CREDIT CARD FRAUD DETECTION USING DECISION TREE ALGORITHM

A Project
Report
Submitted by

D HUSSAIN(19331A1216)
E BHAVYA SATHVIKA(19331A1217)
G MYTHILI (19331A1218)

in partial fulfillment for the Award of the Degree of
BACHELOR OF TECHNOLOGY

IN INFORMATIONTECHNOLOGY

MVGR COLLEGE OF ENGINEERING

2019-2023

PROBLEM STATEMENT

Credit card fraud detection using Decision Tree Algorithm.

INTRODUCTION

Credit card fraud is a sober and major growing problem in banking industries. With the advent of the rise of many web services provided by banks, banking frauds are also on the increase. Banking systems always have a strapping security system in order to detect and prevent fraudulent activities of any category of transactions.

Totally eliminating banking fraud is almost unfeasible, but we can however minimize the frauds and prevent them from happening by machine learning techniques like Data Mining. It represents how to utilize these data and find useful information from data has become an urgent need for detection of fraud.

Therefore, data mining technology has become an effective method for detection of fraud. Thus, we are developing a fraud detection system for credit cards using a decision tree induction algorithm for security using Data Mining Technique.

HARDWARE SPECIFICATION

Hardware (PC) used for building this project has the following specifications:

Processor: Intel(R) Core(TM) i5

RAM: 8 GB

64-bit OS

Windows specifications : Windows 11

Minimum requirements for the hardware specifications to implement the project are:

Processor: i3, i7

RAM: 4GB

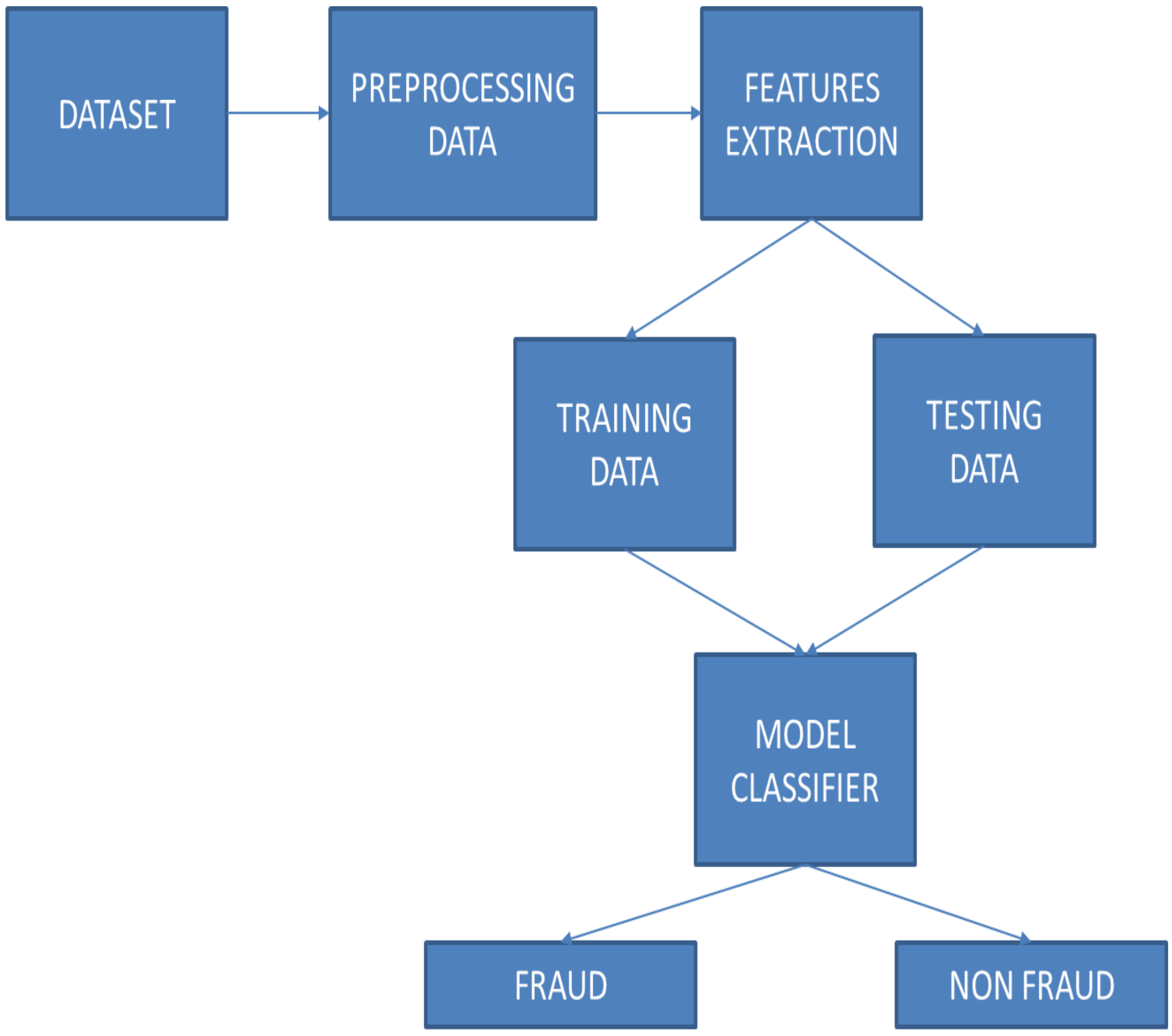
32-bit OS

It is platform independent. So, various operating systems can be used like MacOS, Linux, etc.

SOFTWARE SPECIFICATION

- Operating System: Windows 11
- Executing Language: Python

SYSTEM ARCHITECTURE



ALGORITHM

Before creating a model, we need to find the type of problem statement, whether supervised or unsupervised algorithms.

Decision Tree Analysis is a general, predictive modelling tool with applications spanning several different areas. In general, decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on various conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

The decision rules are generally in the form of if-then-else statements. The deeper the tree, the more complex the rules and fitter the model. The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by **learning simple decision rules** inferred from prior data (training data).

Our problem statement falls under the supervised learning problem meaning the dataset has a target value for each row or sample in the dataset.

Supervised machine learning algorithms are two types

1. Classification Algorithm
2. Regression Algorithm

Decision Tree is considered one of the most useful Machine Learning algorithms since it can solve various problems. Here are a few reasons why we used the Decision Tree:

1. It is considered to be the most understandable Machine Learning algorithm, and it can be easily interpreted. It can be used for classification and regression problems.
2. Unlike most Machine Learning algorithms, it works effectively with non-linear data.
3. Constructing a Decision Tree is a speedy process since it uses only one feature per node to split the data.
4. **Decision Trees** model data as a “**Tree**” of hierarchical branches. They make branches until they reach “**Leaves**” that represent predictions. Due to their branching structure, **Decision Trees** can easily model non-linear relationships.
5. For building the model, the decision tree algorithm considers all the provided features of the date and comes up with the important features. Because of this advantage, the decision tree algorithms are also used in identifying the importance of the feature metrics, which are used in handpicking the features.
6. Once the important features are identified then the model trains with the training data to come up with a set of rules. These rules are used in predicting future cases or for the test dataset.

IMPLEMENTATION

MODULES AND LIBRARIES USED

- sklearn: Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface inPython.

- imblearn: Imbalanced-learn (imported as imblearn) is an open source, MIT-licensed library relying on scikit-learn (imported as sklearn) and providestoolswhendealingwith classificationwithimbalancedclasses.

- numpy: Library consisting of multidimensional array objects and a collection of routines for processing those arrays.

- pandas: pandas is a python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive.

- Matplotlib: matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plottingarea,decoratestheplotwith labels, etc.

- DecisionTreeClassifier: Its ability to use different feature subsets and decision rulesat differentstagesofclassification.

- train_test_split: It is a Python library that offers various features for data processing that can beusedforclassification,clustering,andmodel selection.

- Smote: Smote is an oversampling technique that generates synthetic samples from the minority class.

DATASET:

Description of dataset:

- The dataset contains transactions made by credit cards in September 2013 by European cardholders.
- This dataset presents transactions that occurred in two days
- There are 492 frauds out of 284,807 transactions.
- The dataset is highly unbalanced
- The positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a Principle Component Analysis (PCA) transformation. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'.

