

VARUVAN VADIVELAN INSTITUTE OF TECHNOLOGY

NAAN MUDHALAVAN : IBM

PHASE : 5

TECHNOLOGY : DATA SCIENCE

**PROJECT TITLE : CREDIT CARD FURD
DETECTION**

Problem statement:

The Credit Card Fraud Detection Problem includes modeling past credit card transactions with the knowledge of the ones that turned out to be fraud. This model is then used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications.

Design thinking:

Design thinking have some process here.

- Creditcard data
- Data pre processing
- Data analysis
- Train-test-split
- Evaluation

Modules:

1. Data pre processing
2. Feature engineering
3. Machine learning algorithms

Credit card transaction data:

	Time	V1	V2	V3	...	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	...	0.133558	-0.021053		
149.62	0								
1	0.0	1.191857	0.266151	0.166480	...	-0.008983	0.014724		
2.69	0								
3	1.0	-0.966272	-0.185226	1.792993	...	0.062723	0.061458		
123.50	0								
4	2.0	-1.158233	0.877737	1.548718	...	0.219422	0.215153		
69.99	0								
...		
284802	172786.0	-11.881118	10.071785	-9.834783	...	0.943651			
0.823731	0.77	0							
284803	172787.0	-0.732789	-0.055080	2.035030	...	0.068472	-		
0.053527	24.79	0							
284804	172788.0	1.919565	-0.301254	-3.249640	...	0.004455	-		
0.026561	67.88	0							
284805	172788.0	-0.240440	0.530483	0.702510	...	0.108821			
0.104533	10.00	0							
284806	172792.0	-0.533413	-0.189733	0.703337	...	-0.002415			
0.013649	217.00	0							

[284807 rows x 31 columns]

1. Data preprocessing:

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

some stems is there in preprocessing :

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

Getting the data set:

To create a machine learning model, the first thing required is a dataset as a machine learning model completely works on data. The collected data for a particular problem in a proper format is known as the dataset

Importing libraries:

In order to perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are three specific libraries that we will use for data preprocessing

Numpy:

Numpy Python library is used for including any type of mathematical operation in the code. It is the fundamental package for

scientific calculation in Python. It also supports to add large, multidimensional arrays and matrices. So, in Python, we can import it as:

```
import pandas as pd
```

Matplotlib:

The second library is matplotlib, which is a Python 2D plotting library, and with this library, we need to import a sub library pyplot. This library is used to plot any type of charts in Python for the code

```
import matplotlib.pyplot as plt
```

Pandas:

The last library is the Pandas library, which is one of the most famous Python libraries and used for importing and managing the datasets. It is an open-source data manipulation and analysis library. It will be imported as below:

```
import pandas as pd
```

Importing necessary libraries:

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
```

Handling missing value:

The next step of data preprocessing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset.

Ways to handle missing data:

There are mainly two ways to handle missing data, which are:

- i. By deleting the particular row
- ii. By calculating the mean

By deleting the particular row:

The first way is used to commonly deal with null values. In this way, we just delete the specific row or column which consists of null values. But this way is not so efficient and removing data may lead to loss of information which will not give the accurate output.

By calculating the mean:

In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc. Here, we will use this approach

Encoding Categorical data :

Categorical data is data which has some categories such as, in our dataset; there are two categorical variables, Country, and Purchased. Since machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So it is necessary to encode these categorical variables into numbers.

For Country variable:

Firstly, we will convert the country variables into categorical data. So to do this, we will use LabelEncoder() class from preprocessing library in our code, which contains various libraries for building machine learning models. Here we will use Imputer class of sklearn.preprocessing library.

```
# categorical data
#for Country Variable
```

```
from sklearn.preprocessing import LabelEncoder
label_encoder_x= LabelEncoder()
x[:, 0]= label_encoder_x.fit_transform(x[:, 0])
```

Splitting the Dataset into the Training set and Test set:

In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our machine learning model.

Training Set:

A subset of dataset to train the machine learning model, and we already know the output

Test set:

A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

For splitting the dataset, we will use the below lines of code

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2,
random_state=0)
```

