**WEEK 3**

**AIM:**

Write a program to implement a SVM model to perform classification on a data stored in a .CSV file

**DESCRIPTION:**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

**Hyperplane:** There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

**Support Vectors:** The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

**SVM Kernel:**

The SVM kernel is a function that takes low dimensional input space and transforms it into higher-dimensional space, ie it converts not separable problem to separable problem. It is mostly useful in non-linear separation problems. Simply put the kernel, it does some extremely complex data transformations then finds out the process to separate the data based on the labels or outputs defined.

**Advantages of SVM:**

* Effective in high dimensional cases
* Its memory efficient as it uses a subset of training points in the decision function called support vectors
* Different kernel functions can be specified for the decision functions and its possible to specify custom kernels

**CODE:**

import pandas as pd

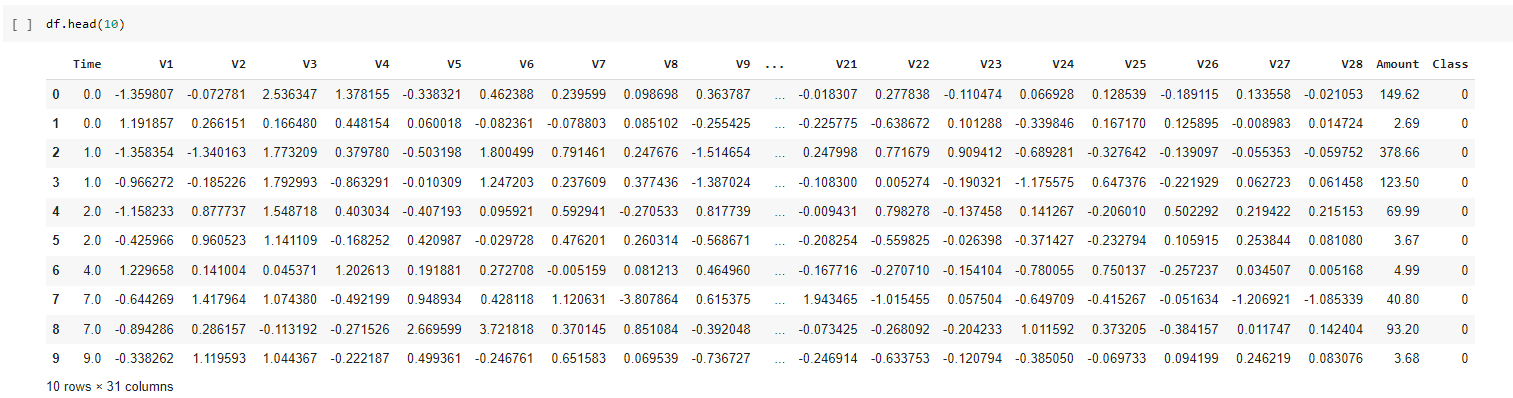
import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv("/content/creditcard.csv")

df.head(10)



count\_class=pd.value\_counts(df["Class"], sort= True)

count\_class.plot(kind= 'bar')

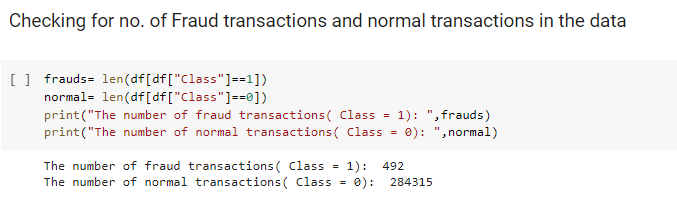


frauds= len(df[df["Class"]==1])

normal= len(df[df["Class"]==0])

print("The number of fraud transactions( Class = 1): ",frauds)

print("The number of normal transactions( Class = 0): ",normal)



#getting the indices of normal and fraud transactions

fraud\_index= np.array(df[df["Class"]==1].index)

normal\_index= df[df["Class"]==0].index

#replace =false meaning no duplicate values.

random\_normal\_indices= np.random.choice(normal\_index, 500, replace= False)

random\_normal\_indices= np.array(random\_normal\_indices)

new\_indices= np.concatenate([fraud\_index, random\_normal\_indices])

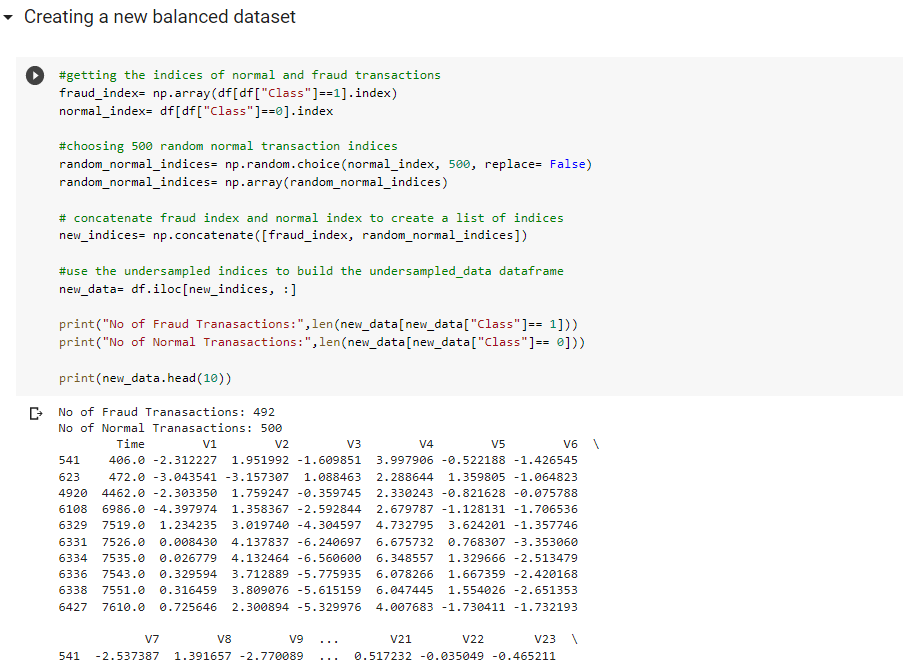
#use the undersampled indices to build the undersampled\_data dataframe

