Project Proposal

Introduction:

In today's rapidly evolving financial landscape, with the popularization of digital payments, credit card fraud is a significant concern. According to the 2023 Credit Card Fraud Report, 65% of credit card holders in the US have been victims of credit card fraud. Previous research on fraudulent credit transactions made use of several machine learning algorithms including Logistic Regression, Random Forest, naive bayes, and Artificial Neural Networks (ANNs) (Dornadula and Geetha, Ileberi et al.). Logistic Regression determines the probability of fraudulent transactions through a linear function (Ileberi et al.) whereas Random Forest combines Decision Trees to determine fraudulence based on majority classification. Naive Bayes assumes feature independence to estimate fraud likelihood (Ileberi et. al.). Sulaiman et al. proposes ANNs, which involves layered nodes for precise classification in a privacy-focused federated learning framework and genetic algorithms aid in feature selection to enhance model accuracy (Ileberi et. al.).

We used a Credit Card Fraud Kaggle dataset which included the following transaction features:

Numeric features:

- distance_from_home - the distance from home where the transaction happened.
- distance_from_last_transaction - the distance from where the last transaction happened.
- ratio_to_median_purchase_price - Ratio of purchased price transaction to median purchase price.

Binary features:

- repeat_retailer - if a transaction has happened from this retailer with this card previously.
- used_chip - if the transaction through chip (credit card)
- used_pin_number - Is the transaction happened by using PIN number.
- online_order - Is the transaction an online order.
- fraud - Is the transaction fraudulent.

We created a model to predict the fraud label based on the remaining features.

Problem Definition:

Scammers are continually adapting to advances in fraud detection systems. Therefore, it remains highly challenging to differentiate between legitimate and fraudulent credit transactions. We aim to develop a model that can predict the authenticity of a transaction with high accuracy based on a wide variety of features.

Methods:

In the implementation of two models for a classification task—Decision Tree (DT) and Deep Neural Network (DNN)—each follows distinct strategies.

Data Preprocessing

We used pandas to retrieve useful information about our dataset features and ensure that all data points are complete. As shown below, there were not any null values in our dataset, so we did not have to do additional processing to deal with incomplete datapoints. Additionally, all of the data was in numeric format, with binary variables represented by 0 or 1, so we did not have to convert textual values to numbers for training the model.

We utilized data visualization to gain a better understanding of our features so that we could decide how to proceed with model implementation. We created a heat map of the correlation matrix to understand the relationships between all the features. As shown below, the ratio of the transaction amount to the cardholder's mean purchase price had the highest correlation of 0.46 to fraud. Other features like distance of the transaction from the cardholder's home address and whether the order was an online order also showed relatively higher correlations of 0.19 to fraud. The features themselves were not correlated, except for distance_from_home and repeat_retailer, which had a correlation of 0.14.

Model Implementation - Decision Tree

We chose the Decision Tree model for its simplicity and intuitive decision-making approach. We started our classification model implementation with a Decision Tree, since it does not require normalization or scaling of data, to get a sense of how easily the data can be classified into the two classes: fraud or not fraud. It works by creating a tree-like model of decisions, an intuitive method that mimics human decision-making.

We attended to data preprocessing to make sure that the given dataset was omitted of blank spaces and all the features were in reasonable range. After loading the dataset in a Pandas DataFrame, we performed stratified random sampling, where we split the dataset into test and training subsets. Since the dataset was highly imbalanced in the sense that less than 10% of the data points represented fraudulent transactions and the remaining were legitimate, we used stratification to ensure that a proportionate amount of each class is used for training and to prevent the model from training on only legitimate transactions. We allocated 20% of the data for testing and we made sure to shuffle the data points so we ensure a more robust subset. After splitting the data, we use the MinMaxScaler to normalize the values for the features. Given sensitive inputs, this technique scales all feature values between 0 and 1 to account for any overly sensitive data values. After we scale the training set accordingly, we transform both the training and testing sets to make sure these are suitable for the given model. We then trained the model using scikit-learn's "DecisionTreeClassifier" and evaluated on metrics like accuracy and precision. Challenges like overfitting and imbalanced data were addressed by controlling tree depth and preprocessing techniques, respectively.

Model Implementation - DNN

The DNN model provides a more sophisticated approach, capable of capturing complex patterns in the data. It's particularly effective in scenarios where relationships between variables are non-linear and intricate.

