CREDIT CARD FRAUD DEDUCTION USING APPLIED DATA SCIENCE

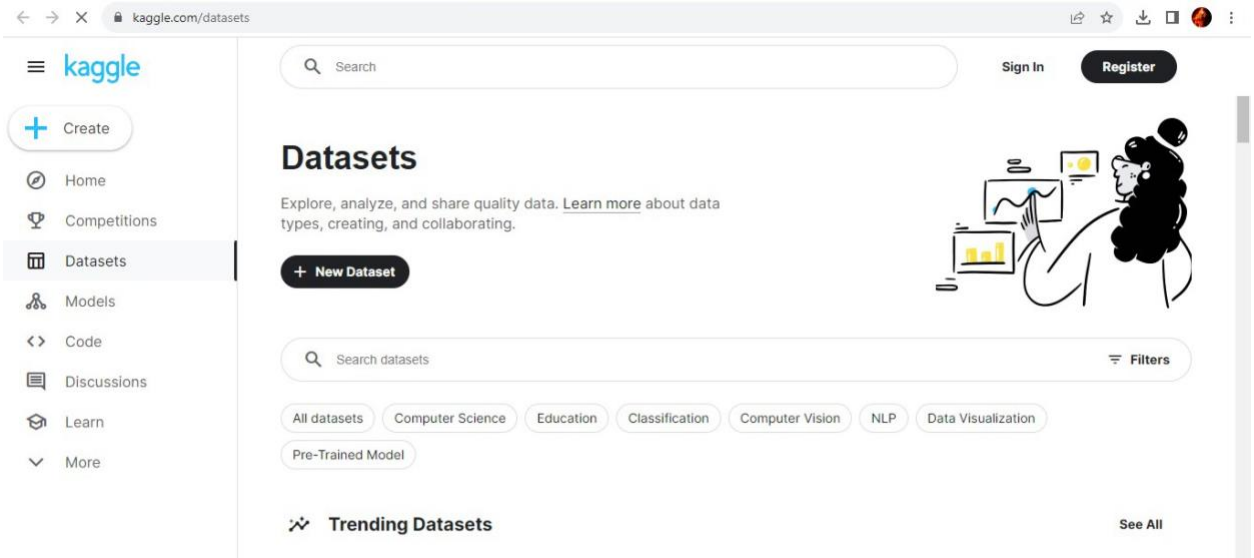
PHASE 3 : DEVELOPMENT PART - 1

Data science is an interdisciplinary field that encompasses a set of techniques, processes, and methods for extracting valuable insights, knowledge, and patterns from data. It combines elements of statistics, computer science, domain expertise, and data engineering to collect, analyze, and interpret large and complex datasets. The ultimate goal of data science is to use data to inform decision-making, solve problems, and drive improvements in various domains, including business, healthcare, finance, and more. Data scientists use a combination of data analysis, machine learning, data visualization, and domain-specific expertise to uncover meaningful information from data and provide valuable insights for organizations and individuals.

1. DATASEET AND ITS DETAIL EXPLANATION :

This dataset is taken from the website name ”kaggle” and the link of the website is given below

