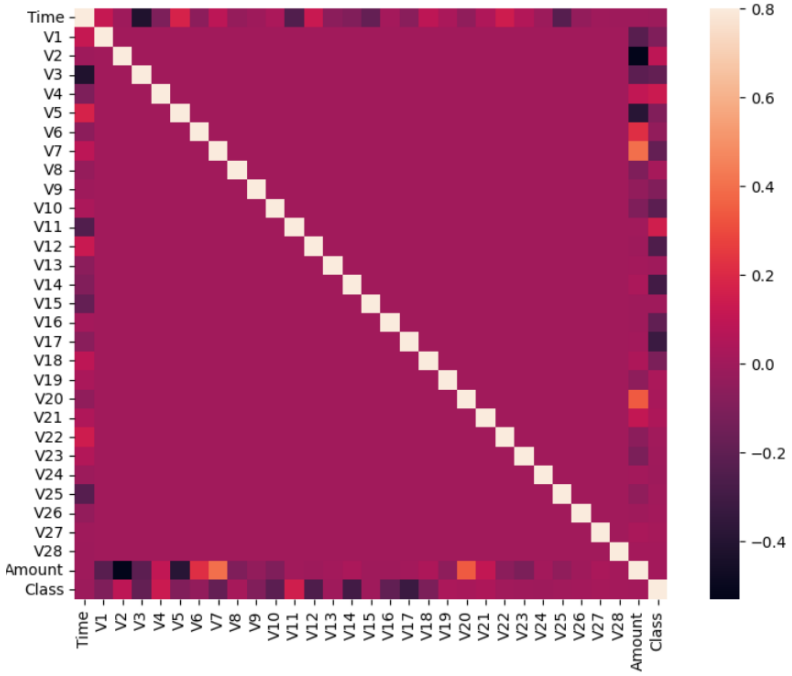
Logistic Regression



Long Short Term Memory



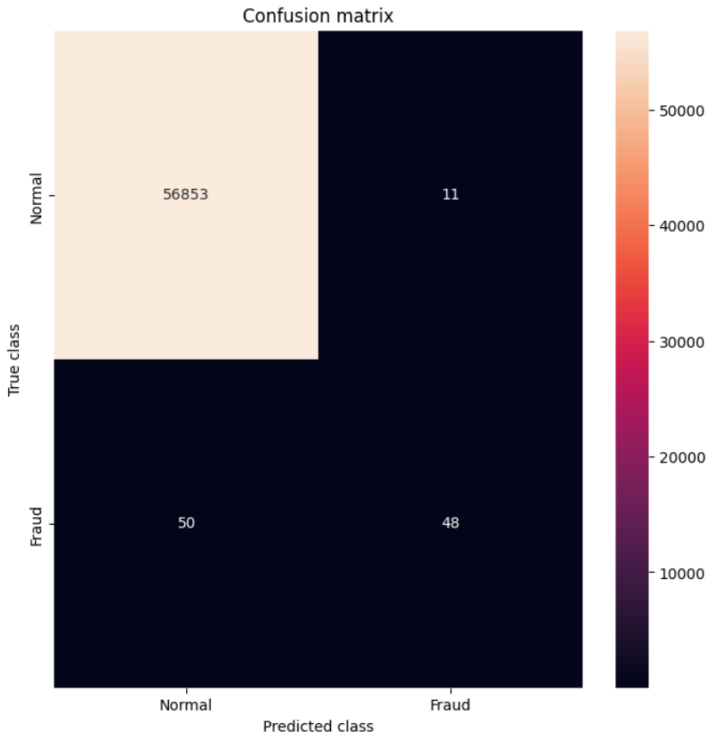
Random Forest



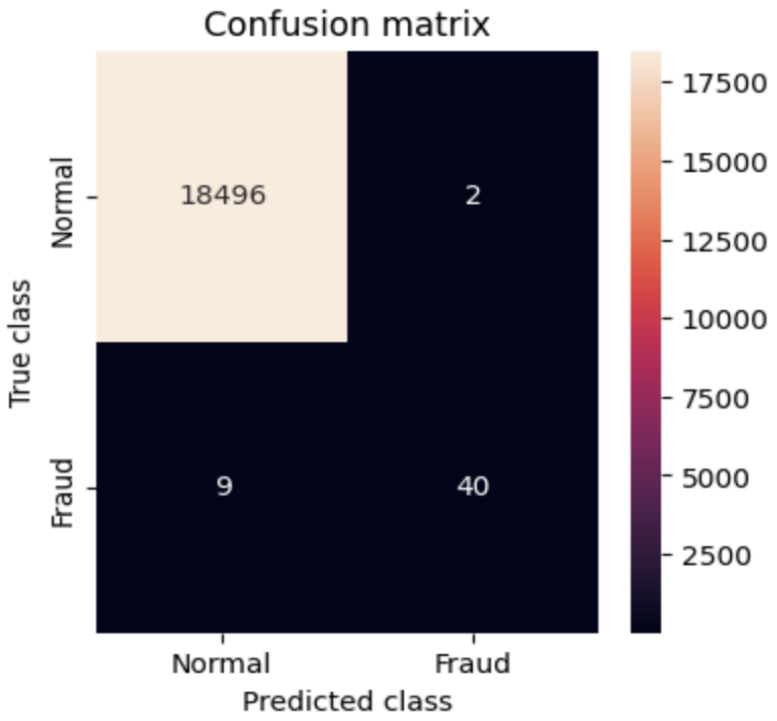
The correlation matrix is like a map that shows how different features in our fraud detection data relate to each other. Each entry in the matrix tells us the strength and direction of the relationship between two features. If the number is close to 1, it means they move together in the same direction, and if it's close to -1, it means they move in opposite directions. The correlation of 0 suggests no linear relationship. By examining the correlation matrices of the three models, we observe that the Logistic Regression and Random Forest models exhibit minimal correlation with other features. However, in the LSTM model, there is a more noticeable correlation between features.