Headers

'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset.
'Amount' is the transaction Amount; this feature can be used for example-dependent cost-sensitive learning.
'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

SAMPLE OF DATASET

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390	-0.311169	1.468177	-0.470401	0.207971	0.025791	0.403993	0.251412	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.143772	0.635558	0.463917	-0.114805	-0.183361	-0.145783	-0.069083	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293	-0.165946	2.345865	-2.890083	1.109969	-0.121359	-2.261857	0.524980	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757	-0.287924	-0.631418	-1.059647	-0.684093	1.965775	-1.232622	-0.208038	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158	0.87773	1.54871	0.40303	-0.407	0.09592	0.59294	-0.270	0.81773	0.75307	-0.822	0.53819	1.34585	-1.119	0.17512	-0.451	-0.237	-0.038	0.80348	0.40854	-0.009	0.79827	-0.137	0.14126	-0.206	0.50229	0.21942	0.21515	69.99	0

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
		233	7	8	4	193	1	1	533	9	4	843	6	2	670	1	449	033	195	7	2	431	8	458	7	010	2	2	3		

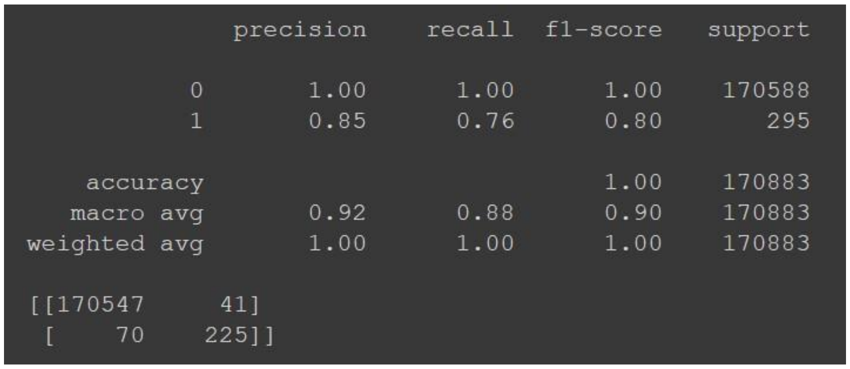
SOURCE CODE

```
# IMPORTING REQUIRED MODULES

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier
import collections

from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.metrics import confusion_matrix , precision_score, recall_score, \
f1_score, roc_auc_score, accuracy_score, classification_report
from sklearn.model_selection import KFold, StratifiedKFold

import warnings
warnings.filterwarnings('ignore')

# DATA

df = pd.read_csv('/content/drive/MyDrive/AITTA PBL/Dataset/creditcard.csv')
df.head()

#TRAIN - TEST SPLIT

df_train, df_test = train_test_split(df, test_size=0.2,random_state=123,stratify=df["Class"])
df_train, df_val = train_test_split(df_train,
test_size=0.25,random_state=123,stratify=df_train["Class"])

#SCALING the columns that are left to scale (AMOUNT AND TIME)

from sklearn.preprocessing import StandardScaler, RobustScaler

# RobustScaler is less prone to outliers.

std_scaler = StandardScaler()
rob_scaler = RobustScaler()
```

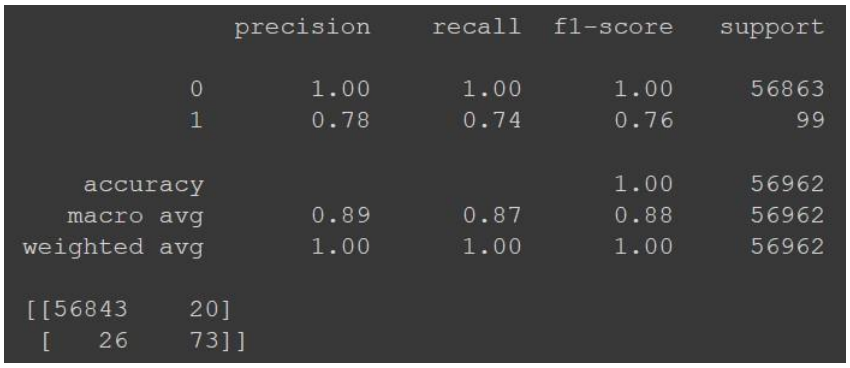
```
df_train['scaled_amount'] = rob_scaler.fit_transform(df_train['Amount'].values.reshape(-
1,1))
df_train['scaled_time'] = rob_scaler.fit_transform(df_train['Time'].values.reshape(-1,1))

df_train.drop(['Time','Amount'], axis=1, inplace=True)

X_train = df_train.drop(["Class"], axis = 1)
y_train = df_train["Class"]

df_val['scaled_amount'] = rob_scaler.fit_transform(df_val['Amount'].values.reshape(-1,1))
df_val['scaled_time'] = rob_scaler.fit_transform(df_val['Time'].values.reshape(-1,1))
df_val.drop(['Time','Amount'], axis=1, inplace=True)

X_val = df_val.drop(["Class"], axis = 1)
y_val = df_val["Class"]

#TRAIN

dtc_cfl = DecisionTreeClassifier(random_state=1,max_depth=2)

dtc_cfl.fit(X_train, y_train)

y_predict = dtc_cfl.predict(X_train)

# evaluate the model
print(classification_report(y_train, y_predict))
print(confusion_matrix(y_train, y_predict))

#VAL

dtc_cfl = DecisionTreeClassifier(random_state=1,max_depth=2)

dtc_cfl.fit(X_train, y_train)

y_predict = dtc_cfl.predict(X_val)

# evaluate the model
print(classification_report(y_val, y_predict))
print(confusion_matrix(y_val, y_predict))
#Accuracy
print("Accuracy: ", accuracy_score(y_val,y_predict))
```