LINK : www.Kaggle.com/data

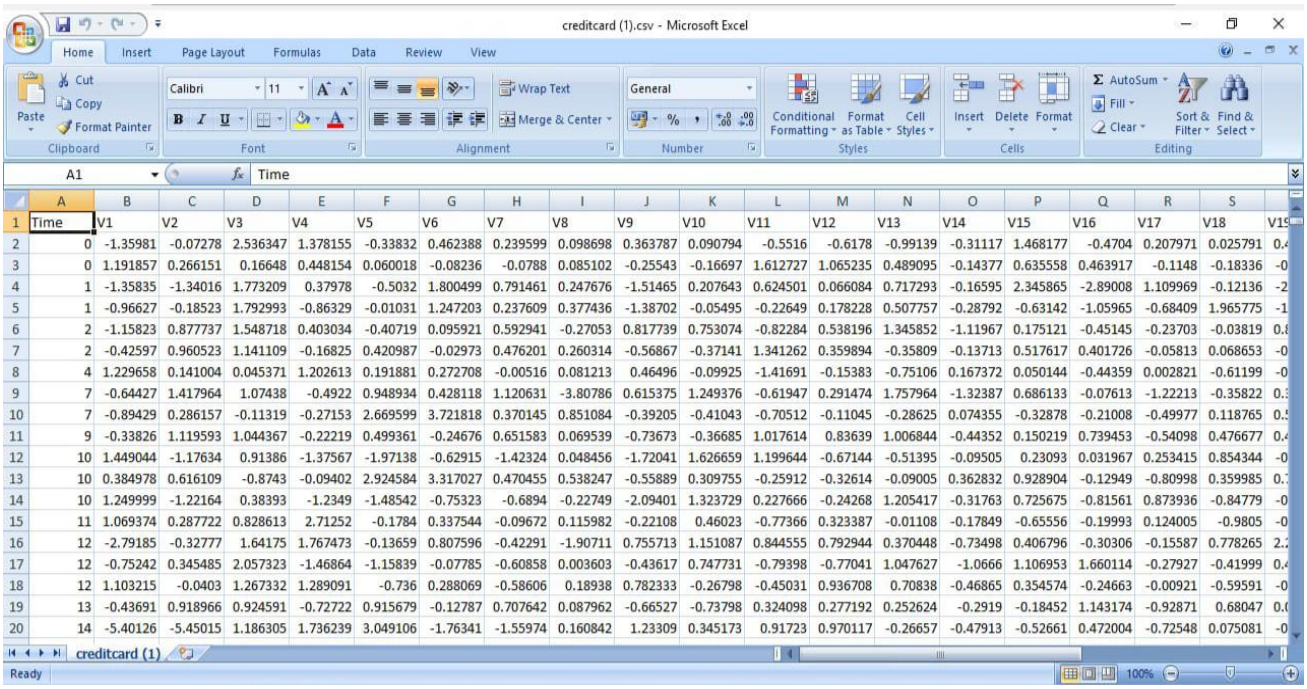


In this page search for your desired dataset about the credit card fraud deduction

The link for this project**:**

https://www.kaggle.com/datasets/mig-ulb/creditcardfraud

DETAILS ABOUT DATASET :



Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19
0	-1.35981	-0.07278	2.536347	1.378155	-0.33832	0.462388	0.239599	0.098698	0.363787	0.090794	-0.5516	-0.6178	-0.99139	-0.31117	1.468177	-0.4704	0.207971	0.025791	0.4
0	1.191857	0.266151	0.16648	0.448154	0.060018	-0.08236	-0.0788	0.085102	-0.25543	-0.16697	1.612727	1.065235	0.489095	-0.14377	0.635558	0.463917	-0.1148	-0.18336	-0
1	-1.35835	-1.34016	1.773209	0.37978	-0.5032	1.800499	0.791461	0.247676	-1.51465	0.207643	0.624501	0.066084	0.717293	-0.16595	2.345865	-2.89008	1.109969	-0.12136	-2
1	-0.96627	-0.18523	1.792993	-0.86329	-0.01031	1.247203	0.237609	0.377436	-1.38702	-0.05495	-0.22649	0.178228	0.507757	-0.28792	-0.63142	-1.05965	-0.68409	1.965775	-1
2	-1.15823	0.877737	1.548718	0.403034	-0.40719	0.095921	0.592941	-0.27053	0.817739	0.753074	-0.82284	0.538196	1.345852	-1.11967	0.175121	-0.45145	-0.23703	-0.03819	0.8
2	-0.42597	0.960523	1.141109	-0.16825	0.420987	-0.02973	0.476201	0.260314	-0.56867	-0.37141	1.341262	0.359894	-0.35809	-0.13713	0.517617	0.401726	-0.05813	0.068653	-0
4	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.00516	0.081213	0.46496	-0.09925	-1.41691	-0.15383	-0.75106	0.167372	0.050144	-0.44359	0.002821	-0.61199	-0
7	-0.64427	1.417964	1.07438	-0.4922	0.948934	0.428118	1.120631	-3.80786	0.615375	1.249376	-0.61947	0.291474	1.757964	-1.32387	0.686133	-0.07613	-1.22213	-0.35822	0.3
7	-0.89429	0.286157	-0.11319	-0.27153	2.669599	3.721818	0.370145	0.851084	-0.39205	-0.41043	-0.70512	-0.11045	-0.28625	0.074355	-0.32878	-0.21008	-0.49977	0.118765	0.5
9	-0.33826	1.119593	1.044367	-0.22219	0.499361	-0.24676	0.651583	0.069539	-0.73673	-0.36685	1.017614	0.83639	1.006844	-0.44352	0.150219	0.739453	-0.54098	0.476677	0.4
10	1.449044	-1.17634	0.91386	-1.37567	-1.97138	-0.62915	-1.42324	0.048456	-1.72041	1.626659	1.199644	-0.67144	-0.51395	-0.09505	0.23093	0.031967	0.253415	0.854344	-0
10	0.384978	0.616109	-0.8743	-0.09402	2.924584	3.317027	0.470455	0.538247	-0.55889	0.309755	-0.25912	-0.32614	-0.09005	0.362832	0.928904	-0.12949	-0.80998	0.359985	0.7
10	1.249999	-1.22164	0.38393	-1.2349	-1.48542	-0.75323	-0.6894	-0.22749	-2.09401	1.323729	0.227666	-0.24268	1.205417	-0.31763	0.725675	-0.81561	0.873936	-0.84779	-0
11	1.069374	0.287722	0.828613	2.71252	-0.1784	0.337544	-0.09672	0.115982	-0.22108	0.46023	-0.77366	0.323387	-0.01108	-0.17849	-0.65556	-0.19993	0.124005	-0.9805	-0
12	-2.79185	-0.32777	1.64175	1.767473	-0.13659	0.807596	-0.42291	-1.90711	0.755713	1.151087	0.844555	0.792944	0.370448	-0.73498	0.406796	-0.30306	-0.15587	0.778265	2.2
12	-0.75242	0.345485	2.057323	-1.46864	-1.15839	-0.07785	-0.60858	0.003603	-0.43617	0.747731	-0.79398	-0.77041	1.047627	-1.0666	1.106953	1.660114	-0.27927	-0.41999	0.4
12	1.103215	-0.0403	1.267332	1.289091	-0.736	0.288069	-0.58606	0.18938	0.782333	-0.26798	-0.45031	0.936708	0.70838	-0.46865	0.354574	-0.24663	-0.00921	-0.59591	-0
13	-0.43691	0.918966	0.924591	-0.72722	0.915679	-0.12787	0.707642	0.087962	-0.66527	-0.73798	0.324098	0.277192	0.252624	-0.2919	-0.18452	1.143174	-0.92871	0.68047	0.0
14	-5.40126	-5.45015	1.186305	1.736239	3.049106	-1.76341	-1.55974	0.160842	1.23309	0.345173	0.91723	0.970117	-0.26657	-0.47913	-0.52661	0.472004	-0.72548	0.075081	-0

The dataset consists about the data's of the credit cards and the time in which the fraud hasbeen taken place with the attributes

- In the given dataset for Credit card fraud detection, the rows are aligned from V1 to V28 and with amount and class.
- The columns are aligned with integers from 0 to 114.
- The values in dataset are both positive and negative. In the amount column, the highest value is 3828 and the least value is 0.75.
- In the class column, the values are almost equal to zero's and one's(Boolean values) There are almost 284808 tuples present