We preprocessed the data by splitting it into training, validation, and test sets, standardizing features, ensuring class distribution with stratify, and calculating class weights for handling imbalanced classes. The steps we took to preprocess the data for this model include splitting the data set into three sets, one for training, the other for validation, and one for testing. The train_test_split method splits the dataset into the aforementioned sets. We also passed in a consistent percentage for fraud for each of the sets. This avoids biases and imbalances in the dataset. Additionally, we standardized the features of the dataset using StandardScalar and stratify ensures that the class distribution is maintained. Lastly, the class weights are calculated to ensure that imbalanced classes are properly handled. When the model is able to accurately predict that a transaction is not fraud, it still maintains a high accuracy. However, when a transaction is misclassified as fraud, the model is forced to pay more attention to these instances to improve its accuracy.

Compared to DT, the DNN model is more sophisticated, ideal for capturing complex data patterns with non-linear variable relationships. Its process includes feature standardization, dividing the dataset into training, validation, and testing sets, and constructing a multi-layer architecture with Dense and Dropout layers. Class weights are used to counteract data imbalance. The model is trained with the "adam" optimizer and assessed using similar metrics as the DT.

Model Implementation - SVM

We also chose to utilize the Support Vector Machine (SVM) model, as they are generally effective in high dimensional spaces and we are working with a dataset that has a large number of features. SVM essentially tries to find the hyperplane that separates the data into the appropriate classes and maximizes the margin, and thus it is less susceptible to overfitting.

Similar to the previous two models, we preprocessed the data by splitting it into training and test sets, standardizing features, ensuring class distribution with stratify, and calculating class weights for handling imbalanced classes. The steps we took to preprocess the data for this model include splitting the data set into two sets, one for training and one for testing using the train_test_split method. We passed in a consistent percentage for fraud for each of the sets to prevent biases and imbalances in our model training and testing. We standardized the features of the dataset using StandardScalar and stratify ensures that the class distribution is maintained. We then trained the model using scikit-learn's "SVC" and evaluated it on several metrics.

Results and Discussion:

Discussion

As stated in the model results and as shown by the confusion matrix, the Decision Tree model had a very high accuracy of 99.9975% in predicting credit card fraud based on the other features. Recall is of specific interest in fraud detection cases, as it represents the number of correctly identified fraudulent cases out of the total number of true positives. Our Decision Tree-based model correctly flagged 99.9827% of fraudulent credit card transactions in our test dataset, which is especially significant to this context as it is crucial that fraudulent transactions do not get marked as legitimate. As shown by the confusion matrix, only 3 out of 17,388 fraudulent transactions were incorrectly labeled as legitimate. Precision, representing the ratio of correctly identified fraudulent cases out of the total number of cases marked as fraudulent, is also of particular interest to ensure that we do not mistakenly flag a large proportion of legitimate cases as fraud. This model had a very high precision rate of 99.9885%. As shown by the confusion matrix, only 2 out of 17,387 cases predicted to be fraud were actually legitimate, thus our model has a low incidence of falsely alarming people of credit card fraud. Thus, our model had impressive accuracy, recall, and precision, which is indicated by the high F1 Score and attests to the validity of our model.

Model 2: DNN - Deep Neural Network

Discussion

As stated in the model results and as shown by the confusion matrix, the DNN model had a very high accuracy of 98.4275% in predicting credit card fraud based on the other features. As stated earlier, recall is of specific interest in fraud detection cases, as it represents the number of correctly identified fraudulent cases out of the total number of true positives. Our DNN-based model correctly flagged 82.0777% of fraudulent credit card transactions in our test dataset, which is especially significant to this context as it is crucial that fraudulent transactions do not get marked as legitimate. Though this is a high percentage, it did not perform as well as the Decision Tree. As shown by the confusion matrix, 3,133 out of 17,388 fraudulent transactions were incorrectly labeled as legitimate. Thus, a large number of fraudulent transactions were not flagged, which is a significant problem considering our motive is to effectively alert cardholders if a fraudulent transaction was made so that they can quickly take action. Precision, representing the ratio of correctly identified fraudulent cases out of the total number of cases marked as fraudulent, is also of particular interest to ensure that we do not mistakenly flag a large proportion of legitimate cases as fraud. This model had a very high precision rate of 99.9164%. As shown by the confusion matrix, only 12 out of 14,360 cases predicted to be fraud were actually legitimate, thus our model has a relatively low incidence of falsely alarming people of credit card fraud. Thus, our model had very high accuracy and precision, contributing to the high F1 Score, but it needs to work on improving recall.

