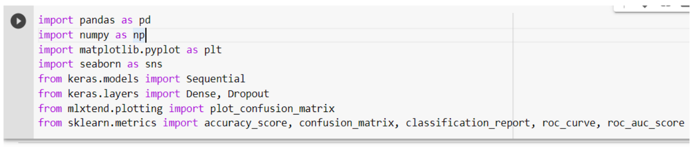
Data Science Intern Assignment

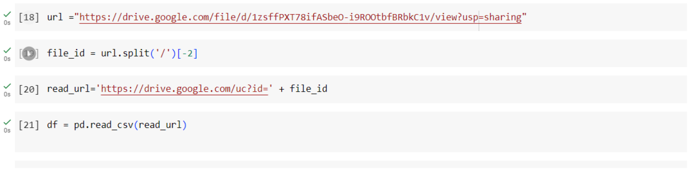
Kommanaboyina Venkata Nikhil
kommanaboyinavenkatanikhil@gmail.com

I. CODE



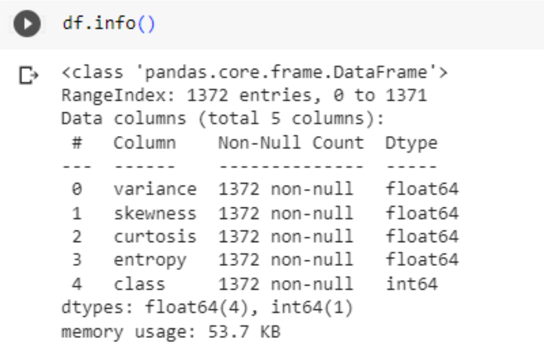
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from keras.models import Sequential
from keras.layers import Dense, Dropout
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, roc_auc_score
```

Here we are importing the libraries that are required for designing the model. The libraries pandas, and NumPy this two are used for data manipulation. libraries seaborn and matplotlib are used for the plotting of the graphs and plots. libraries Keras and sklearn are used to deploy the model.

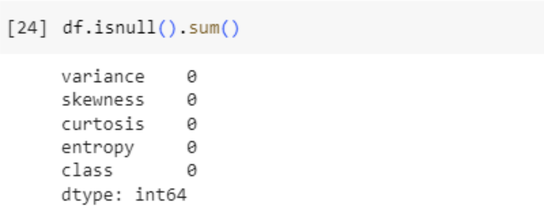


```
[18] url ="https://drive.google.com/file/d/1zsffPXT78ifASbeO-i9ROOtbfBRbkC1v/view?usp=sharing"

[19] file_id = url.split('/')[-2]

[20] read_url='https://drive.google.com/uc?id=' + file_id

[21] df = pd.read_csv(read_url)
```

Here I am reading the data from the drive directly. I used the Drive link because when we are using the google colab we need to upload every time when the google colab link is expired. so to avoid this issue I used the google drive link.

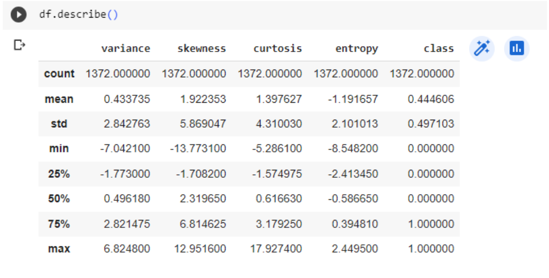


```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1372 entries, 0 to 1371
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   variance  1372 non-null   float64
 1   skewness  1372 non-null   float64
 2   curtosis  1372 non-null   float64
 3   entropy   1372 non-null   float64
 4   class     1372 non-null   int64  
dtypes: float64(4), int64(1)
memory usage: 53.7 KB
```