2.BEGIN BUILDING THE PROJECT BY IMPORTING THE DATASET

importing the libraries

import numpy as np import pandas as pd

import time

import matplotlib.pyplot as plt matplotlib inline

import seaborn as sns

from scipy import stats

from scipy.stats import norm, skew from scipy.special import boxcoxip

from scipy.stats import boxcox_normmax

from sklearn import preprocessing

from sklearn.preprocessing import StandardScaler

import sklearn

from sklearn import metrics

from sklearn.metrics import roc curve, auc, roc auc_score

Fom sklearn.metrics import classification report, confusion matrix from sklearn.metrics Import average precision score, precision_recall_curve

from sklearn.model selection import train_test_split

from sklearn.model_selection import Stratified Fold

From sklearn.model selection import GridSearchCV, RandomizedSearchCV

from scipy.special import boxcoxip

from scipy.stats import boxcox_normmax

from sklearn import preprocessing

from sklearn.preprocessing import StandardScaler

import sklearn

from sklearn import metrics

from sklearn.metrics import roc_curve, auc, roc_auc_score

from sklearn.metrics import classification_report, confusion matrix

from sklearn.metrics import average precision score, precision_recall_curve

from sklearn.model_selection import train_test_split

from sklearn.model_selection import Stratified Fold from sklearn.model_selection import GridsearchCV, RandomizedSearchCV

from sklearn.linear_model import Ridge, Lasso, LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear_model import LogisticRegressionCV from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import AdaBoostClassifier from sklearn.ensemble import RandomForestClassifier

import xgboost as xgb

from xgboost import XGBClassifier

From sklearn.ensemble import AdaBoostClassifier

READING THE DATA FROM THE DATASET :

By using Google Colaboratory

Colab is a hosted Jupyter Notebook service that requires no setup to use and

provides free access to computing resources, including GPUs and TPUs. Colab is especially well suited to machine learning, data science, and education.

IMPORTING DATASET IN GOOGLE COLAB:

We can use google colaboratory by creating a file namely creditcard.csv and import them.

From google.colab import drive drive.mount("/content/gdrive")

Mounted at /content/gdrive

Importing dataset in any other softwares:

Df = pd.read_csv("gdrive/MyDrive/Colab Notebooks/creditcard.csv")

DISPLAYING ONLY THE FIRST FIVE DATAS FROM THE DATASET:

The head() method returns a specified number of rows, string from the top. The head() method returns the first 5 rows if a number is not specified. Note: The column names will also be returned, in addition to the specified rows.

Df.head()

Exploratory data analysis

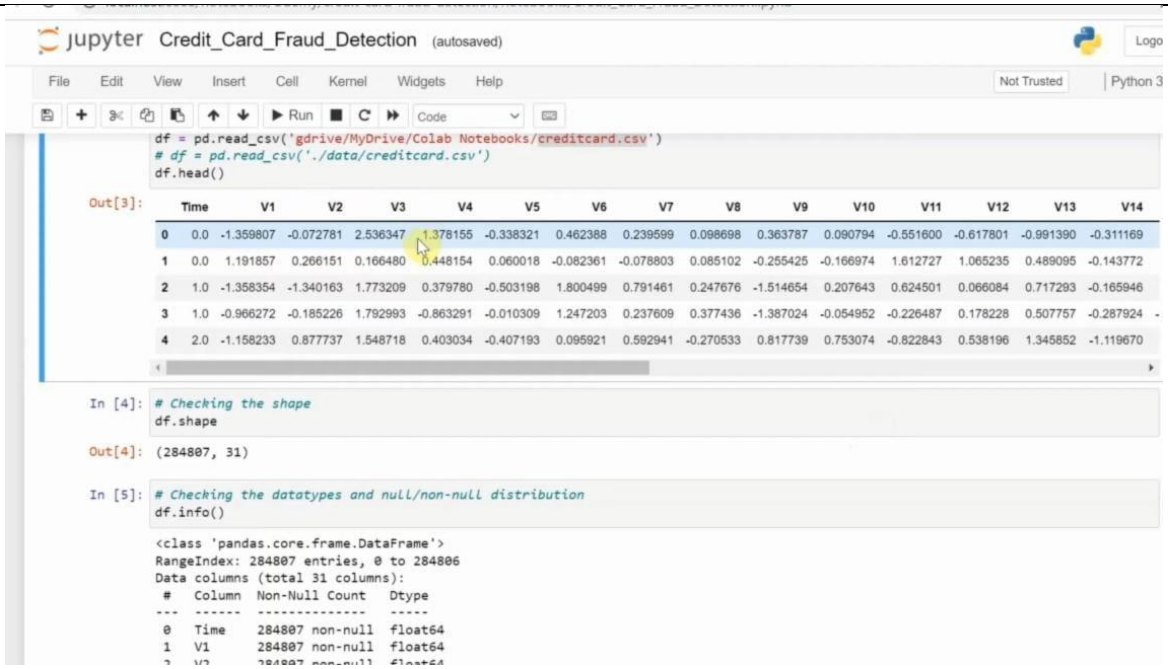
```
[ ] # Mounting the google drive
    from google.colab import drive
    drive.mount('/content/gdrive')

    Mounted at /content/gdrive

[ ] # Loading the data
    df = pd.read_csv('gdrive/MyDrive/Colab Notebooks/creditcard.csv')
    # df = pd.read_csv('./data/creditcard.csv')
    df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16
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
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
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
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
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852	-1.119670	0.175121	-0.451449

DISPLAYING THE INFORMATION ABOUT THE DATASET:

The info() method prints information about the DataFrame. The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values). Note: the info() method actually prints the info.

Df.info()



DISPLAYING THE DESCRIPTION ABOUT THE DATASET:

The describe() method returns description of the data in the DataFrame. If the DataFrame contains numerical data, the description contains these information for each column: count – The number of not-empty values. Mean – The average (mean) value.

Df.describe

