```
# Importing Libraries

import sklearn
import pandas as pd
import numpy as np
from tensorflow import keras
import tensorflow as tf
import matplotlib.pyplot as plt
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the `%tens`

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in

```
Saving creditcard.csv to creditcard.csv
```

```
import io
#Basic Dataset characterestics - initial EDA
cc_df = pd.read_csv(io.BytesIO(uploaded["creditcard.csv"]))
cc_df.shape
```

```
(284807, 31)
```

```
cc_df.head()
```

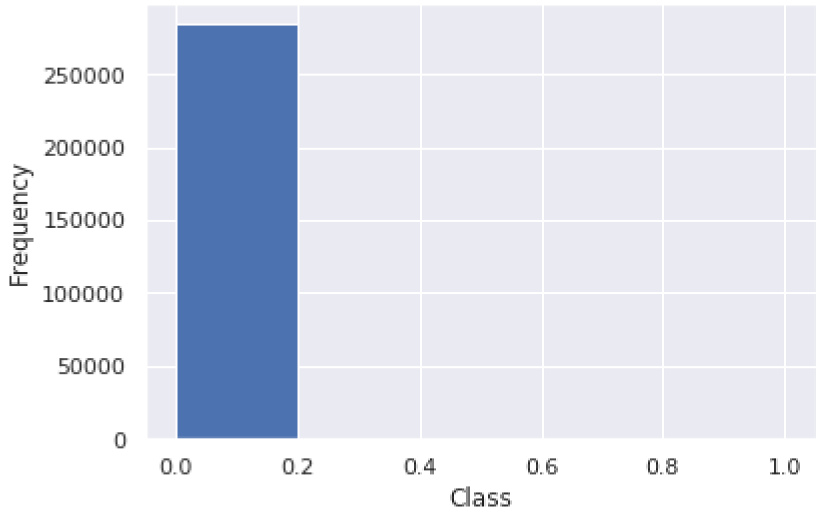
	Time	V1	V2	V3	V4	V5	V6	V7	V
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.09869
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.08510
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.24767
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.37743
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.27053

```
cc_df.tail()
```

	Time	V1	V2	V3	V4	V5	V6	V
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.91821
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.02433
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.29682
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.68618
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.57700

```
X = cc_df.iloc[:,0:30]
y = cc_df[['Class']]
```

```
#Frequency distribution of fraudulent and non-fraudulent txns
import seaborn as sns
sns.set()
plt.hist(cc_df['Class'],bins=5)
plt.xlabel('Class')
plt.ylabel('Frequency')
plt.show()
```



```
print(X.head(),"\n",X.tail())
```

```
   Time        V1        V2        V3  ...       V26       V27       V28  Amount
0   0.0 -1.359807 -0.072781  2.536347  ... -0.189115  0.133558 -0.021053  149.62
1   0.0  1.191857  0.266151  0.166480  ...  0.125895 -0.008983  0.014724    2.69
2   1.0 -1.358354 -1.340163  1.773209  ... -0.139097 -0.055353 -0.059752  378.66
3   1.0 -0.966272 -0.185226  1.792993  ... -0.221929  0.062723  0.061458  123.50
4   2.0 -1.158233  0.877737  1.548718  ...  0.502292  0.219422  0.215153   69.99

[5 rows x 30 columns]
            Time         V1         V2  ...       V27       V28  Amount
284802  172786.0 -11.881118  10.071785  ...  0.943651  0.823731    0.77
284803  172787.0  -0.732789  -0.055080  ...  0.068472 -0.053527   24.79
284804  172788.0   1.919565  -0.301254  ...  0.004455 -0.026561   67.88
284805  172788.0  -0.240440   0.530483  ...  0.108821  0.104533   10.00
284806  172792.0  -0.533413  -0.189733  ... -0.002415  0.013649  217.00

[5 rows x 30 columns]
```

```
print(y.head(),"\n",y.tail())
```

```
   Class
0      0
1      0
2      0
3      0
4      0
        Class
284802      0
284803      0
284804      0
284805      0
284806      0
```

```
#Computing number of fraudulent and authentic transactions in the dataset
fraud_Txn = cc_df.loc[cc_df['Class'] == 1]
nonfraud_Txn = cc_df.loc[cc_df['Class'] == 0]
print("No of fraudulent transactions: " + str(len(fraud_Txn)))
print("No of non-fraudulent transactions: " + str(len(nonfraud_Txn)))
```

```
No of fraudulent transactions: 492
No of non-fraudulent transactions: 284315
```

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

```
#Train test split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.33, random_state = 42,
```

```
#Implementing NN - with optimizer adam and loss fn = binary_crossentropy. These 2 combinatior
model = keras.Sequential()
```

```
model.add(keras.layers.Dense(30, input_dim=30, activation='relu'))     # kernel_initializer='
model.add(keras.layers.Dense(1, activation='sigmoid'))                 # kernel_initializer='
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/op
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/op
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 30)                930       
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 31        
=================================================================
Total params: 961
Trainable params: 961
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(X_train,y_train,epochs = 10)
```

```
Train on 190820 samples
Epoch 1/10
190820/190820 [==============================] - 9s 46us/sample - loss: 6.9182 - acc: 0.
Epoch 2/10
190820/190820 [==============================] - 9s 49us/sample - loss: 5.4182 - acc: 0.
Epoch 3/10
190820/190820 [==============================] - 8s 43us/sample - loss: 4.2953 - acc: 0.
Epoch 4/10
190820/190820 [==============================] - 8s 42us/sample - loss: 4.3614 - acc: 0.
Epoch 5/10
190820/190820 [==============================] - 8s 42us/sample - loss: 3.6643 - acc: 0.
Epoch 6/10
190820/190820 [==============================] - 8s 42us/sample - loss: 2.7140 - acc: 0.
Epoch 7/10
190820/190820 [==============================] - 8s 44us/sample - loss: 2.0495 - acc: 0.
Epoch 8/10
190820/190820 [==============================] - 8s 43us/sample - loss: 1.4380 - acc: 0.
Epoch 9/10
190820/190820 [==============================] - 8s 43us/sample - loss: 1.2719 - acc: 0.
Epoch 10/10
190820/190820 [==============================] - 8s 42us/sample - loss: 1.0775 - acc: 0.
<tensorflow.python.keras.callbacks.History at 0x7f83a68decf8>
```

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

```
#Predict test set
y_predicted = model.predict(X_test.as_matrix()).T[0].astype(int)
#Printing the confusion matrix
cm = confusion_matrix(y_test, y_predicted)
print("Confusion matrix:\n%s" % cm)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: FutureWarning: Method .a
  """Entry point for launching an IPython kernel.
Confusion matrix:
[[93816     9]
 [  141    21]]
```

```
a = 93816+21
b = 93816+9+141+21
accuracy_score = a/b
precision = 93816/(93816+141)
recall = 93816/(93816+9)
#Computing the f1 score as the harmonic mean between precision and recall
f1_score = 2*((precision*recall)/(precision+recall))
#Print Accuracy and F1 metrics
print("Accuracy: " + str(accuracy_score))
print("F1 Score: " + str(f1_score))
print(classification_report(y_test,y_predicted))
```

```
Accuracy: 0.9984040346005298
F1 Score: 0.9992012013931048
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     93825
           1       0.70      0.13      0.22       162

    accuracy                           1.00     93987
   macro avg       0.85      0.56      0.61     93987
weighted avg       1.00      1.00      1.00     93987
```