```
[24] df.isnull().sum()

variance    0
skewness    0
curtosis    0
entropy     0
class       0
dtype: int64
```

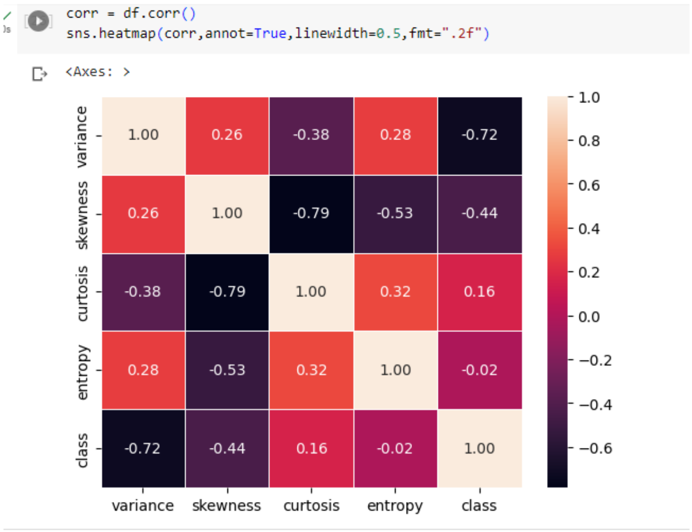
Here we can see that the total number of rows is 1372 and 4 columns and we can also see that the dataset is complete without any missing values.



```
df.describe()
```

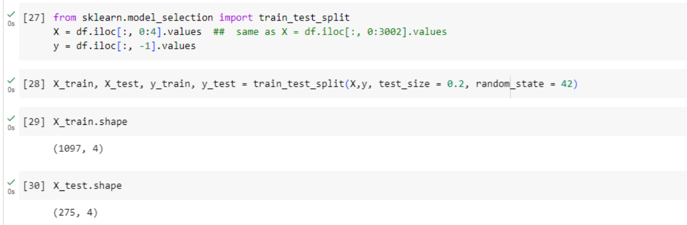
	variance	skewness	curtosis	entropy	class
count	1372.000000	1372.000000	1372.000000	1372.000000	1372.000000
mean	0.433735	1.922353	1.397627	-1.191657	0.444606
std	2.842763	5.869047	4.310030	2.101013	0.497103
min	-7.042100	-13.773100	-5.286100	-8.548200	0.000000
25%	-1.773000	-1.708200	-1.574975	-2.413450	0.000000
50%	0.496180	2.319650	0.616630	-0.586650	0.000000
75%	2.821475	6.814625	3.179250	0.394810	1.000000
max	6.824800	12.951600	17.927400	2.449500	1.000000

Here we can see the statistical analysis of the dataset.



```
corr = df.corr()
sns.heatmap(corr,annot=True,linewidth=0.5,fmt=".2f")

<Axes: >
```

Here we can see the correlation between each variable of the dataset.

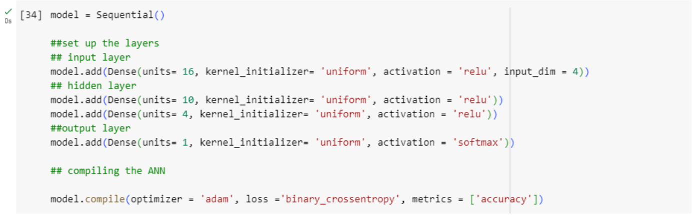


```
[27] from sklearn.model_selection import train_test_split
     X = df.iloc[:, 0:4].values  ##  same as X = df.iloc[:, 0:3002].values
     y = df.iloc[:, -1].values

[28] X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state = 42)

[29] X_train.shape

     (1097, 4)

[30] X_test.shape

     (275, 4)
```

Here I am splitting the dataset into two parts. the parts are training and testing data.80 percent of the data is the training

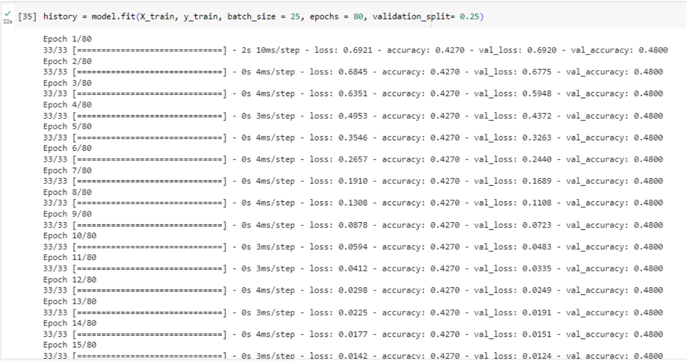
Identify applicable funding agency here. If none, delete this.

part and the remaining 20 percent is the testing part. We also can see the shape of the both training and testing parts also. after splitting the data set into testing and training parts designing the model is next step

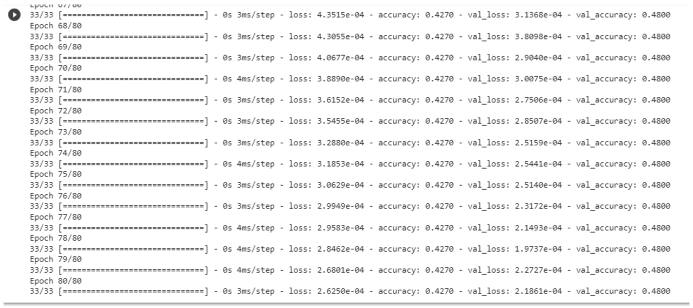


```
[34] model = Sequential()

     ##set up the layers
     ## input layer
     model.add(Dense(units= 16, kernel_initializer= 'uniform', activation = 'relu', input_dim = 4))
     ## hidden layer
     model.add(Dense(units= 10, kernel_initializer= 'uniform', activation = 'relu'))
     model.add(Dense(units= 4, kernel_initializer= 'uniform', activation = 'relu'))
     ##output layer
     model.add(Dense(units= 1, kernel_initializer= 'uniform', activation = 'softmax'))

     ## compiling the ANN

     model.compile(optimizer = 'adam', loss ='binary_crossentropy', metrics = ['accuracy'])
```

here we designed the ANN model. while designing the model three parts are required. first input layer, the second part is the hidden layer and the final part is the output layer. as a kernel initializer, I am using Unifrom and as an activation function, we are using the relu as an activation function, and the output layer I am using softmax as the activation function. While completing the model we are using the Adam as the optimizer and for the losses we are using the binary crosssentropy function.

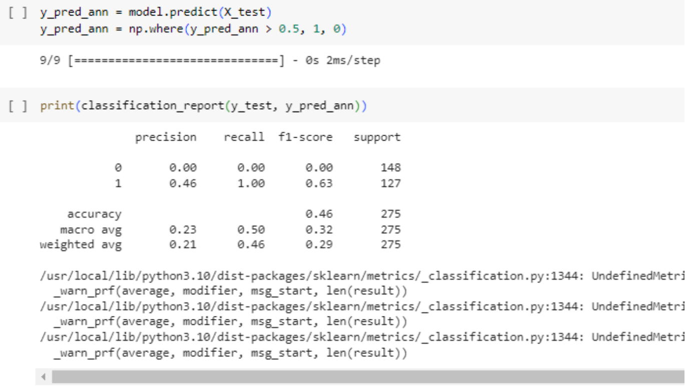


```
[35] history = model.fit(X_train, y_train, batch_size = 25, epochs = 80, validation_split= 0.25)

     Epoch 1/80
     33/33 [==============================] - 2s 10ms/step - loss: 0.6921 - accuracy: 0.4270 - val_loss: 0.6920 - val_accuracy: 0.4800
     Epoch 2/80
     33/33 [==============================] - 0s 4ms/step - loss: 0.6845 - accuracy: 0.4270 - val_loss: 0.6775 - val_accuracy: 0.4800
     Epoch 3/80
     33/33 [==============================] - 0s 4ms/step - loss: 0.6351 - accuracy: 0.4270 - val_loss: 0.5948 - val_accuracy: 0.4800
     Epoch 4/80
     33/33 [==============================] - 0s 3ms/step - loss: 0.4953 - accuracy: 0.4270 - val_loss: 0.4372 - val_accuracy: 0.4800
     Epoch 5/80
     33/33 [==============================] - 0s 4ms/step - loss: 0.3546 - accuracy: 0.4270 - val_loss: 0.3263 - val_accuracy: 0.4800
     Epoch 6/80
     33/33 [==============================] - 0s 4ms/step - loss: 0.2657 - accuracy: 0.4270 - val_loss: 0.2440 - val_accuracy: 0.4800
     Epoch 7/80
     33/33 [==============================] - 0s 4ms/step - loss: 0.1910 - accuracy: 0.4270 - val_loss: 0.1689 - val_accuracy: 0.4800
     Epoch 8/80
     33/33 [==============================] - 0s 4ms/step - loss: 0.1308 - accuracy: 0.4270 - val_loss: 0.1108 - val_accuracy: 0.4800
     Epoch 9/80
     33/33 [==============================] - 0s 4ms/step - loss: 0.0878 - accuracy: 0.4270 - val_loss: 0.0723 - val_accuracy: 0.4800
     Epoch 10/80
     33/33 [==============================] - 0s 3ms/step - loss: 0.0594 - accuracy: 0.4270 - val_loss: 0.0483 - val_accuracy: 0.4800
     Epoch 11/80
     33/33 [==============================] - 0s 3ms/step - loss: 0.0412 - accuracy: 0.4270 - val_loss: 0.0335 - val_accuracy: 0.4800
     Epoch 12/80
     33/33 [==============================] - 0s 4ms/step - loss: 0.0298 - accuracy: 0.4270 - val_loss: 0.0249 - val_accuracy: 0.4800
     Epoch 13/80
     33/33 [==============================] - 0s 3ms/step - loss: 0.0225 - accuracy: 0.4270 - val_loss: 0.0191 - val_accuracy: 0.4800
     Epoch 14/80
     33/33 [==============================] - 0s 4ms/step - loss: 0.0177 - accuracy: 0.4270 - val_loss: 0.0151 - val_accuracy: 0.4800
     Epoch 15/80
     33/33 [==============================] - 0s 3ms/step - loss: 0.0142 - accuracy: 0.4270 - val_loss: 0.0124 - val_accuracy: 0.4800
```

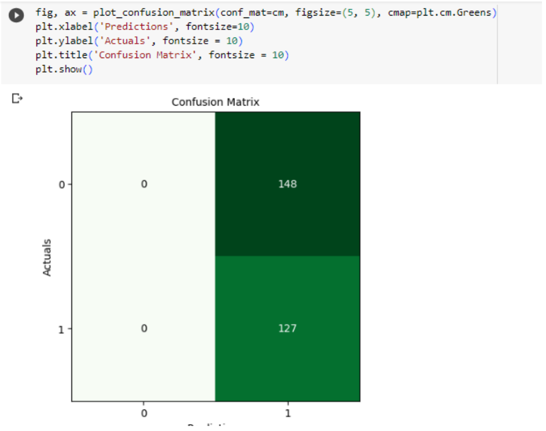