OUTPUT

Output for **train** data (evaluated into confusion matrix and shown as report)


```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    170588
           1       0.85      0.76      0.80       295

    accuracy                           1.00    170883
   macro avg       0.92      0.88      0.90    170883
weighted avg       1.00      1.00      1.00    170883

[[170547     41]
 [    70    225]]
```

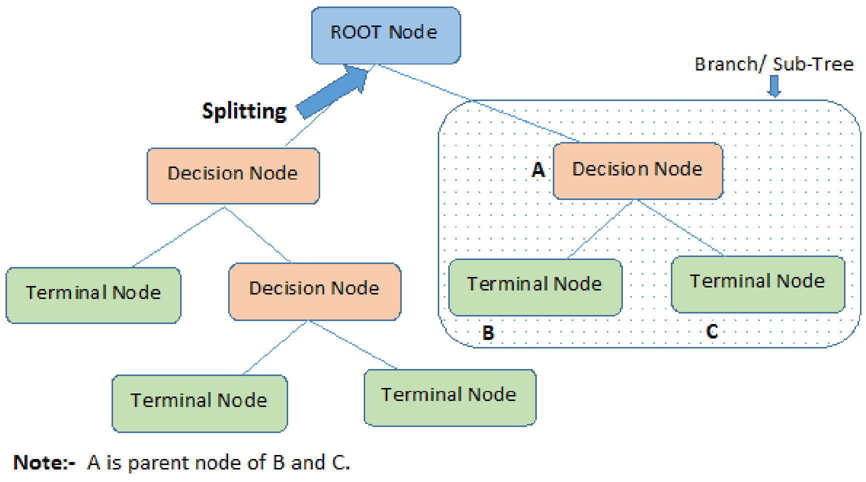
Output for **test** data (evaluated into confusion matrix and shown as report)


```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56863
           1       0.78      0.74      0.76        99

    accuracy                           1.00     56962
   macro avg       0.89      0.87      0.88     56962
weighted avg       1.00      1.00      1.00     56962

[[56843    20]
 [   26    73]]
```

Accuracy: `0.9991924440855307`

INFERENCE

Decision tree:



1. **Root Node:** It represents the entire population or sample, and this further gets divided into two or more homogeneous sets.
2. **Leaf/ Terminal Node:** Nodes which are not split is called Leaf or Terminal node.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called a decision node.
4. **Branch / Sub-Tree:** A subsection of the entire tree is called a branch or sub-tree.
5. **Parent and Child Node:** A node, which is divided into sub-nodes, is called the parent node of sub-nodes, whereas sub-nodes are the child of the parent node.
6. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
7. **Pruning:** Pruning is when we selectively remove branches from a tree. The goal is to remove unwanted branches, improve the tree's structure, and direct new, healthy growth.

What is Decision Tree?

A **Decision Tree** is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers to the question, and the leaves represent the actual output or class label. They are used in non-linear decision making with a simple linear decision surface.

The Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can solve **regression** and **classification problems**.

In Decision Trees, we start from the tree's root for predicting a class label for a record. We compare the values of the root attribute with the record's attribute. Based on the comparison, we follow the branch corresponding to that value and jump to the next node.

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of the root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree.

The complete algorithm can be better divided into the following steps:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset.
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

CONCLUSION

Credit card fraud has become more and more widespread in recent years. To increase merchant riskmanagement level in an automatic and active way, structure a perfect and easy handling credit card risk monitoring system is one of the key tasks for the merchant banks. So, we propose a credit card fraud detection problem for the resolution of reducing the bank's risk. With the historical profile patterns, make use of credit card fraud detection models to equal the transaction information to predict the probability of being fraudulent for a new transaction. It offers a scientificbasisfortheauthorizationmechanisms.