#Code in Python

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

%matplotlib inline

data = pd.read\_csv("creditcard.csv")

print ("Fraud")

print (df.Amount[df.Class == 1].describe())

print ()

print ("Normal")

print (df.Amount[df.Class == 0].describe())

f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(12,4))

bins = 30

ax1.hist(df.Amount[df.Class == 1], bins = bins)

ax1.set\_title('Fraud')

ax2.hist(df.Amount[df.Class == 0], bins = bins)

ax2.set\_title('Normal')

plt.xlabel('Amount ($)')

plt.ylabel('Number of Transactions')

plt.yscale('log')

plt.show()

count\_classes = pd.value\_counts(data['Class'], sort = True).sort\_index()

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

plt.title("Fraud class histogram")

plt.xlabel("Class")

plt.ylabel("Frequency")

def convert\_totime(seconds):

return datetime.datetime.fromtimestamp(seconds);

timeAnalysis = creditCardData[['Time', 'Amount', 'Class']].copy()

timeAnalysis['datetime'] = timeAnalysis.Time.apply(convert\_totime)

timeAnalysis['hour of the day'] = timeAnalysis.datetime.dt.hour

timeAnalysisGrouped = timeAnalysis.groupby(['Class', 'hour of the day'])['Amount'].count()

plt.figure(figsize = (10, 6))

validTransactions = timeAnalysisGrouped[0].copy()

validTransactions.name = 'Number of transactions'

validTransactions.plot.bar(title = '# of legitimate credit card transactions per hour', legend = True)

timeDelta = datetime.datetime.utcnow() - datetime.datetime.now()

plt.figure(figsize = (10, 6))

timeAnalysis['hour of the day'] = timeAnalysis.datetime + timeDelta

timeAnalysis['hour of the day'] = timeAnalysis['hour of the day'].dt.hour

timeAnalysisGrouped = timeAnalysis.groupby(['Class', 'hour of the day'])['Amount'].count()

validTransactions = timeAnalysisGrouped[0].copy()

validTransactions.name = 'Number of transactions'

validTransactions.plot.bar(title = '# of legitimate credit card transactions per hour', legend = True)

plt.figure(figsize = (10, 6))

fraudTransactions = timeAnalysisGrouped[1].copy()

fraudTransactions.name = 'Number of transactions'

fraudTransactions.plot.bar(title = '# of fraud credit card transactions per hour', legend = True)

#Data smoting function

import random

from sklearn.neighbors import NearestNeighbors

import numpy as np

class Smote:

def \_\_init\_\_(self,samples,N=10,k=5):

self.n\_samples,self.n\_attrs=samples.shape

self.N=N

self.k=k

self.samples=samples

self.newindex=0

# self.synthetic=np.zeros((self.n\_samples\*N,self.n\_attrs))

def over\_sampling(self):

N=int(self.N/100)

self.synthetic = np.zeros((self.n\_samples \* N, self.n\_attrs))

neighbors=NearestNeighbors(n\_neighbors=self.k).fit(self.samples)

print 'neighbors',neighbors

for i in range(len(self.samples)):

nnarray=neighbors.kneighbors(self.samples[i].reshape(1,-1),return\_distance=False)[0]

#print nnarray

self.\_populate(N,i,nnarray)

return self.synthetic

# for each minority class samples,choose N of the k nearest neighbors and generate N synthetic samples.

def \_populate(self,N,i,nnarray):

for j in range(N):

nn=random.randint(0,self.k-1)

dif=self.samples[nnarray[nn]]-self.samples[i]

gap=random.random()

self.synthetic[self.newindex]=self.samples[i]+gap\*dif

self.newindex+=1

a=np.array([[1,2,3],[4,5,6],[2,3,1],[2,1,2],[2,3,4],[2,3,4]])

s=Smote(a,N=100)

print s.over\_sampling()

#Start Data Smoting

from imblearn.over\_sampling import SMOTE

from sklearn.cross\_validation import train\_test\_split

def data\_prepration(x):

x\_features= x.ix[:,x.columns != "Class"]

x\_labels=x.ix[:,x.columns=="Class"] x\_features\_train,x\_features\_test,x\_labels\_train,x\_labels\_test = train\_test\_split(x\_features,x\_labels,test\_size=0.3)

print("length of training data")

print(len(x\_features\_train))

print("length of test data")

print(len(x\_features\_test))

return(x\_features\_train,x\_features\_test,x\_labels\_train,x\_labels\_test)

data\_train\_X,data\_test\_X,data\_train\_y,data\_test\_y=data\_prepration(df)

print pd.value\_counts(data\_test\_y['Class'])

#Using smote

os = SMOTE(random\_state=0)

os\_data\_X,os\_data\_y=os.fit\_sample(data\_train\_X.values,data\_train\_y.values.ravel())

from \_\_future\_\_ import division

columns = data\_train\_X.columns

os\_data\_X = pd.DataFrame(data=os\_data\_X,columns=columns )

print len(os\_data\_X)

os\_data\_y= pd.DataFrame(data=os\_data\_y,columns=["Class"])

# Now displayed statistics with oversampled data

print("length of oversampled data is ",len(os\_data\_X))

print("Number of normal transcation",len(os\_data\_y[os\_data\_y["Class"]==0]))

print("Number of fraud transcation",len(os\_data\_y[os\_data\_y["Class"]==1]))

print("Proportion of Normal data in oversampled data is ",len(os\_data\_y[os\_data\_y["Class"]==0])/len(os\_data\_X))

print("Proportion of fraud data in oversampled data is ",len(os\_data\_y[os\_data\_y["Class"]==1])/len(os\_data\_X))