```
In [6]: # Checking distribution of numerical values in the dataset
        df.describe()
```

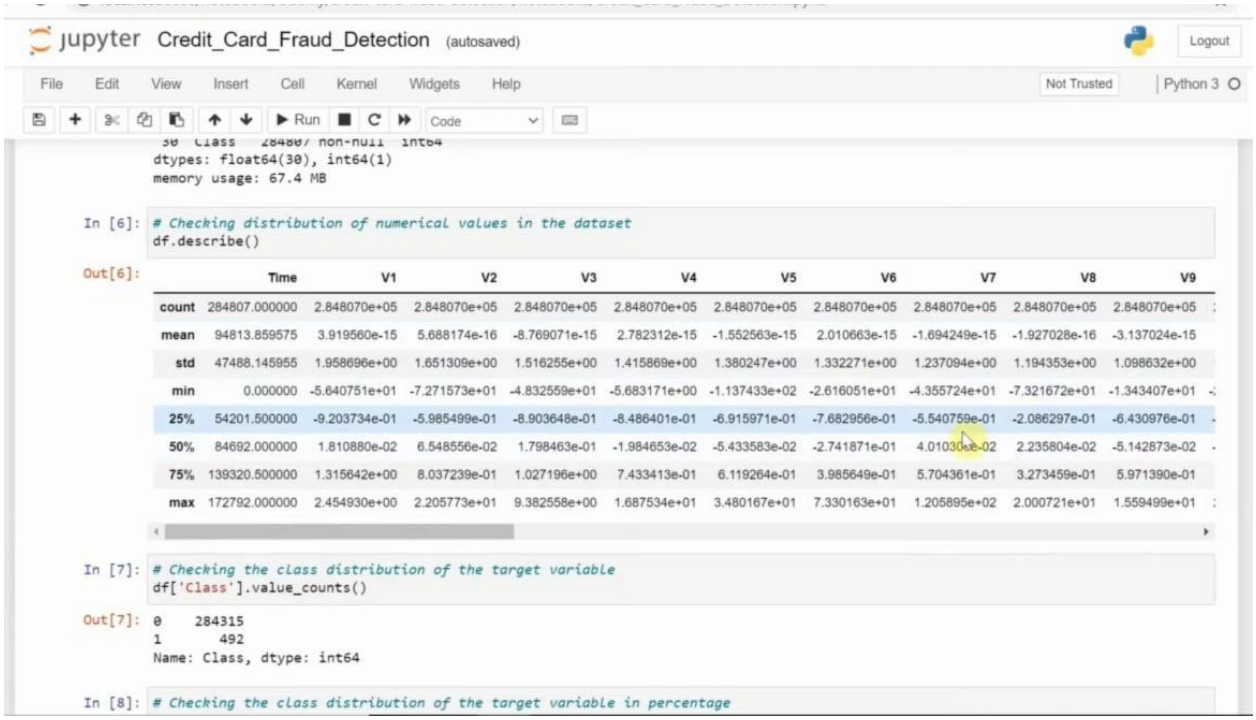
Out[6]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	3.919560e-15	5.688174e-16	-8.769071e-15	2.782312e-15	-1.552563e-15	2.010663e-15	-1.694249e-15	-1.927028e-16	-3.137024e-15
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01

GETTING THE INDEPENDENT VARIABLE FROM THE DATASET:

To set the difference between independent and dependent variables we need to consider the class column as y variable

Df['class'].value_count()



Where, Value_count() defines about the value_counts() function returns a Series that contain counts of unique values. It returns an object that will be in descending order so that its first element will be the most frequently-occurred element. By default, it excludes NA values.

3.PREPROCESS DATASET PERFORMING:

FEATURE ENGINEERING :

Feature engineering refers to manipulation — addition, deletion, combination, mutation — of your data set to improve model training, leading to better performance and greater accuracy. Effective feature engineering is based on sound knowledge of the business problem and the available data sources.

SPLIT TRAIN AND TEST DATA :

Splitting the dataset into x and y