Confusion Matrix:

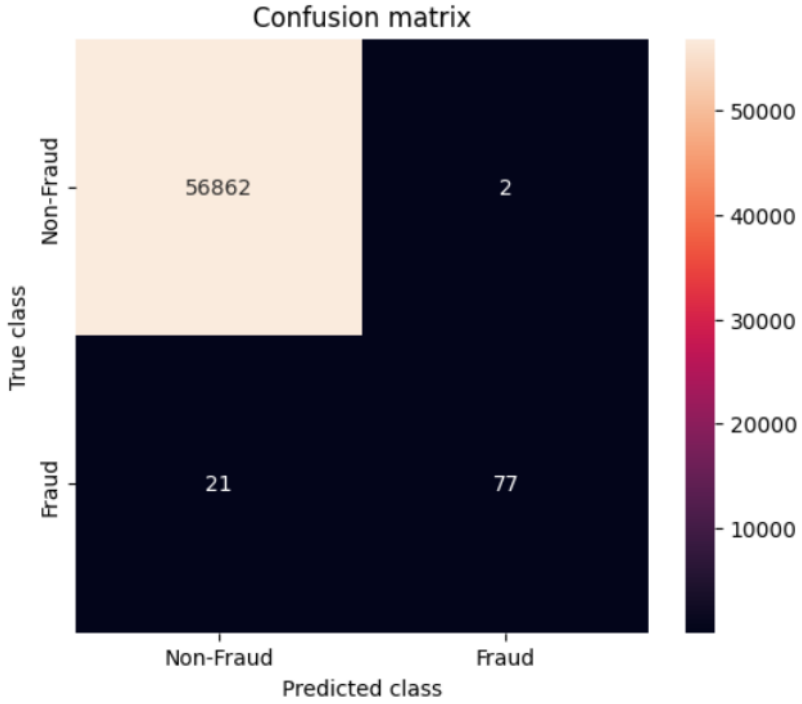
Logistic Regression



Long Short Term Memory



Random Forest



Model Names	True Negative (Non Fraud)	False Positive	False Negative	True Positive (Fraud)
Logistic Regression	**56853**	**11**	**50**	**48**
Long Short Term Memory	**18496**	**2**	**9**	**40**
Random Forest	**56869**	**2**	**21**	**77**

From the values above from the confusion matrix we can compare the efficiency of our models like below:

Logistic Regression: Logistic Regression demonstrates a good balance between correctly identifying non-fraudulent transactions (True Negatives) and correctly identifying fraudulent transactions (True Positives). However, as seen above, it has the highest number of false positives and false negatives compared to the other two models, indicating potential room for improvement.

