In [48]:

```
import pandas as pd
import numpy as np
import os
import joblib as jb
import sklearn
import pydotplus
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [9]:

```
train_data=pd.read_excel('Training_Updated.xlsx')
valid_data=test_data=pd.read_excel('Validation_final.xlsx')
test_data=pd.read_excel('Test_final.xlsx')
```

In [34]:

```
train_data.info()
!pip install tensorflow
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   AgeCategory           32561 non-null  int64
 1   Workclass             32561 non-null  int64
 2   Education             32561 non-null  int64
 3   EducationNum          32561 non-null  int64
 4   MaritalStatus         32561 non-null  int64
 5   Occupation            32561 non-null  int64
 6   Relationship          32561 non-null  int64
 7   Sex                   32561 non-null  int64
 8   NativeCountry         32561 non-null  int64
 9   Race                  32561 non-null  int64
 10  FnlwgtCategory        32561 non-null  int64
 11  CapitalGainCategory   32561 non-null  int64
 12  CapitalLossCategory   32561 non-null  int64
 13  HoursPerWeekCategory  32561 non-null  int64
 14  Class                 32561 non-null  int64
dtypes: int64(15)
memory usage: 3.7 MB
^C
```

In [46]:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

In [43]:

```
X=train_data.drop(['Class'],axis=1)
y=train_data['Class']

X_train=train_data.drop(['Class'],axis=1)
y_train=train_data['Class']

X_val=valid_data.drop(['Class'],axis=1)
y_val=valid_data['Class']

X_test=test_data.drop(['Class'],axis=1)
y_test=test_data['Class']

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [47]:

```
def build_model(num_layers):
    model = Sequential()
    model.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))

    for _ in range(num_layers - 1):
        model.add(Dense(64, activation='relu'))

    model.add(Dense(1, activation='sigmoid'))
    return model

num_layers_list = [1, 2, 3]

for num_layers in num_layers_list:
    model = build_model(num_layers)
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_val, y_val)

    _, accuracy = model.evaluate(X_val, y_val)
    print(" ")
    print(f"Neural Network with {num_layers} hidden layer(s) accuracy: {accuracy}")
    print(" ")
```

```
Epoch 1/10
1018/1018 [==============================] - 3s 2ms/step - loss: 0.4352 -
accuracy: 0.8066 - val_loss: 0.4129 - val_accuracy: 0.8132
Epoch 2/10
1018/1018 [==============================] - 2s 2ms/step - loss: 0.3909 -
accuracy: 0.8224 - val_loss: 0.3769 - val_accuracy: 0.8251
Epoch 3/10
1018/1018 [==============================] - 2s 2ms/step - loss: 0.3708 -
accuracy: 0.8272 - val_loss: 0.3596 - val_accuracy: 0.8315
Epoch 4/10
1018/1018 [==============================] - 2s 2ms/step - loss: 0.3586 -
accuracy: 0.8333 - val_loss: 0.3628 - val_accuracy: 0.8325
Epoch 5/10
1018/1018 [==============================] - 2s 2ms/step - loss: 0.3586 -
accuracy: 0.8316 - val_loss: 0.3564 - val_accuracy: 0.8323
Epoch 6/10
1018/1018 [==============================] - 2s 2ms/step - loss: 0.3591 -
accuracy: 0.8317 - val_loss: 0.3595 - val_accuracy: 0.8298
Epoch 7/10
1018/1018 [==============================] - 2s 2ms/step - loss: 0.3599 -
accuracy: 0.8304 - val_loss: 0.3638 - val_accuracy: 0.8342
Epoch 8/10
1018/1018 [==============================] - 2s 2ms/step - loss: 0.3559 -
accuracy: 0.8335 - val_loss: 0.3512 - val_accuracy: 0.8349
Epoch 9/10
1018/1018 [==============================] - 2s 2ms/step - loss: 0.3541 -
accuracy: 0.8338 - val_loss: 0.3518 - val_accuracy: 0.8331
Epoch 10/10
1018/1018 [==============================] - 2s 2ms/step - loss: 0.3536 -
accuracy: 0.8350 - val_loss: 0.3513 - val_accuracy: 0.8358
255/255 [==============================] - 0s 1ms/step - loss: 0.3513 - ac
curacy: 0.8358

Neural Network with 1 hidden layer(s) accuracy: 0.835769534111023

Epoch 1/10
1018/1018 [==============================] - 3s 2ms/step - loss: 0.4595 -
accuracy: 0.8007 - val_loss: 0.4415 - val_accuracy: 0.7917
Epoch 2/10
1018/1018 [==============================] - 2s 2ms/step - loss: 0.4038 -
accuracy: 0.8146 - val_loss: 0.3731 - val_accuracy: 0.8274
Epoch 3/10
1018/1018 [==============================] - 2s 2ms/step - loss: 0.3700 -
accuracy: 0.8255 - val_loss: 0.3622 - val_accuracy: 0.8349
Epoch 4/10
1018/1018 [==============================] - 2s 2ms/step - loss: 0.3619 -
accuracy: 0.8307 - val_loss: 0.3669 - val_accuracy: 0.8226
Epoch 5/10
1018/1018 [==============================] - 2s 2ms/step - loss: 0.3601 -
accuracy: 0.8309 - val_loss: 0.3536 - val_accuracy: 0.8336
Epoch 6/10
1018/1018 [==============================] - 2s 2ms/step - loss: 0.3562 -
accuracy: 0.8337 - val_loss: 0.3516 - val_accuracy: 0.8317
Epoch 7/10
1018/1018 [==============================] - 3s 3ms/step - loss: 0.3555 -
accuracy: 0.8321 - val_loss: 0.3647 - val_accuracy: 0.8261
Epoch 8/10
```

In [ ]: