In [ ]:

*## Name : Satyam kumar ## Roll no:I4250*

*## Subject:LP-IV(DL)*

In [1]:

**import** pandas **as** pd **import** numpy **as** np **import** tensorflow **as** tf

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

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

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.metrics **import** confusion\_matrix, recall\_score, accuracy\_score, precision\_score

RANDOM\_SEED **=** 2021

TEST\_PCT **=** 0.3

LABELS **=** ["Normal","Fraud"]

In [2]:

dataset **=** pd**.**read\_csv("creditcard.csv")

In [3]:

*#check for any null values*

print("Any nulls in the dataset",dataset**.**isnull()**.**values**.**any()) print(' ')

print("No. of unique labels",len(dataset['Class']**.**unique())) print("Label values",dataset**.**Class**.**unique())

*#0 is for normal credit card transcation*

*#1 is for fraudulent credit card transcation*

print(' ')

print("Break down of Normal and Fraud Transcations") print(pd**.**value\_counts(dataset['Class'],sort**=True**))

Any nulls in the dataset False

No. of unique labels 2 Label values [0 1]

In [4]:

Break down of Normal and Fraud Transcations 0 284315

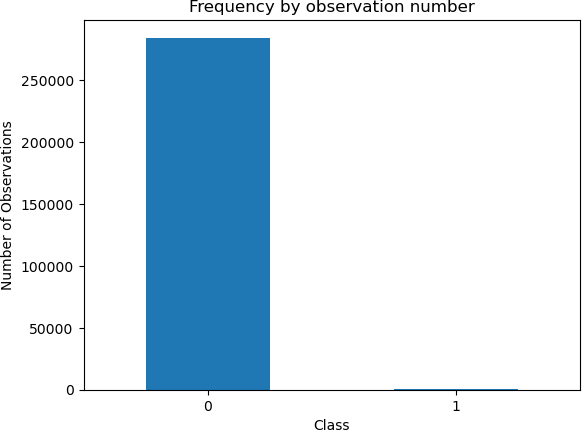
1 492

Name: Class, dtype: int64

*#visualizing the imbalanced dataset*

count\_classes **=** pd**.**value\_counts(dataset['Class'],sort**=True**) count\_classes**.**plot(kind**=**'bar',rot**=**0) plt**.**xticks(range(len(dataset['Class']**.**unique())),dataset**.**Class**.**unique()) plt**.**title("Frequency by observation number")

plt**.**xlabel("Class") plt**.**ylabel("Number of Observations")

Out[4]: Text(0, 0.5, 'Number of Observations')

In [4]:

*#Save the normal and fradulent transcations in seperate dataframe*

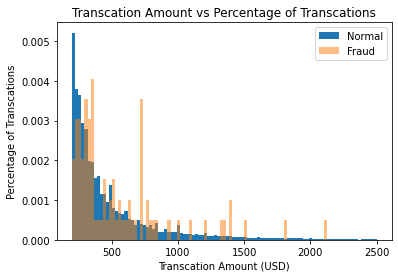
normal\_dataset **=** dataset[dataset**.**Class **==** 0] fraud\_dataset **=** dataset[dataset**.**Class **==** 1]

*#Visualize transcation amounts for normal and fraudulent transcations*

bins **=** np**.**linspace(200,2500,100) plt**.**hist(normal\_dataset**.**Amount,bins**=**bins,alpha**=**1,density**=True**,label**=**'Normal') plt**.**hist(fraud\_dataset**.**Amount,bins**=**bins,alpha**=**0.5,density**=True**,label**=**'Fraud') plt**.**legend(loc**=**'upper right')

plt**.**title("Transcation Amount vs Percentage of Transcations") plt**.**xlabel("Transcation Amount (USD)")

plt**.**ylabel("Percentage of Transcations") plt**.**show()



In [5]:

dataset

Out[5]: **Time V1 V2 V3 V4 V5 V6 V7 V8 V9 ... 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 ... -0.018307 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 ... -0.225775 -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 ... 0.247998 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 ... -0.108300 0.005274 -0.190321 -1.175575 0.647376 -0.221929 0.062723 0.061458 | 123.50 | 0 |
| **4** | 2.0 -1.158233 0.877737 1.548718 0.403034 -0.407193 0.095921 0.592941 -0.270533 0.817739 ... -0.009431 0.798278 -0.137458 0.141267 -0.206010 0.502292 0.219422 0.215153 | 69.99 | 0 |

**...** ...

...