```
Epoch 67/80
33/33 [==============================] - 0s 3ms/step - loss: 4.3515e-04 - accuracy: 0.4270 - val_loss: 3.1368e-04 - val_accuracy: 0.4800
Epoch 68/80
33/33 [==============================] - 0s 3ms/step - loss: 4.3055e-04 - accuracy: 0.4270 - val_loss: 3.8098e-04 - val_accuracy: 0.4800
Epoch 69/80
33/33 [==============================] - 0s 3ms/step - loss: 4.0677e-04 - accuracy: 0.4270 - val_loss: 2.9040e-04 - val_accuracy: 0.4800
Epoch 70/80
33/33 [==============================] - 0s 4ms/step - loss: 3.8890e-04 - accuracy: 0.4270 - val_loss: 3.0075e-04 - val_accuracy: 0.4800
Epoch 71/80
33/33 [==============================] - 0s 3ms/step - loss: 3.6152e-04 - accuracy: 0.4270 - val_loss: 2.7506e-04 - val_accuracy: 0.4800
Epoch 72/80
33/33 [==============================] - 0s 3ms/step - loss: 3.5455e-04 - accuracy: 0.4270 - val_loss: 2.8507e-04 - val_accuracy: 0.4800
Epoch 73/80
33/33 [==============================] - 0s 3ms/step - loss: 3.2880e-04 - accuracy: 0.4270 - val_loss: 2.5159e-04 - val_accuracy: 0.4800
Epoch 74/80
33/33 [==============================] - 0s 4ms/step - loss: 3.1853e-04 - accuracy: 0.4270 - val_loss: 2.5441e-04 - val_accuracy: 0.4800
Epoch 75/80
33/33 [==============================] - 0s 3ms/step - loss: 3.0629e-04 - accuracy: 0.4270 - val_loss: 2.5140e-04 - val_accuracy: 0.4800
Epoch 76/80
33/33 [==============================] - 0s 3ms/step - loss: 2.9949e-04 - accuracy: 0.4270 - val_loss: 2.3172e-04 - val_accuracy: 0.4800
Epoch 77/80
33/33 [==============================] - 0s 4ms/step - loss: 2.9583e-04 - accuracy: 0.4270 - val_loss: 2.1493e-04 - val_accuracy: 0.4800
Epoch 78/80
33/33 [==============================] - 0s 4ms/step - loss: 2.8462e-04 - accuracy: 0.4270 - val_loss: 1.9737e-04 - val_accuracy: 0.4800
Epoch 79/80
33/33 [==============================] - 0s 4ms/step - loss: 2.6801e-04 - accuracy: 0.4270 - val_loss: 2.2727e-04 - val_accuracy: 0.4800
Epoch 80/80
33/33 [==============================] - 0s 3ms/step - loss: 2.6250e-04 - accuracy: 0.4270 - val_loss: 2.1861e-04 - val_accuracy: 0.4800
```

Precision can be seen as a measure of quality, and recall as a measure of quantity. Higher precision means that an algorithm returns more relevant results than irrelevant ones, and high recall means that an algorithm returns most of the relevant results (whether or not irrelevant ones are also returned). coming to F1 score is a machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model. The accuracy metric computes how many times a model made a correct prediction across the entire dataset.

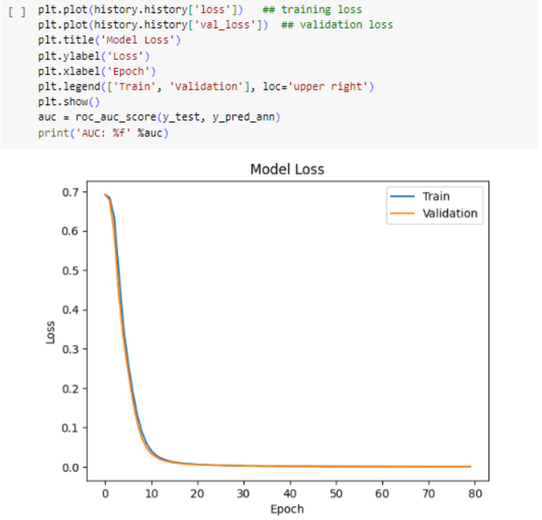


```
[ ] y_pred_ann = model.predict(X_test)
    y_pred_ann = np.where(y_pred_ann > 0.5, 1, 0)

    9/9 [==============================] - 0s 2ms/step

[ ] print(classification_report(y_test, y_pred_ann))

                  precision    recall  f1-score   support

               0       0.00      0.00      0.00       148
               1       0.46      1.00      0.63       127

        accuracy                           0.46       275
       macro avg       0.23      0.50      0.32       275
    weighted avg       0.21      0.46      0.29       275

    /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetri
      _warn_prf(average, modifier, msg_start, len(result))
    /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetri
      _warn_prf(average, modifier, msg_start, len(result))
    /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetri
      _warn_prf(average, modifier, msg_start, len(result))
```



```
fig, ax = plot_confusion_matrix(conf_mat=cm, figsize=(5, 5), cmap=plt.cm.Greens)
plt.xlabel('Predictions', fontsize=10)
plt.ylabel('Actuals', fontsize = 10)
plt.title('Confusion Matrix', fontsize = 10)
plt.show()
```

confusion matrix for the model. Here from the plot for the



```
[ ] plt.plot(history.history['loss'])   ## training loss
    plt.plot(history.history['val_loss'])  ## validation loss
    plt.title('Model Loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'], loc='upper right')
    plt.show()
    auc = roc_auc_score(y_test, y_pred_ann)
    print('AUC: %f' %auc)
```

number of epochs and loss of the model. The model is exactly fit. it is not under-fitted or over-fitted.