Credit Card Fraud Detection Using Machine Learning

Name:	**Kartikey Singh**
Registration No./Roll No.:	20146
Institute/University Name:	Indian Institute Of Science Education And Research, Bhopal
Program/Stream:	Electrical Engineering And Computer Science (EECS)
Problem Release date:	January 12, 2023
Date of Submission:	April 16, 2023

Problem Statement

Predict fraudulent banking transactions using various machine learning techniques.

Approach And Methods

Data Handling

The dataset provided is highly imbalanced and contains around **99.75%** non-fraudulent data (Class label : **0** and **0.25%** fraud data (Class label : **1**). Such high imbalance in data is treated by using **Upsampling** and **Downsampling**. We increase the examples of minority class (fraud data) by repeating the samples of it while downsample the majority class by reducing/removing samples of it. We improve the ratio of samples to **70 : 30** by sampling.

Exploratory Data Analysis

In this section, we will be performing EDA on our dataset to get important insights and to find important features. The first method used for this purpose are the **correlation heat maps** to find which of the features are highly correlated to the target variable **Y**. For this we have made 3 different heat maps for clarity and tried to choose features which are highly correlated to Y.
To get a much better choice of features, we have used **Extra Tree Classifier**, also known as Extremely Randomized Tree Classifier is an inbuilt library in sklearn that is specifically designed for choosing best features according to a chosen importance function. In our case, we have chosen **Gini Entropy** function which is also taken as default parameter in this library. Next we also have tried to analyse features individually by plotting density and scatter plots for correlation of each of them. The link to the ipynb file is attached here.

Methods And Techniques

We present the following solutions and techniques to be used in our model. All the below solutions will be tested and trained for their best performance using various cross validation techniques and fine tuning. The value of **K** was fixed to **10** for cross-validation over each model.

- The first solution we implemented is **Logistic Regression** model tuned by using GridSearchCV. This model came up with an accuracy of **97.3078%** and f-score of **0.97**. The best hyperparameter settings were : **C = 0.01 , Penalty = L1, solver = liblinear**, where **C** is the regularisation constant and **L1** is the Lasso regularisation.

- The next model we implemented is the **Random Forest Classifier** which was also tuned via GridSearchCV. This model increased the accuracy on the training data upto **97.9752%** while f-score increased to **1.00**. The best parameter settings were : **Criterion = entropy, Max_depth = 9**, where **Criterion** is the splitting measure and **Max_depth** is the depth of each decision tree ensembled in random forest classifier.

- The next model we took is the **K Nearest Neighbors** model which was too fine tuned via GridSearchCV over the paramters of **distance function** and **number of neighbors**. The accuracy of this model increased to **99.9451%** and f-score increased to **1.00**. The best parameter settings were : **Minkowski Distance Function with Neighbors = 3**. This also turned out to be the model with highest accuracy and f-score.

- The next solution we implemented is a simple **5 Layered Artificial Neural Network or Multilayer Perceptron** model with an **input layer** with **40 units**, **3 hidden layers with 30, 20 and 10 units** and an output layer with a **single unit**. The first four layers are provided with **ReLu activation** function while last layer is activated via **Sigmoid** function. This solution gave us an accuracy of **97.8964%** and an f-score of **0.98**. The best parameter settings were : **Optimizer = Adam, Loss = Binary Cross Entropy, Batch Size = 100 , Epochs = 20**.

- Next solution we present is using **Support Vector Machines**. The accuracy of this model was about **99.97%** with an f-score of **1.00**. The best parameter settings for this model were : **kernel = rbf, C = 0.1, gamma = 0.1**.

- The next solution we present is using **AdaBoost Classifier**. The accuracy of this model was about **99.9807%** and f-score of **1.00**. The best parameter settings for this model were : **base estimator = Decision Tree, Max Depth = 3, Min Sample Leaves = 5, learning rate = 0.1, n_estimators = 250**.

- The last solution we present is using **Decision Tree Classifier**. The accuracy of this model went upto **98.4994%** with f-score of **0.99**. The best parameter settings for this model were : **Max Leaf Nodes = 9, Min Sample Split = 2, Criterion = Gini**.

Observations

- All models were initially trained on original data but due to high imbalance, visible bias towards majority class (Non-fraud data) was seen (Accuracy was good while f-score was poor).

- With upsampling and downsampling of the data, the results of the models went up by a lot.

- AdaBoost Classifier and Support Vector Machines Turned Out to the best models in terms of accuracy on training data. The score merit chosen for training was **f-score**.

Contributions

Amit Shukla, Neural Network, Adaboost Classifier implementation, Report Making.
Kartikey Singh, Logistic Regression, K Nearest Neighbors implementation, Report Making.
Yash Gupta, Random Forest, Decision Tree Classifier, Support Vector Machines, implementation, Report Making.

Resources

Towards Data Science blog, Medium, Analytics Vidya, Machine Learning mastery, GeeksForGeeks.