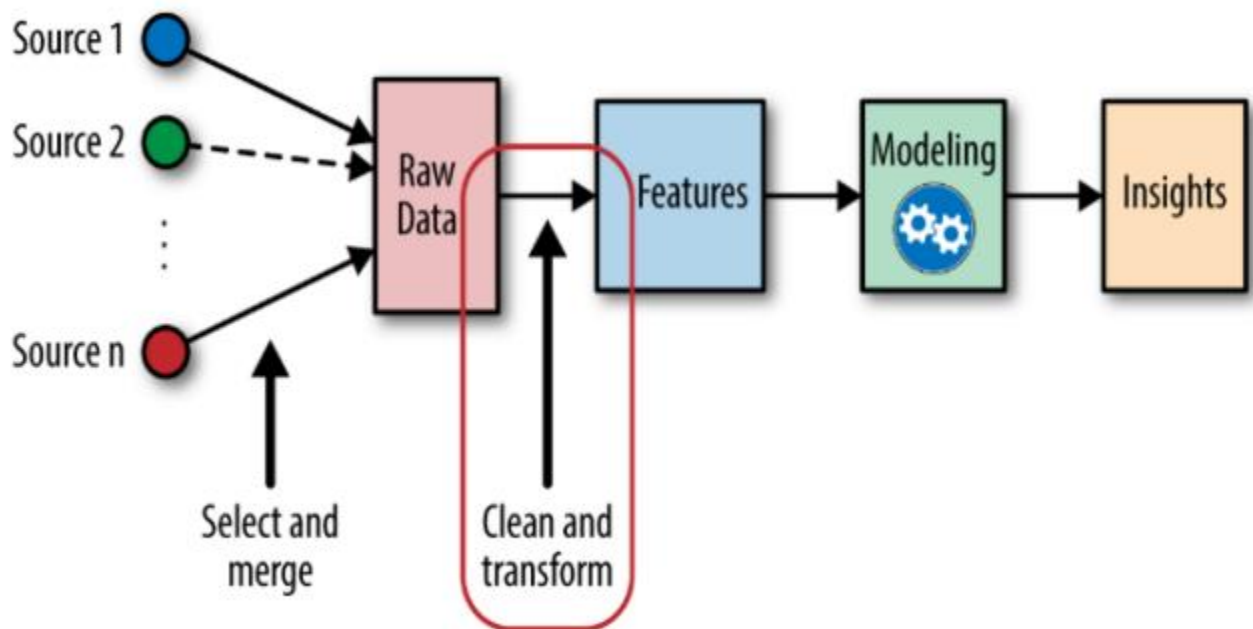
Feature Scaling:

Feature scaling is the final step of data preprocessing in machine learning. It is a technique to standardize the independent variables of the dataset in a specific range. In feature scaling, we put our variables in the same range and in the same scale so that no any variable dominate the other variable

2. Feature engineering:

Feature engineering is the process of selecting, manipulating, and transforming raw data into features that can be used in

supervised learning. In order to make machine learning work well on new tasks, it might be necessary to design and train better features. As you may know, a “feature” is any measurable input that can be used in a predictive model — it could be the color of an object or the sound of someone’s voice. Feature engineering, in simple terms, is the act of converting raw observations into desired features using statistical or machine learning approaches



Feature engineering various process:

Creating features involves creating new variables which will be most helpful for our model.

Transformations:

Feature transformation is simply a function that transforms features from one representation to another.

Feature Extraction: Feature extraction is the process of extracting features from a data set to identify useful information.

Exploratory Data Analysis :

Exploratory data analysis (EDA) is a powerful and simple tool that can be used to improve your understanding of your data, by exploring its properties.

Benchmark :

A Benchmark Model is the most user-friendly, dependable, transparent, and interpretable model against which you can measure your own.

Feature engineering techniques:

Lets see a few feature engineering best techniques that you can use. Some of the techniques listed may work better with certain algorithms or datasets, while others may be useful in all situations.

- Imputation
- Handling outliers
- Log transform
- One hot encoding
- Scaling

Imputation:

When it comes to preparing your data for machine learning, missing values are one of the most typical issues

1.Numerical Imputation

#Filling all missing values with 0

```
data = data.fillna(0)
```


2.categorical Imputation

#Max fill function for categorical columns

Handling outliers:

Outlier handling is a technique for removing outliers from a dataset. This method can be used on a variety of scales to produce a more accurate data representation. This has an impact on the model's performance.

