In [8]:

```
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]:

```
from sklearn.preprocessing import LabelEncoder
```

In [15]:

```
pd.read_excel('Combined_Updated.xlsx')
valid_data=test_data=pd.read_excel('Validation_final.xlsx')
```

In [17]:

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

In [10]:

```
data=pd.read_excel('Combined_Updated.xlsx')
```

In [11]:

```
xc=['AgeCategory','Workclass','Education','EducationNum','MaritalStatus','Occupation','R
y=['Yes','No']
all_input=data[xc]
all_class=data['Class']
```

In [12]:

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

In [13]:

```
(X_train,X_test,Y_train,Y_test)=train_test_split(all_input,all_class,train_size=0.67,ran
```

In [14]:

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

In [22]:

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

    # Compile and train the model
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    model.fit(X_train, Y_train, epochs=10, batch_size=32)

    # Evaluate the model on the testing data
    loss, accuracy = model.evaluate(X_test, Y_test)
    print(" ")
    print(f"Neural Network with {num_layers} hidden layer(s) accuracy: {accuracy}")
    print(" ")
```

```
Epoch 1/10
1023/1023 [==============================] - 2s 2ms/step - loss: 0.5514 -
accuracy: 0.7902
Epoch 2/10
1023/1023 [==============================] - 1s 1ms/step - loss: 0.3959 -
accuracy: 0.8215
Epoch 3/10
1023/1023 [==============================] - 1s 1ms/step - loss: 0.3749 -
accuracy: 0.8279
Epoch 4/10
1023/1023 [==============================] - 1s 1ms/step - loss: 0.3625 -
accuracy: 0.8315
Epoch 5/10
1023/1023 [==============================] - 1s 1ms/step - loss: 0.3571 -
accuracy: 0.8333
Epoch 6/10
1023/1023 [==============================] - 2s 2ms/step - loss: 0.3546 -
accuracy: 0.8341
Epoch 7/10
1023/1023 [==============================] - 2s 2ms/step - loss: 0.3524 -
accuracy: 0.8360
Epoch 8/10
1023/1023 [==============================] - 2s 2ms/step - loss: 0.3503 -
accuracy: 0.8374
Epoch 9/10
1023/1023 [==============================] - 2s 2ms/step - loss: 0.3495 -
accuracy: 0.8366
Epoch 10/10
1023/1023 [==============================] - 1s 1ms/step - loss: 0.3485 -
accuracy: 0.8383
504/504 [==============================] - 1s 1ms/step - loss: 0.3523 - ac
curacy: 0.8352

Neural Network with 1 hidden layer(s) accuracy: 0.8351532220840454

Epoch 1/10
1023/1023 [==============================] - 2s 1ms/step - loss: 0.4360 -
accuracy: 0.8034
Epoch 2/10
1023/1023 [==============================] - 2s 1ms/step - loss: 0.3913 -
accuracy: 0.8238
Epoch 3/10
1023/1023 [==============================] - 2s 2ms/step - loss: 0.3692 -
accuracy: 0.8303
Epoch 4/10
1023/1023 [==============================] - 1s 1ms/step - loss: 0.3637 -
accuracy: 0.8335
Epoch 5/10
1023/1023 [==============================] - 2s 1ms/step - loss: 0.3614 -
accuracy: 0.8331
Epoch 6/10
1023/1023 [==============================] - 2s 2ms/step - loss: 0.3594 -
accuracy: 0.8347
Epoch 7/10
1023/1023 [==============================] - 2s 2ms/step - loss: 0.3593 -
accuracy: 0.8333
Epoch 8/10
```

```
In [ ]:
```