```
Y=df['class']

X=df.drop(['class'],axis=1)
```

Checking some rows of x

```
x.head()
```

Checking some rows of y

```
y. head()
```

Splitting The Dataset Using Train Test Split :

```
X_train,x_test,y_train,y_test=train_test_split(x,y,random_state=100,test_size=0.20)
```

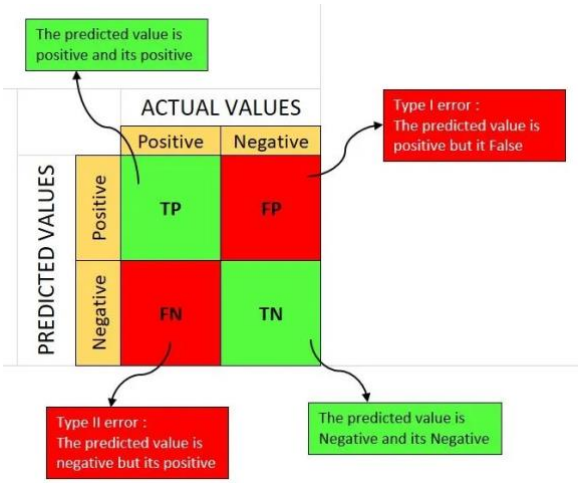
Preserve x_test and y_ test to evaluate

Checking The Spread Of Data Post Split :

Print(np sum(y))

Print(np sum(y_train))

Print(np sum(y_test))



CONFUSION MATRIX :

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes.

The following 4 are the basic terminology which will help us in determining the metrics we are looking for.

True Positives (TP): when the actual value is Positive and predicted is also Positive.

True Negatives (TN): when the actual value is Negative and prediction is also Negative.

False Positives (FP): When the actual is negative but prediction is Positive. Also known as the Type 1 error

False Negatives (FN): When the actual is Positive but the prediction is Negative. Also known as the Type 2 error.

4. PERFORMING DIFFERENT ANALYSIS :

IMORTING DEPENDIES:

- Python
- Pandas
- Xgboost

- Scikit learn
- Matplotlib
- Seaborn
- Scipy
- Statsmodels

PYTHON:

The objective of Credit Card Fraud Detection is to accurately identify fraudulent transactions from a large pool of credit card transactions by building a predictive model based on past transaction data. The aim is to detect all fraudulent transactions with minimum false alarms.

Syntax: import numpy as np

PANDAS:

Pandas is a Python package built for a broad range of data analysis and manipulation including tabular data, time series and many types of data sets.

Import pandas as pd, read the credit card data in and assign it to df.

Use .info() to print information about df.

Use .value_counts() to get the count of fraudulent and non-fraudulent transactions in the 'Class' column. Assign the result to occ.

Get the ratio of fraudulent transactions over the total number of transactions in the dataset.

Syntax: import pandas as pd

Df = pd.read_csv(cc3_file

Df..info()mport pandas as pd

XGBOOST :

Although many traditional approaches are already available for fraud transaction prediction, existing methods lack accuracy, and it can be increased by ensemble techniques such as XGBoost.

Syntax: import xgboost as xbg

SCIKIT LEARN :

Scikit-learn is an open source data analysis library, and the gold standard for Machine Learning (ML) in the Python ecosystem.

MATPLOTLIB:

Matplotlib is a cross-platform, data visualization and graphical plotting library (histograms, scatter plots, bar charts, etc) for Python and its numerical extension NumPy.

Syntax: import matplotlib.pyplot as plt

SEABORN:

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Syntax: import seaborn as sns

Example:

```
import matplotlib.pyplot as plt

import seaborn as sns
```

```
Sns.distplot([0, 1, 2, 3, 4, 5], hist=False)

Plt.show()
```

SCIPY:

SciPy is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing.

Syntax: from scipy import special

Example:

```
import numpy as np

From scipy import io as sio

Array = np.ones((4, 4))

Sio.savemat('example.mat', {'ar': array})

Data=sio.loadmat('example.mat', struct_as_record=True)

Data['ar']
```

STATSMODELS:

Statsmodels is a Python package that allows users to explore data, estimate statistical models, and perform statistical tests. An extensive list of descriptive statistics, statistical tests, plotting

Syntax: import statsmodels. Api as sm

Import statsmodels.api as sm

>>> import numpy as np

>>>duncan_prestige=sm.datasets.get_rdataset(“Duncan”, “carData”)

>>> Y = duncan_prestige.data[‘income’]

>>> X = duncan_prestige.data[‘education’]

>>> X = sm.add_constant(X)

>>> model = sm.OLS(Y,X)

>>> results = model.fit()

>>> results.params

Const 10.603498

Education 0.594859

Dtype: float64

HANDLING THE MISSING DATA :

Missing values can affect the performance of the anomaly Detection algorithm. Therefore, it is essential to check whether there are anyMissing values in the dataset and take appropriate action.

Check if there are any missing values in the dataset

Syntax: Print(df.isnull().sum())

Example:

Data.isnull().sum()

Time	**0**
V1	0
V2	0
V3	0
V4	0
V5	0
V6	0
V7	0
V8	0
V9	0
V10	0
V11	0
V12	0
V13	0
V14	0
V15	0
V16	0
V17	0
V18	0

V19 0

V20 0

V21 0

V22 0

V23 0

V24 0

V25 0

V26 0

V27 0

V28 0

Amount 0

Class 0

Dtype: int64

From the above table – There are no missing values in the dataset.

CATEGORICAL DATA :

Categorical data refers to a form of information that can be stored and identified based on their names or labels. It is a type of qualitative data that can be grouped into categories instead of being measured numerically.

In the given dataset, there is no categorical data. If there is categorical data,it must first be transformed into numerical data

SKLEARN HELPS IN MISSING VALUES:

The scikit-learn library works only with arrays, thus when performing every operation, a dataframe column must be converted to an array.

This can be achieved through the numpy.array() function, which receives the dataframe column as input. In addition, the fit() function receives as input an array of arrays, each representing a sample of the dataset. Thus the reshape() function could be used to convert a standard array to an array of arrays.

CONFUSION MATRIX:

Confusion matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance.

```
Confusion_matrix_dt=confusion_matrix(test_Y,predictions_dt.round())

Print("Confusion Matrix – Decision Tree")

Print(confusion_matrix_dt)

Plot_confusion_matrix(confusion_matrix_dt, classes=[0, 1], title= "Confusion Matrix – Decision Tree")
```