Log Transform:

Log Transform is the most used technique among data scientists. It's mostly used to turn a skewed distribution into a normal or less-skewed distribution. We take the log of the values in a column and utilise those values as the column in this transform.

```
//Log Example
```

```
df[log_price] = np.log(df['Price'])
```

One-hot encoding:

A one-hot encoding is a type of encoding in which an element of a finite set is represented by the index in that set, where only one element has its index set to "1" and all other elements are assigned indices within the range $[0, n-1]$.

Scaling:

Feature scaling is one of the most pervasive and difficult problems in machine learning, yet it's one of the most important things to get right. In order to train a predictive model, we need data with a known set of features that needs to be scaled.

There are two types in scaling

- Normalization
- Standardization

Normalization:

All values are scaled in a specified range between 0 and 1 via normalisation (or min-max normalisation). This modification has no influence on the feature's distribution, however it does exacerbate the effects of outliers due to lower standard deviations. As a result, it is advised that outliers be dealt with prior to normalisation.

Standardization:

Standardization (also known as z-score normalisation) is the process of scaling values while accounting for standard deviation. If the standard deviation of features differs, the range of those features will likewise differ. The effect of outliers in the characteristics is reduced as a result. To arrive at a distribution with a 0 mean and 1 variance, all the data points are subtracted by their mean and the result divided by the distribution's variance.

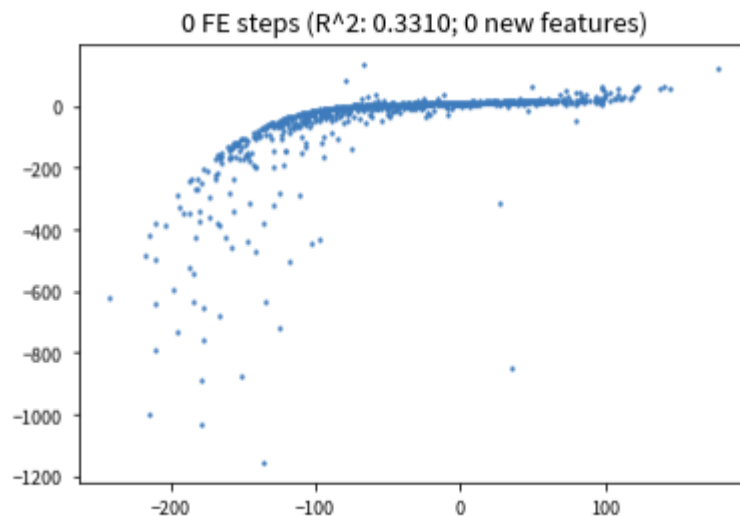
Feature Engineering Tools:

There are many tools which will help you in automating the entire feature engineering process and producing a large pool of features in a short period of time for both classification and regression tasks.

AutoFeat:

AutoFeat helps to perform Linear Prediction Models with Automated Feature Engineering and Selection. AutoFeat allows you to select the units of the input variables in order to avoid the construction of physically nonsensical features.

Example:



Ts Fresh:

TsFresh is a python package. It calculates a huge number of time series characteristics, or features, automatically. In addition, the package includes methods for assessing the explanatory power and significance of such traits in regression and classification tasks.

Example:



ONEBM :

OneBM interacts directly with a database's raw tables. It slowly joins the tables, taking different paths on the relational tree. It recognises simple data types in the joint results and applies pre-defined feature engineering.

EXPLORE KIT:

Based on the idea that extremely informative features are typically the consequence of manipulating basic ones, ExploreKit identifies common operators to alter each feature independently or combine multiple of them. Instead of running feature selection on all developed features

3.Machine learning algorithm:

Machine Learning algorithms are the programs that can learn the hidden patterns from the data, predict the output, and improve the performance from experiences on their own. Different algorithms can be used in machine learning for different tasks

Types of Machine Learning Algorithms

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Supervised learning:

Supervised learning is a type of Machine learning in which the machine needs external supervision to learn. The supervised learning models are trained using the labeled dataset. Once the training and processing are done, the model is tested by providing a sample test data to check whether it predicts the correct output.

