AMERICAN EXPRESS MAKETHON CHALLENGE
ML FRAUD DETECTION

Team- IIT Gandhinagar girls

Objective:

ML-based fraud detection aims to detect fraudulent activities in real time with minimal false positives and false negatives. This addresses the challenges of traditional rule-based systems that require manual updates and often fail to detect new fraud patterns, protecting businesses and individuals from financial and reputational damage.

In this project, we aim to classify a given credit transaction as fraudulent or valid based on various factors, including the time of the transaction. We have trained our model using logistic regression, simple classifying modes and neural network models. The dataset used for now is a credit transaction history of a bank which has time, amount and fraud status as known columns. The other columns' names are hidden for privacy reasons; Thus, these values are assumed to be scaled. We then test the learnt model on the test set and calculate the accuracies for the same.

Implementation:

This can be classified into 4 parts broadly:

1. Data Analysis-

To gain an understanding of our data, we need to conduct an initial analysis. However, due to privacy reasons, we only have information on the transaction amount, while the other columns, labelled V1 to V28, have been scaled and are unknown.

Our analysis shows that:

- The average transaction amount is around USD 88, indicating that most of the transactions are relatively small.
- We don't have any null values that need to be replaced.
- The dataset is heavily imbalanced, with fraud transactions accounting for only 0.17% of the total transactions. Thus, we must use appropriate techniques to address this imbalance in our dataset.
- Furthermore, the data description informs us that all features, except time and amount, have undergone a PCA transformation, which is a dimensionality reduction technique.

- The scaled V features are assumed to have been previously scaled for the PCA transformation.

Why does an imbalanced dataset cause a problem?

The original dataset is heavily skewed towards non-fraud transactions, indicating that most of the transactions recorded are not fraudulent. Using this imbalanced dataset as a basis for our predictive models and analysis may result in significant errors, as our algorithms may overfit and assume that most transactions are non-fraudulent. This is undesirable, as our aim is to build a model that can accurately detect patterns and signals of fraudulent activity.

2. Data preprocessing-

Scaling and sample making:

In this kernel phase, we will scale the Time and Amount columns and create a sub-sample of the data frame with equal amounts of fraud and non-fraud cases. This will help the algorithms better understand fraud patterns in transactions.

Here, our subsample data frame will have an equal amount of fraud and non-fraud transactions, creating a 50/50 ratio.

Why do we need to do this?

- Using the original imbalanced data frame can cause overfitting, as models will assume that frauds are rare.
- This can also result in wrong correlations between features and the outcome.
- It's important to have a balanced dataset to understand the true correlations and train the model to detect fraud accurately.

Splitting the dataset:

Before applying the Random UnderSampling technique, we need to separate the original data frame. This is because we want to test our models on the original testing set, not the testing set created by either undersampling or oversampling techniques. The aim is to fit the model with the undersampled or oversampled data frames to detect patterns and test it on the original testing set.

Summary:

- We have two scaled columns: scaled amount and scaled time.

- To create a new sub-sample with equal fraud and non-fraud cases, we randomly select 492 cases of non-fraud.
- We combine the 492 cases of fraud with the 492 cases of non-fraud to create a new sub-sample.
- We created a new data frame as we need the model to be tested on imbalanced cases. The balanced samples created would be fitted in the model using undersampling or oversampling to detect patterns and predict accordingly.

3. Performing Random UnderSampling and OverSampling:

In this project phase, we will use "Random Under Sampling" to balance our highly imbalanced dataset and prevent overfitting. Firstly, we will determine the class imbalance by using "value_counts()" on the class column to count the number of fraud transactions. We will then randomly select 492 non-fraud transactions to match the number of fraud transactions, creating a sub-sample with a 50/50 fraud-to-non-fraud ratio. To ensure the model's accuracy, we will shuffle the data. However, this technique has a significant disadvantage as it may cause information loss due to reducing the non-fraud transactions from 284,315 to 492.

We then performed various data analysis and preprocessing techniques on the new dataset:

- Correlation Matrices:
- Interquartile Range Method:
- Understanding t-SNE:

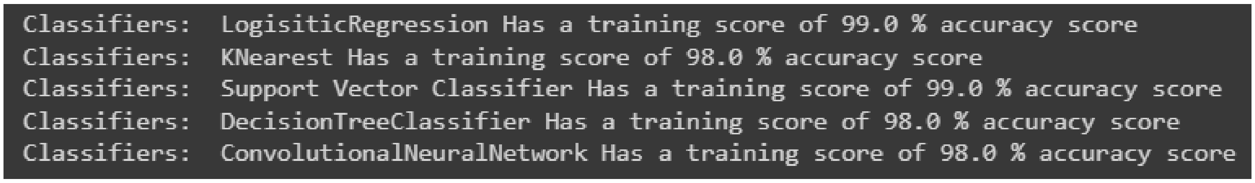
We then conducted classifier (underSampling) testing by training five types of classifiers and choosing the best.

Synthetic Minority Over-sampling Technique (SMOTE) (over-sampling) creates new synthetic points to have an equal balance of the classes.

4. Testing:

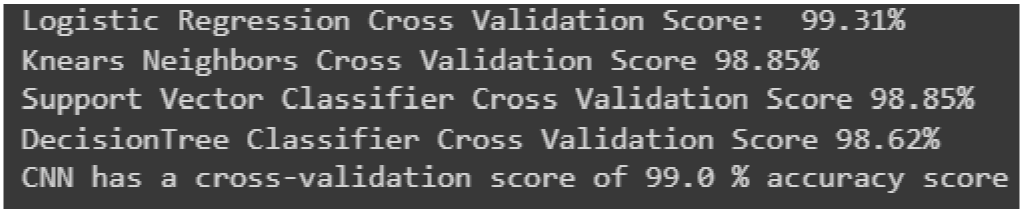
Due to time constraints, we couldnt test for over-sampling but the results as found for undersampling case are as followed:

Training accuracy:



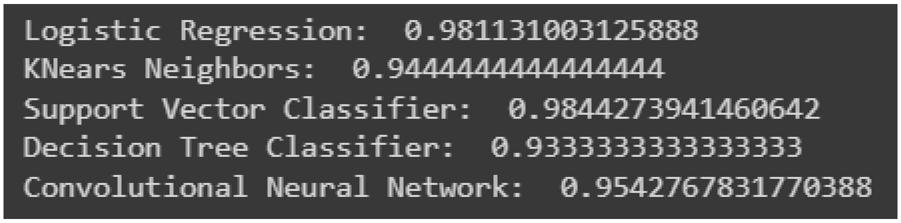
```
Classifiers:  LogisiticRegression Has a training score of 99.0 % accuracy score
Classifiers:  KNearest Has a training score of 98.0 % accuracy score
Classifiers:  Support Vector Classifier Has a training score of 99.0 % accuracy score
Classifiers:  DecisionTreeClassifier Has a training score of 98.0 % accuracy score
Classifiers:  ConvolutionalNeuralNetwork Has a training score of 98.0 % accuracy score
```

Training accuracy with cross validation:



```
Logistic Regression Cross Validation Score:  99.31%
Knears Neighbors Cross Validation Score 98.85%
Support Vector Classifier Cross Validation Score 98.85%
DecisionTree Classifier Cross Validation Score 98.62%
CNN has a cross-validation score of 99.0 % accuracy score
```

Roc-auc score:



```
Logistic Regression:  0.981131003125888
KNears Neighbors:  0.9444444444444444
Support Vector Classifier:  0.9844273941460642
Decision Tree Classifier:  0.9333333333333333
Convolutional Neural Network:  0.9542767831770388
```

5. Conclusion:

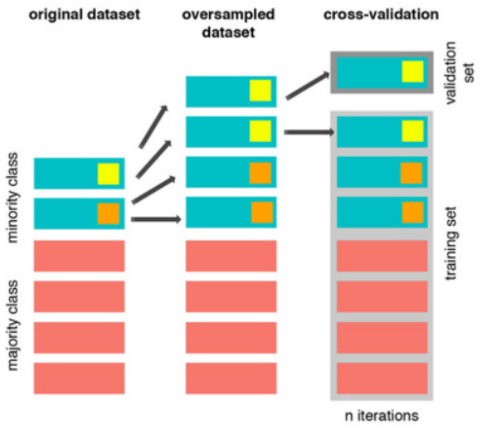
These are definitely a result of overfitting.

Cross Validation Overfitting Mistake:

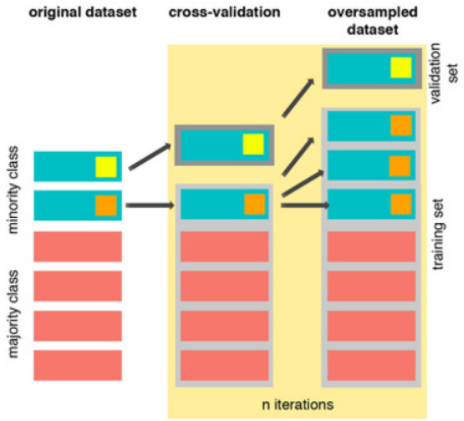
During our analysis of undersampling, we made a mistake due to which overfitting occurs during cross-validation. It's important to avoid undersampling or oversampling the data before cross-validation because doing so can cause a "data leakage" problem, where the validation set is directly influenced before implementing cross-validation. This can result in highly accurate precision and recall scores, but the data is overfitting.

Thus, here we will have to use SMOTE technique as it is applied "during" the cross-validation process and not "before" it. This means that synthetic data is only generated for the training set and does not affect the validation set.

Reason:



Undersampling cv



SMOTE cv

Applications:

Financial institutions, such as banks and credit card companies, use fraud detection models to identify fraudulent transactions and prevent financial losses.

E-commerce companies use fraud detection models to identify and prevent fraudulent transactions on their platforms.

Insurance companies use fraud detection models to identify fraudulent claims and prevent insurance fraud.

Healthcare providers use fraud detection models to identify fraudulent medical claims and prevent healthcare fraud.

Government agencies use fraud detection models to identify fraudulent tax returns and prevent tax fraud.