In [1]:

```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [2]:

```
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LeakyReLU, PReLU, ELU
from keras.layers import Dropout
```

In [3]:

```
data = pd.read_csv("churn_Modelling.csv")
```

In [ ]:

```
data.head()
```

In [4]:

```
Geography = pd.get_dummies(data['Geography'],drop_first='True')
Gender = pd.get_dummies(data['Gender'],drop_first='True')
```

In [5]:

```
data = data.drop(['RowNumber','CustomerId','Surname','Geography','Gender'],axis=1)
```

In [6]:

```
data = pd.concat([data,Geography,Gender],axis=1)
```

In [7]:

```
data.head()
```

Out[7]:

ditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
619	42	2	0.00	1	1	1	101348.88
608	41	1	83807.86	1	0	1	112542.58
502	42	8	159660.80	3	1	0	113931.57
699	39	1	0.00	2	0	0	93826.63
850	43	2	125510.82	1	1	1	79084.10

In [8]:

```
data.shape
```

Out[8]:

```
(10000, 12)
```

In [9]:

```
test = data['Exited']
test.head(2)
```

Out[9]:

```
0    1
1    0
Name: Exited, dtype: int64
```

In [10]:

```
train = data.drop(['Exited'],axis=1)
```

In [ ]:

```
train.head(2)
```

In [11]:

```
xtrain,xtest,ytrain,ytest = train_test_split(train,test,test_size=0.2,random_state=4
```

In [12]:

```
from sklearn.preprocessing import StandardScaler
```

In [13]:

```
ss = StandardScaler()

xtrain = ss.fit_transform(xtrain)
xtest = ss.transform(xtest)
```

In [14]:

```
classifier = Sequential()

classifier.add(Dense(units = 10, kernel_initializer = 'he_uniform', activation = 're
classifier.add(Dropout(0.3))
classifier.add(Dense(units = 20 , kernel_initializer = 'he_uniform', activation = 'r
classifier.add(Dropout(0.3))

classifier.add(Dense(units = 15 , kernel_initializer = 'he_uniform', activation = 'r
classifier.add(Dropout(0.3))

classifier.add(Dense(units = 1 , kernel_initializer = 'glorot_uniform' , activation

classifier.compile(optimizer = 'Adamax' , loss = 'binary_crossentropy', metrics = ['

model_history = classifier.fit(xtrain,ytrain ,
                               validation_split = 0.33 ,
                               batch_size = 10 , epochs = 100)
```

```
Epoch 95/100
536/536 [==============================] - 3s 5ms/step - loss: 0.4355
- accuracy: 0.7979 - val_loss: 0.3971 - val_accuracy: 0.8292
Epoch 96/100
536/536 [==============================] - 2s 4ms/step - loss: 0.4365
- accuracy: 0.7977 - val_loss: 0.3976 - val_accuracy: 0.8300
Epoch 97/100
536/536 [==============================] - 2s 3ms/step - loss: 0.4253
- accuracy: 0.8088 - val_loss: 0.3965 - val_accuracy: 0.8296
Epoch 98/100

536/536 [==============================] - 2s 3ms/step - loss: 0.4478
- accuracy: 0.7939 - val_loss: 0.3949 - val_accuracy: 0.8296
Epoch 99/100
536/536 [==============================] - 2s 3ms/step - loss: 0.4486
- accuracy: 0.7927 - val_loss: 0.3960 - val_accuracy: 0.8300
Epoch 100/100
536/536 [==============================] - 2s 3ms/step - loss: 0.4360
- accuracy: 0.8024 - val_loss: 0.3976 - val_accuracy: 0.8304
```

In [ ]:

```
xtrain.shape
```

In [ ]:

```
model_history.history.keys()
```

In [ ]:

```
import matplotlib.pyplot as plt
```

In [ ]:

```
plt.plot(model_history.history['acc'])
plt.plot(model_history.history['val_acc'])
plt.title('Model accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['train','test'], loc = 'upper left')
plt.show()


# summarize history for loss
plt.plot(model_history.history['loss'])
plt.plot(model_history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

In [ ]:

```
ypred = classifier.predict(xtest)
```

In [ ]:

```
ypred = (ypred >0.5)
```

In [ ]:

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

In [ ]:

```
cm = confusion_matrix(ytest,ypred)
```

In [ ]:

```
cm
```

In [ ]:

```
cr = classification_report(ytest,ypred)
```

In [ ]:

```
print(cr)
```

In [ ]:

```
from sklearn.metrics import accuracy_score
```

In [ ]:

```
score = accuracy_score(ytest,ypred)
```

In [ ]:

```
score
```

In [ ]:

```

```