Supervised learning can be divided further into two categories of problem:

- Classification
- Regression

Unsupervised Learning Algorithm:

It is a type of machine learning in which the machine does not need any external supervision to learn from the data, hence called unsupervised learning. The unsupervised models can be trained using the unlabelled dataset that is not classified, nor categorized, and the algorithm needs to act on that data without any supervision.

it can be classified into two types:

- clustering
- Association

Reinforcement Learning:

In Reinforcement learning, an agent interacts with its environment by producing actions, and learn with the help of feedback. The feedback is given to the agent in the form of rewards, such as for each good action, he gets a positive reward, and for each bad action, he gets a negative reward.

Reinforcement classified into some few types.

- Linear Regression Algorithm
- Decision Tree
- SVM
- Naïve Bayes
- K-Means Clustering
- Random Forest

Linear regression algorithm:

Linear regression is one of the most popular and simple machine learning algorithms that is used for predictive analysis.

It tries to best fit a line between the dependent and independent variables, and this best fit line is known as the regression line.

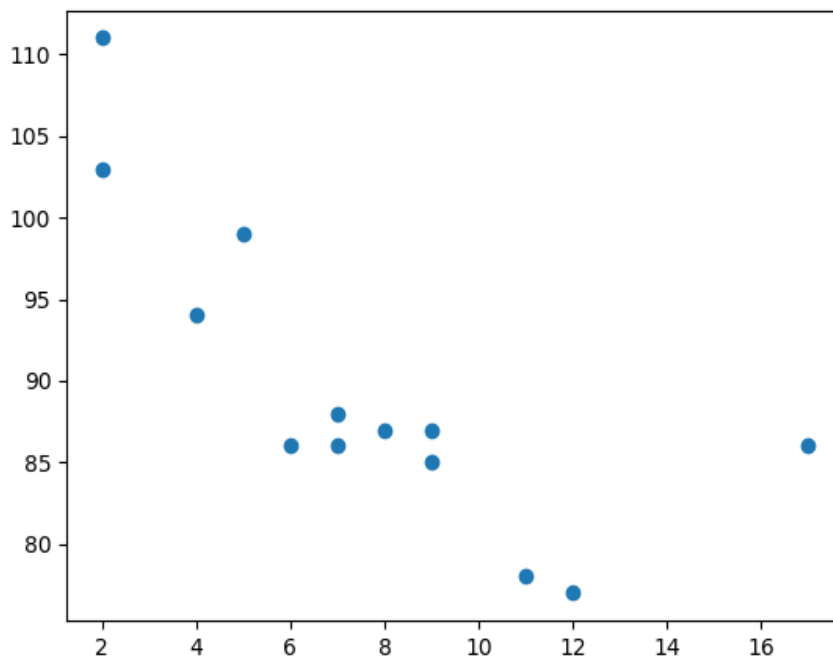
Program:

```
import matplotlib.pyplot as plt

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

plt.scatter(x, y)
plt.show()
```

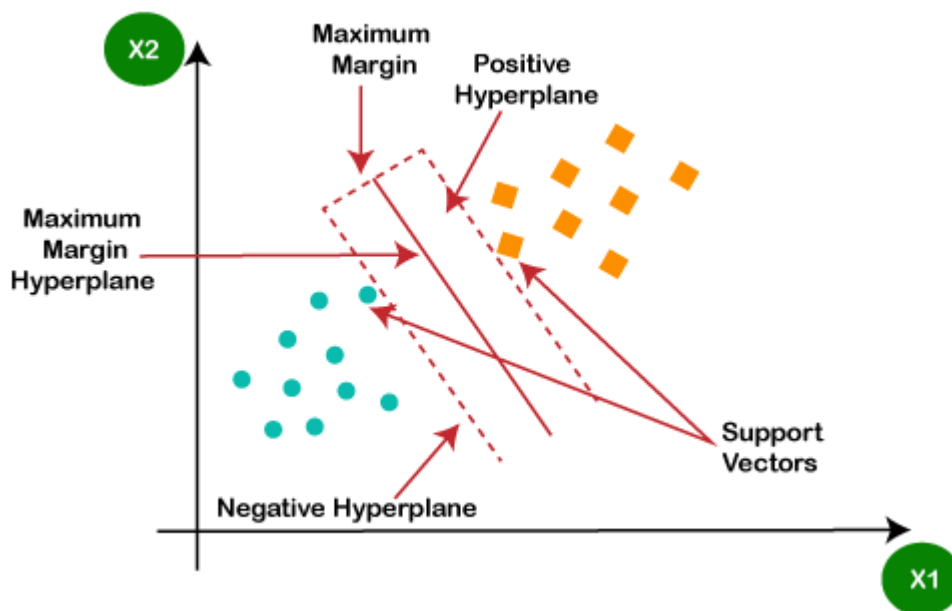
Output:



Support Vector Machine Algorithm:

A support vector machine or SVM is a supervised learning algorithm that can also be used for classification and regression problems. However, it is primarily used for classification problems. The goal of SVM is to create a hyperplane or decision boundary that can segregate datasets into different class.