...

...

...

...

...

...

...

... ...

...

...

...

...

...

...

...

... ... ...

|  |  |  |
| --- | --- | --- |
| **284802** 172786.0 -11.881118 10.071785 -9.834783 -2.066656 -5.364473 -2.606837 -4.918215 7.305334 1.914428 ... 0.213454 0.111864 1.014480 -0.509348 1.436807 0.250034 0.943651 0.823731 | 0.77 | 0 |
| **284803** 172787.0 -0.732789 -0.055080 2.035030 -0.738589 0.868229 1.058415 0.024330 0.294869 0.584800 ... 0.214205 0.924384 0.012463 -1.016226 -0.606624 -0.395255 0.068472 -0.053527 | 24.79 | 0 |
| **284804** 172788.0 1.919565 -0.301254 -3.249640 -0.557828 2.630515 3.031260 -0.296827 0.708417 0.432454 ... 0.232045 0.578229 -0.037501 0.640134 0.265745 -0.087371 0.004455 -0.026561 | 67.88 | 0 |
| **284805** 172788.0 -0.240440 0.530483 0.702510 0.689799 -0.377961 0.623708 -0.686180 0.679145 0.392087 ... 0.265245 0.800049 -0.163298 0.123205 -0.569159 0.546668 0.108821 0.104533 | 10.00 | 0 |
| **284806** 172792.0 -0.533413 -0.189733 0.703337 -0.506271 -0.012546 -0.649617 1.577006 -0.414650 0.486180 ... 0.261057 0.643078 0.376777 0.008797 -0.473649 -0.818267 -0.002415 0.013649  284807 rows × 31 columns | 217.00 | 0 |

In [6]:

sc **=** StandardScaler()

dataset['Time'] **=** sc**.**fit\_transform(dataset['Time']**.**values**.**reshape(**-**1,1)) dataset['Amount'] **=** sc**.**fit\_transform(dataset['Amount']**.**values**.**reshape(**-**1,1))

In [7]:

raw\_data **=** dataset**.**values

*#The last element contains if the transcation is normal which is represented by 0 and if fraud then 1*

labels **=** raw\_data[:,**-**1]

*#The other data points are the electrocadriogram data*

data **=** raw\_data[:,0:**-**1]

train\_data,test\_data,train\_labels,test\_labels **=** train\_test\_split(data,labels,test\_size **=** 0.2,random\_state **=**2021)

In [8]:

min\_val **=** tf**.**reduce\_min(train\_data) max\_val **=** tf**.**reduce\_max(train\_data)

train\_data **=** (train\_data **-** min\_val) **/** (max\_val **-** min\_val) test\_data **=** (test\_data **-** min\_val) **/** (max\_val **-** min\_val)

train\_data **=** tf**.**cast(train\_data,tf**.**float32) test\_data **=** tf**.**cast(test\_data,tf**.**float32)

In [9]:

train\_labels **=** train\_labels**.**astype(bool) test\_labels **=** test\_labels**.**astype(bool)

*#Creating normal and fraud datasets* normal\_train\_data **=** train\_data[**~**train\_labels] normal\_test\_data **=** test\_data[**~**test\_labels]

fraud\_train\_data **=** train\_data[train\_labels] fraud\_test\_data **=** test\_data[test\_labels]

print("No. of records in Fraud Train Data=",len(fraud\_train\_data)) print("No. of records in Normal Train Data=",len(normal\_train\_data)) print("No. of records in Fraud Test Data=",len(fraud\_test\_data)) print("No. of records in Normal Test Data=",len(normal\_test\_data))

In [10]:

No. of records in Fraud Train Data= 389

No. of records in Normal Train Data= 227456 No. of records in Fraud Test Data= 103

No. of records in Normal Test Data= 56859

nb\_epoch **=** 50

batch\_size **=** 64

input\_dim **=** normal\_train\_data**.**shape[1]

*#num of columns,30*

encoding\_dim **=** 14

hidden\_dim1 **=** int(encoding\_dim **/** 2) hidden\_dim2 **=** 4

learning\_rate **=** 1e-7

In [11]:

*#input layer*

input\_layer **=** tf**.**keras**.**layers**.**Input(shape**=**(input\_dim,))

*#Encoder*

encoder **=** tf**.**keras**.**layers**.**Dense(encoding\_dim,activation**=**"tanh",activity\_regularizer **=** tf**.**keras**.**regularizers**.**l2(learning\_rate))(input\_layer) encoder **=** tf**.**keras**.**layers**.**Dropout(0.2)(encoder)

encoder **=** tf**.**keras**.**layers**.**Dense(hidden\_dim1,activation**=**'relu')(encoder)

encoder **=** tf**.**keras**.**layers**.**Dense(hidden\_dim2,activation**=**tf**.**nn**.**leaky\_relu)(encoder)

*#Decoder*

decoder **=** tf**.**keras**.**layers**.**Dense(hidden\_dim1,activation**=**'relu')(encoder) decoder **=** tf**.**keras**.**layers**.**Dropout(0.2)(decoder)

decoder **=** tf**.**keras**.**layers**.**Dense(encoding\_dim,activation**=**'relu')(decoder) decoder **=** tf**.**keras**.**layers**.**Dense(input\_dim,activation**=**'tanh')(decoder)

*#Autoencoder*

autoencoder **=** tf**.**keras**.**Model(inputs **=** input\_layer,outputs **=** decoder) autoencoder**.**summary()

Model: "model"

Layer (type) Output Shape Param #

=================================================================

|  |  |  |
| --- | --- | --- |
| input\_1 (InputLayer) | [(None, 30)] | 0 |
| dense (Dense) | (None, 14) | 434 |
| dropout (Dropout) | (None, 14) | 0 |
| dense\_1 (Dense) | (None, 7) | 105 |
| dense\_2 (Dense) | (None, 4) | 32 |
| dense\_3 (Dense) | (None, 7) | 35 |
| dropout\_1 (Dropout) | (None, 7) | 0 |
| dense\_4 (Dense) | (None, 14) | 112 |
| dense\_5 (Dense) | (None, 30) | 450 |

=================================================================

Total params: 1,168

Trainable params: 1,168

Non-trainable params: 0

In [12]:

cp **=** tf**.**keras**.**callbacks**.**ModelCheckpoint(filepath**=**"autoencoder\_fraud.h5",mode**=**'min',monitor**=**'val\_loss',verbose**=**2,save\_best\_only**=True**)

*#Define our early stopping*

early\_stop **=** tf**.**keras**.**callbacks**.**EarlyStopping( monitor**=**'val\_loss', min\_delta**=**0.0001, patience**=**10,

verbose**=**11, mode**=**'min',

restore\_best\_weights**=True**

)

In [13]:

autoencoder**.**compile(metrics**=**['accuracy'],loss**=** 'mean\_squared\_error',optimizer**=**'adam')

In [14]:

history **=** autoencoder**.**fit(normal\_train\_data,normal\_train\_data,epochs **=** nb\_epoch,

batch\_size **=** batch\_size,shuffle **= True**, validation\_data **=** (test\_data,test\_data), verbose**=**1,

callbacks **=** [cp,early\_stop])**.**history

In [15]:

Epoch 1/50

3543/3554 [============================>.] - ETA: 0s - loss: 0.0033 - accuracy: 0.0372

Epoch 1: val\_loss improved from inf to 0.00002, saving model to autoencoder\_fraud.h5

3554/3554 [==============================] - 11s 2ms/step - loss: 0.0033 - accuracy: 0.0372 - val\_loss: 2.0179e-05 - val\_accuracy: 0.0343 Epoch 2/50

3545/3554 [============================>.] - ETA: 0s - loss: 1.9583e-05 - accuracy: 0.0609