Model 3: SVM - Support Vector Machine

Discussion

As stated in the model results and as shown by the confusion matrix, the SVM model had a very high accuracy of 99.8710% in predicting credit card fraud based on the other features. As stated earlier, recall is of specific interest in fraud detection cases, as it represents the number of correctly

identified fraudulent cases out of the total number of true positives. Our SVM-based model correctly flagged 99.0598% of fraudulent credit card transactions in our test dataset, which is especially significant to this context as it is crucial that fraudulent transactions do not get marked as legitimate. Though this is much higher than the recall of the DNN-based model, it did not perform as well as the Decision Tree, though its performance was comparable. As shown by the confusion matrix, 160 out of 17,388 fraudulent transactions were incorrectly labeled as legitimate. Mislabeling fraudulent transactions is a significant problem considering our motive is to effectively alert cardholders if a fraudulent transaction was made so that they can quickly take action, but this model did correctly flag most of them. Precision, representing the ratio of correctly identified fraudulent cases out of the total number of cases marked as fraudulent, is also of particular interest to ensure that we do not mistakenly flag a large proportion of legitimate cases as fraud. This model had a very high precision rate of 99.4589%. As shown by the confusion matrix, only 94 out of 14,360 cases predicted to be fraud were actually legitimate, thus our model has a relatively low incidence of falsely alarming people of credit card fraud. Thus, our model had impressive accuracy, recall, and precision, which is indicated by the high F1 Score and attests to the validity of our model. Its performance was significantly better than the DNN-based model on this dataset and was comparable to the Decision Tree model, though the Decision Tree maintained the best performance across all metrics.

Conclusions:

The Decision Tree model and the SVM model performed much better than the DNN model for this dataset. Both Decision Tree and SVM reported ratios higher than 99% across all metrics: accuracy, precision, recall, and F1 score. In fact, Decision Tree reported all metrics over 99.9%, making it the model that most effectively classifies this dataset, which is very important in light of the real-world implications of misclassifying a credit card transaction. The DNN was effective in terms of accuracy and precision, but the significantly lower value for recall shows that it struggled with classifying a certain subset of the data. Though 82.0777% is still quite a high ratio, recall represents the percentage of fraudulent data points being correctly classified, and this means that 17.9223% of fraudulent transactions were labeled as legitimate. And with the millions of credit card transactions that happen around the globe every day, that 17.9223% of unflagged fraudulent transactions can turn into a very large number and thus does not help in solving the major problem of credit card fraud. Thus, based on our implementation and analysis, Decision Tree proved to be the best model for classifying whether a credit card transaction is fraudulent or not.

Sources:

- 2023 credit card fraud report. Security.org. (2023, May 22). https://www.security.org/digital-safety/credit-card-fraud-report/
- Bin Sulaiman, R., Schetinin, V. & Sant, P. Review of Machine Learning Approach on Credit Card Fraud Detection. Hum-Cent Intell Syst 2, 55–68 (2022). https://doi.org/10.1007/s44230-022-00004-0
- Dornadula, V. N. and Geetha, S. Credit Card Fraud Detection using Machine Learning Algorithms. Procedia Computer Science 165, 631–641 (2019). https://doi.org/10.1016/j.procs.2020.01.057

- Ileberi, E., Sun, Y. & Wang, Z. A machine learning based credit card fraud detection using the GA algorithm for feature selection. J Big Data 9, 24 (2022). https://doi.org/10.1186/s40537-022-00573-8

Contributions:

- Introduction: Shreya Malpani and Ishita Verma
- Problem Definition: Shreya Malpani and Ishita Verma
- M1 Implementation: Priyanka Mosur, Khushi Bhardwaj
- M2 Implementation: Priyanka Mosur, Khushi Bhardwaj, Amirita Manickandan, Ishita Verma, Shreya Malpani
- M3 Implementation: Priyanka Mosur, Khushi Bhardwaj, Amirita Manickandan
- Methods: Amirita Manickandan, Ishita Verma, Shreya Malpani
- Results & Discussion: Amirita Manickandan, Shreya Malpani, Ishita Verma
- Gantt Chart: Shreya Malpani and Ishita Verma
- GitHub Pages: Khushi Bhardwaj, Priyanka Mosur, Amirita Manickandan, Shreya Malpani

Team Member	Task(s)
Shreya Malpani	Introduction, Problem Definition, M2 Implementation, Results & Discussion, Gantt Chart, GitHub Pages
Ishita Verma	Introduction, Problem Definition, M2 Implementation, Methods, Gantt Chart
Khushi Bhardwaj	M1 Implementation, M2 Implementation, M3 Implementation, GitHub Pages
Amirita Manickandan	M2 Implementation, M3 Implementation, Methods, Results & Discussion, GitHub Pages
Priyanka Mosur	M1 Implementation, M2 Implementation, M3 Implementation, GitHub Pages