Some real-life applications of SVM are face detection, image classification, Drug discovery, etc. Consider the below diagram:



Decision Tree Algorithm:

A decision tree is a supervised learning algorithm that is mainly used to solve the classification problems but can also be used for solving the regression problems. It can work with both categorical variables and continuous variables. It shows a tree-like structure that includes nodes and

branches, and starts with the root node that expand on further branches till the leaf node.

Program:

```
import pandas  
df = pandas.read_csv("data.csv")  
print(df)
```

Output:

Creditcard.csv file.

Naïve Bayes Algorithm:

Naïve Bayes classifier is a supervised learning algorithm, which is used to make predictions based on the probability of the object. The algorithm named as Naïve Bayes as it is based on Bayes theorem.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

K-Means Clustering:

K-means clustering is one of the simplest unsupervised learning algorithms, which is used to solve the clustering problems. The datasets are grouped into K different clusters based on similarities and dissimilarities.

K-means, K-refers to the number of clusters, and **means** refer to the averaging the dataset in order to find the centroid.

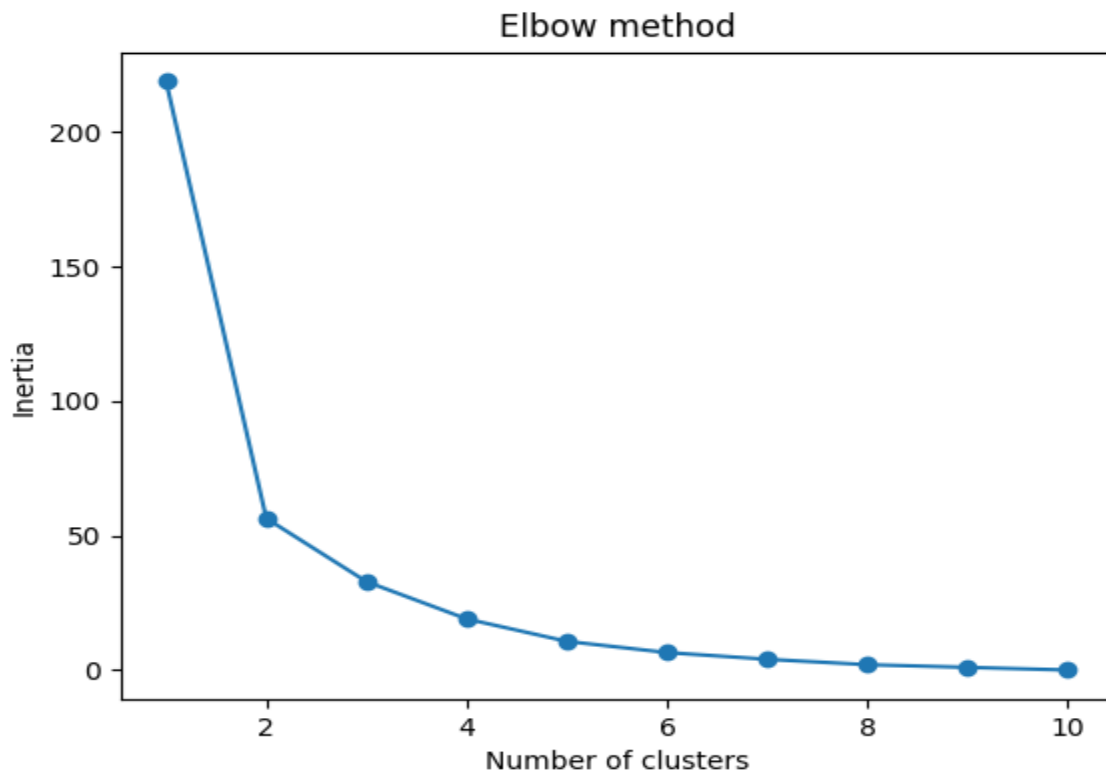
Program:

```
from sklearn.cluster import KMeans  
data = list(zip(x, y))
```



```
inertias = []  
for i in range(1,11):  
    kmeans = KMeans(n_clusters=i)  
    kmeans.fit(data)  
    inertias.append(kmeans.inertia_)  
plt.plot(range(1,11), inertias, marker='o')  
plt.title('Elbow method')  
plt.xlabel('Number of clusters')  
plt.ylabel('Inertia')  
plt.show()
```