Epoch 2: val\_loss did not improve from 0.00002

3554/3554 [==============================] - 6s 2ms/step - loss: 1.9599e-05 - accuracy: 0.0609 - val\_loss: 2.0190e-05 - val\_accuracy: 0.0078 Epoch 3/50

3517/3554 [============================>.] - ETA: 0s - loss: 1.9562e-05 - accuracy: 0.0619

Epoch 3: val\_loss improved from 0.00002 to 0.00002, saving model to autoencoder\_fraud.h5

3554/3554 [==============================] - 7s 2ms/step - loss: 1.9580e-05 - accuracy: 0.0619 - val\_loss: 2.0025e-05 - val\_accuracy: 0.0420 Epoch 4/50

3527/3554 [============================>.] - ETA: 0s - loss: 1.9545e-05 - accuracy: 0.0599

Epoch 4: val\_loss did not improve from 0.00002

3554/3554 [==============================] - 6s 2ms/step - loss: 1.9530e-05 - accuracy: 0.0601 - val\_loss: 2.0277e-05 - val\_accuracy: 0.2168 Epoch 5/50

3549/3554 [============================>.] - ETA: 0s - loss: 1.8826e-05 - accuracy: 0.1758

Epoch 5: val\_loss improved from 0.00002 to 0.00002, saving model to autoencoder\_fraud.h5

3554/3554 [==============================] - 6s 2ms/step - loss: 1.8831e-05 - accuracy: 0.1759 - val\_loss: 1.8344e-05 - val\_accuracy: 0.2184 Epoch 6/50

3539/3554 [============================>.] - ETA: 0s - loss: 1.7526e-05 - accuracy: 0.2362

Epoch 6: val\_loss improved from 0.00002 to 0.00002, saving model to autoencoder\_fraud.h5

3554/3554 [==============================] - 8s 2ms/step - loss: 1.7518e-05 - accuracy: 0.2363 - val\_loss: 1.7095e-05 - val\_accuracy: 0.3538 Epoch 7/50

3516/3554 [============================>.] - ETA: 0s - loss: 1.8826e-05 - accuracy: 0.1125

Epoch 7: val\_loss did not improve from 0.00002

3554/3554 [==============================] - 5s 2ms/step - loss: 1.8813e-05 - accuracy: 0.1137 - val\_loss: 1.7990e-05 - val\_accuracy: 0.2041 Epoch 8/50

3529/3554 [============================>.] - ETA: 0s - loss: 1.7206e-05 - accuracy: 0.2141

Epoch 8: val\_loss did not improve from 0.00002

3554/3554 [==============================] - 5s 1ms/step - loss: 1.7328e-05 - accuracy: 0.2143 - val\_loss: 1.7153e-05 - val\_accuracy: 0.2781 Epoch 9/50

3518/3554 [============================>.] - ETA: 0s - loss: 1.6837e-05 - accuracy: 0.2481

Epoch 9: val\_loss improved from 0.00002 to 0.00002, saving model to autoencoder\_fraud.h5

3554/3554 [==============================] - 6s 2ms/step - loss: 1.6825e-05 - accuracy: 0.2482 - val\_loss: 1.6811e-05 - val\_accuracy: 0.3518 Epoch 10/50

3516/3554 [============================>.] - ETA: 0s - loss: 1.6692e-05 - accuracy: 0.2509

Epoch 10: val\_loss improved from 0.00002 to 0.00002, saving model to autoencoder\_fraud.h5

3554/3554 [==============================] - 6s 2ms/step - loss: 1.6682e-05 - accuracy: 0.2510 - val\_loss: 1.6469e-05 - val\_accuracy: 0.3492 Epoch 11/50

3526/3554 [============================>.] - ETA: 0s - loss: 1.6569e-05 - accuracy: 0.2484

Epoch 11: val\_loss improved from 0.00002 to 0.00002, saving model to autoencoder\_fraud.h5 Restoring model weights from the end of the best epoch: 1.