Long Short Term Memory (LSTM): LSTM exhibits strong performance in correctly classifying non-fraudulent transactions (high True Negatives) and effectively identifying fraudulent transactions (high True Positives). The low values for false positives and false negatives suggest a robust ability to discriminate between the classes.

Random Forest: Random Forest, similar to Logistic Regression, maintains a balance between True Negatives and True Positives. It has a lower number of false positives and lower number of false negatives compared to Logistic Regression and a higher number of true positives. This suggests that Random Forest may be more sensitive to fraudulent transactions but generates fewer false alarms compared to Logistic Regression.

In summary, each model has its strengths and weaknesses. Logistic Regression and Random Forest demonstrate a balanced performance, while LSTM excels in accurately identifying both non-fraudulent and fraudulent transactions with fewer false predictions. The choice of the most suitable model depends on the specific requirements and trade-offs desired for the credit card fraud detection system.

Challenges

Building this project involves overcoming several challenges:

- **Imbalanced Data:** The dataset is heavily imbalanced, with only 0.2% of transactions labeled as fraudulent and 99.8% as non-fraudulent. This imbalance poses a significant challenge for the machine learning models leading to a bias towards predicting the majority class. Traditional metrics like accuracy may not be reliable, so we use precision, recall, and F1 score for better evaluation **[4]**. F1 score, considering both precision and recall, offers a nuanced view, crucial for accurately predicting the minority class alongside the majority.
- **Interpretability vs. Complexity:** Striking a balance between the interpretability of the model and its complexity proved to be a critical challenge. Achieving both a clear understanding of the model's decisions and high accuracy required thoughtful consideration and exploration.
- **Threshold Setting:** Determining the optimal threshold for classifying transactions played a vital role in influencing precision, recall, and overall model performance. Finding the right balance was essential to ensure the model's effectiveness in identifying fraud without generating excessive false positives.

- **Ethical and Regulatory Compliance:** Navigating ethical and regulatory considerations when handling sensitive financial data introduced complexity to the model development process. The dataset utilized in this project underwent an anonymized Principal Component Analysis (PCA) transformation for the features data which ensured privacy laws.
- **Adaptability to Fraud Tactics:** Constructing a model capable of adapting to evolving fraud tactics presented an ongoing challenge. Continuous monitoring and updates will be necessary to keep the model effective in the face of emerging fraud patterns and techniques.

Conclusion

In summary, our project aimed to detect credit card fraud using three models: Logistic Regression, LSTM, and Random Forest. After thorough comparison, Random Forest emerged as the most effective, surpassing the others in accuracy. We tackled challenges like imbalanced data and interpretability, emphasizing precision and recall. While each model had its strengths, Random Forest, with its ensemble approach, proved highly efficient in distinguishing between legitimate and fraudulent transactions. Ongoing monitoring is crucial for adapting to evolving fraud tactics. This study contributes valuable insights for developing strong, adaptive models to enhance electronic transaction security.

References

1. Machine Learning Group - ULB, “Credit Card Fraud Detection,” Kaggle, March 23, 2018, https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data.
2. “Sigmoid Function.” Wikipedia, November 27, 2023. https://en.wikipedia.org/wiki/Sigmoid_function.
3. Melcher, Kathrin, and Rosaria Silipo. “Fraud Detection Using Random Forest, Neural Autoencoder, and Isolation Forest Techniques.” *InfoQ*, InfoQ, 14 Aug. 2019, www.infoq.com/articles/fraud-detection-random-forest/.
4. Mazumder, Saikat. “5 Techniques to Handle Imbalanced Data for a Classification Problem.” *Analytics Vidhya*, 27 Sept. 2023, www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/.
5. https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/
6. https://medium.com/@divyanshu132/lstm-and-its-equations-5ee9246d04af