Output:



Random forest:

Random forest is the supervised learning algorithm that can be used for both classification and regression problems in machine learning. It is an ensemble learning technique that provides the predictions by combining the multiple classifiers and improve the performance of the model.

It contains multiple decision trees for subsets of the given dataset, and find the average to improve the predictive accuracy of the model. A random-forest should contain 64-128 trees. The greater number of trees leads to higher accuracy of the algorithm.

Model training:

A training model is a dataset that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The training model is used to run the input data through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model.

This iterative process is called “model fitting”. The accuracy of the training dataset or the validation dataset is critical for the precision of the model.

Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved. There are several types of machine learning models, of which the most common ones are supervised and unsupervised learning.

Supervised learning is possible when the training data contains both the input and output values. Each set of data that has the inputs and the expected output is called a supervisory signal. The training is done base on the deviation of the processed result from the document”

Some steps involves in model training:

Defining The Problem

Data collection

Data preparation

Assign the model

Training the model

Evaluation of model

Evaluation metrics:

a classification task, our main task is to predict the target variable which is in the form of discrete values. To evaluate the performance of such a model there are metrics as mentioned below:

- Classification Accuracy
- Logarithmic loss
- Area under Curve
- F1 score
- Precision
- Recall
- Confusion Matrix

Confusion Matrix

It creates a $N \times N$ matrix, where N is the number of classes or categories that are to be predicted. Here we have $N = 2$, so we get a 2×2 matrix. Suppose there is a problem with our practice which is a binary classification.

Final program:

Import necessary libraries

```
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import accuracy_score, confusion_matrix,  
classification_report
```

```
# Load the dataset (replace 'creditcard.csv' with your dataset)
```

```
data = pd.read_csv('creditcard.csv')
```

```
# Data preprocessing
```

```
# You might need to do more extensive data cleaning and feature  
engineering
```

```
X = data.drop(['Class'], axis=1)
```

```
y = data['Class']
```

```
# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,  
random_state=42)
```

```
# Create and train the model
```

```
model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
model.fit(X_train, y_train)
```

Make predictions on the test set

```
y_pred = model.predict(X_test)
```

Evaluate the model

```
accuracy = accuracy_score(y_test, y_pred)
```

```
conf_matrix = confusion_matrix(y_test, y_pred)
```

```
classification_rep = classification_report(y_test, y_pred)
```

```
print("Accuracy:", accuracy)
```

```
print("Confusion Matrix:")
```

```
print(conf_matrix)
```

```
print("Classification Report:")
```

```
print(classification_rep)
```

Output:

Import libraries:

```
import pandas as pd
✓ 7.3s

from sklearn.model_selection import train_test_split
✓ 7.2s

from sklearn.ensemble import RandomForestClassifier
✓ 1.7s

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
✓ 0.0s

data = pd.read_csv('C:/Users/Hemamalini/Desktop/credit card/creditcard.csv')
print(data)
✓ 7.4s
```

Load the transaction data:

...	Time	V1	V2	V3	V4	V5	\
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	
...	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	
...	
0	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838
1	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672
2	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679
3	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274
4	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278
...
284802	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864
284803	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384
284804	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229
284805	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049

Data preprocessing and splitting data:

```
jupyter.ipynb
C:\Users\Hemamalini\Desktop> jupyter.ipynb > print(conf_matrix)
+ Code + Markdown + Run All + Restart + Clear All Outputs + Variables + Outline ...

284805    0
284806    0

[284807 rows x 31 columns]
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

X = data.drop(['Class'], axis=1)
[6] ✓ 0.0s

y = data['Class']
[7] ✓ 0.0s

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
[8] ✓ 0.5s

model = RandomForestClassifier(n_estimators=100, random_state=42)
[9] ✓ 0.0s
```