new\_data= df.iloc[new\_indices, :]

print("No of Fraud Tranasactions:",len(new\_data[new\_data["Class"]== 1]))

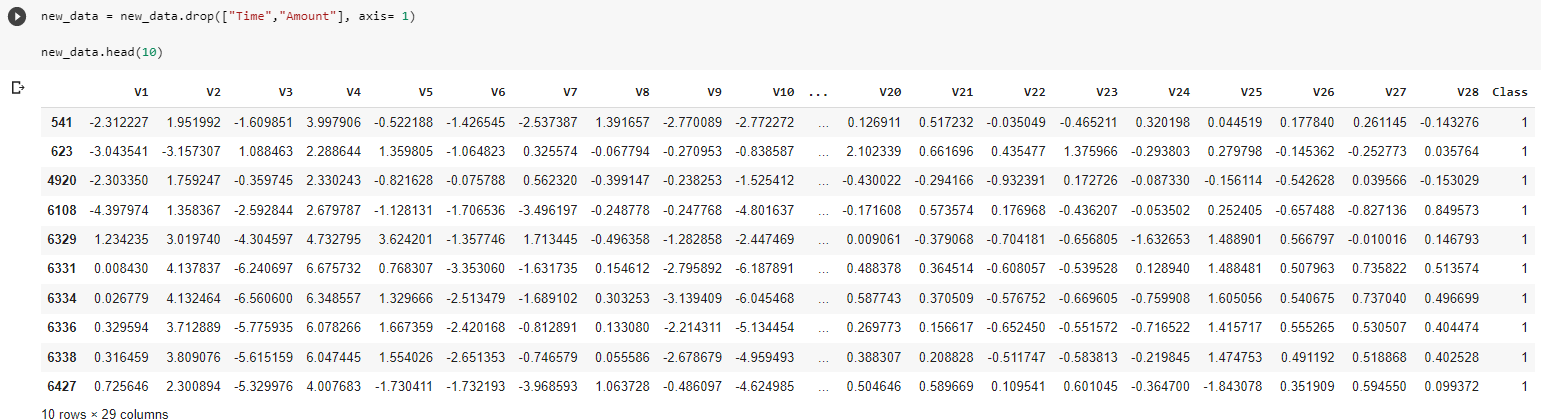
print("No of Normal Tranasactions:",len(new\_data[new\_data["Class"]== 0]))

print(new\_data.head(10))



new\_data = new\_data.drop(["Time","Amount"], axis= 1)

new\_data.head(10)



X = new\_data.iloc[:,new\_data.columns != 'Class'].values

Y = new\_data.iloc[:,new\_data.columns == 'Class'].values

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

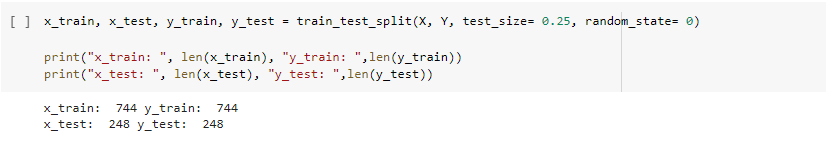
from mlxtend.plotting import plot\_confusion\_matrix

from sklearn.metrics import confusion\_matrix

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size= 0.25, random\_state= 0)

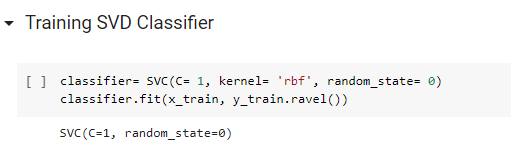
print("x\_train: ", len(x\_train), "y\_train: ",len(y\_train))

print("x\_test: ", len(x\_test), "y\_test: ",len(y\_test))



classifier= SVC(C= 1, kernel= 'rbf', random\_state= 0)

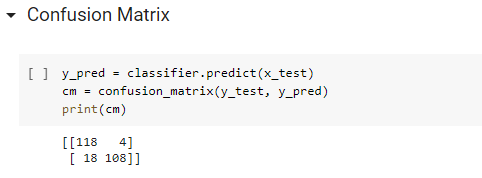
classifier.fit(x\_train, y\_train.ravel())



y\_pred = classifier.predict(x\_test)

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)



from sklearn import metrics

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