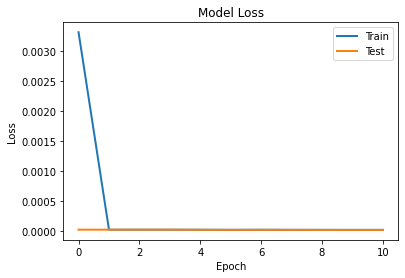
3554/3554 [==============================] - 7s 2ms/step - loss: 1.6561e-05 - accuracy: 0.2484 - val\_loss: 1.6237e-05 - val\_accuracy: 0.2865 Epoch 11: early stopping

plt**.**plot(history['loss'],linewidth **=** 2,label **=** 'Train') plt**.**plot(history['val\_loss'],linewidth **=** 2,label **=** 'Test') plt**.**legend(loc**=**'upper right')

plt**.**title('Model Loss') plt**.**ylabel('Loss') plt**.**xlabel('Epoch')

*#plt.ylim(ymin=0.70,ymax=1)*

plt**.**show()



In [16]:

test\_x\_predictions **=** autoencoder**.**predict(test\_data)

mse **=** np**.**mean(np**.**power(test\_data **-** test\_x\_predictions, 2),axis **=** 1) error\_df **=** pd**.**DataFrame({'Reconstruction\_error':mse,

'True\_class':test\_labels})

1781/1781 [==============================] - 2s 865us/step

In [17]:

threshold\_fixed **=** 50

groups **=** error\_df**.**groupby('True\_class') fig,ax **=** plt**.**subplots()

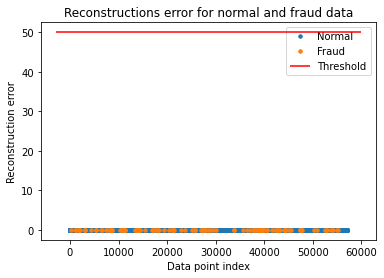
**for** name,group **in** groups:

ax**.**plot(group**.**index,group**.**Reconstruction\_error,marker**=**'o',ms **=** 3.5,linestyle**=**'', label **=** "Fraud" **if** name**==**1 **else** "Normal")

ax**.**hlines(threshold\_fixed,ax**.**get\_xlim()[0],ax**.**get\_xlim()[1],colors**=**"r",zorder**=**100,label**=**"Threshold") ax**.**legend()

plt**.**title("Reconstructions error for normal and fraud data")

plt**.**ylabel("Reconstruction error") plt**.**xlabel("Data point index") plt**.**show()



In [19]:

threshold\_fixed **=** 52

pred\_y **=** [1 **if** e **>** threshold\_fixed **else** 0

**for** e **in**

error\_df**.**Reconstruction\_error**.**values] error\_df['pred'] **=** pred\_y

conf\_matrix **=** confusion\_matrix(error\_df**.**True\_class,pred\_y)

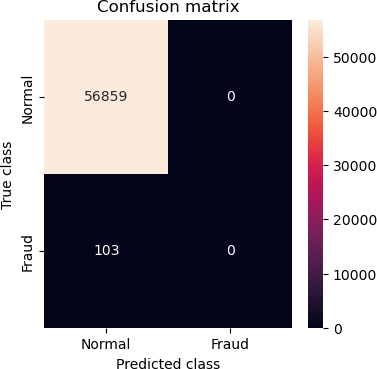
plt**.**figure(figsize **=** (4,4))

sns**.**heatmap(conf\_matrix,xticklabels **=** LABELS,yticklabels **=** LABELS,annot **= True**,fmt**=**"d") plt**.**title("Confusion matrix")

plt**.**ylabel("True class") plt**.**xlabel("Predicted class") plt**.**show()

*#Print Accuracy,Precision and Recall*

print("Accuracy :",accuracy\_score(error\_df['True\_class'],error\_df['pred'])) print("Recall :",recall\_score(error\_df['True\_class'],error\_df['pred'])) print("Precision :",precision\_score(error\_df['True\_class'],error\_df['pred']))



Accuracy : 0.9981917769741231

Recall : 0.0

Precision : 0.0

C:\Users\Manish\.conda\envs\tensorflow\lib\site-packages\sklearn\metrics\\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no pr edicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))