Prediction model:

```
[10] ✓ 28m 48.1s Python
...
RandomForestClassifier
RandomForestClassifier(random_state=42)

y_pred = model.predict(X_test)
[11] ✓ 1.9s Python

accuracy = accuracy_score(y_test, y_pred)
[12] ✓ 0.0s Python

conf_matrix = confusion_matrix(y_test, y_pred)
[13] ✓ 0.1s Python

classification_rep = classification_report(y_test, y_pred)
[14] ✓ 0.3s Python
```

Evaluating the model:

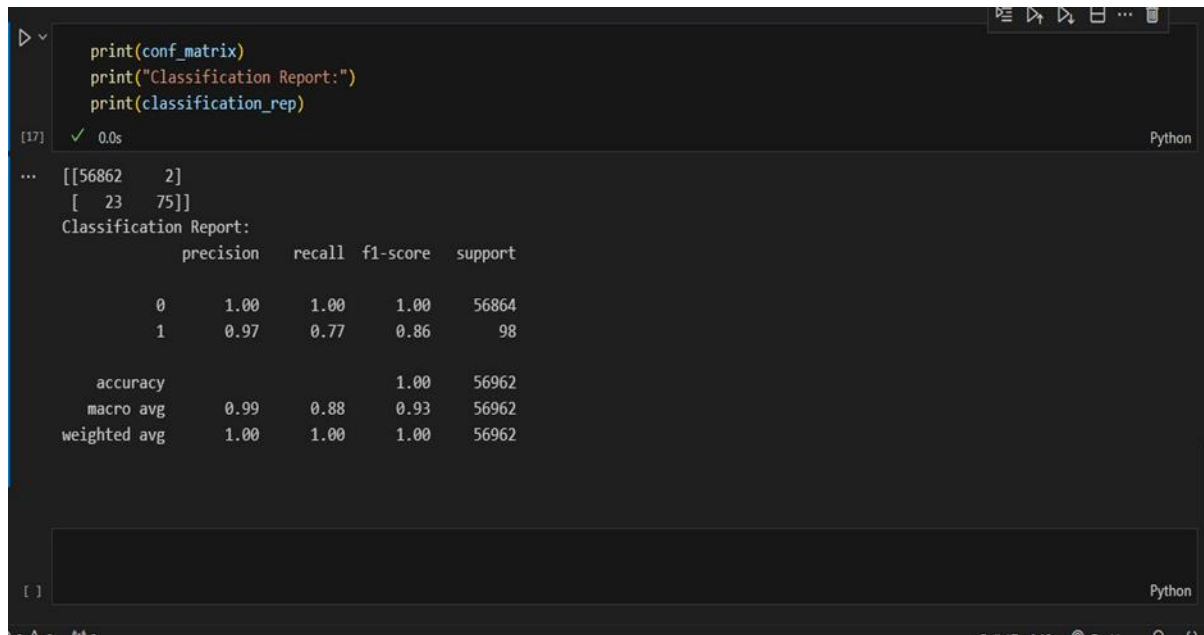
```
classification_rep = classification_report(y_test, y_pred)
[14] ✓ 0.3s

print("Accuracy:", accuracy)
[15] ✓ 0.0s
... Accuracy: 0.9995611109160493

print("Confusion Matrix:")
[16] ✓ 0.0s
... Confusion Matrix:

print(conf_matrix)
print("Classification Report:")
print(classification_rep)
[17] ✓ 0.0s
```

Confusion matrix:



```
print(conf_matrix)
print("Classification Report:")
print(classification_rep)
```

[17] ✓ 0.0s Python

```
... [[56862  2]
      [ 23  75]]
Classification Report:
              precision    recall  f1-score   support

      0       1.00      1.00      1.00     56864
      1       0.97      0.77      0.86        98

 accuracy      0.99
 macro avg     0.99      0.88      0.93     56962
 weighted avg   1.00      1.00      1.00     56962
```

[] Python

Conclusion:

We investigated the data, checking for data unbalancing, visualizing the features, and understanding the relationship between different features. We then investigated two predictive models. The data was split into three parts, a train set, a validation set, and a test set. For the first three models, we only used the train and test set