Chapter 5 - Ex1: Social Network Ads

Cho dữ liệu Social_Network_Ads.csv

Sử dụng thuật toán KNN để dự đoán khách hàng mua (1) hay không mua sản phẩm (0) dựa trên các thông tin được cung cấp:

1. Đọc dữ liệu và gán cho biến data. Tiền xử lý dữ liệu (nếu cần)
2. Tạo inputs data với các cột trừ cột *Purchased*, và outputs data với 1 cột là *Purchased*
3. Từ inputs data và outputs data => Tạo X_train, X_test, y_train, y_test với tỷ lệ 70-30
4. Thực hiện KNN với X_train, y_train
5. Dự đoán y từ X_test => so sánh với y_test
6. Đánh giá mô hình => Nhận xét
7. Ghi mô hình (nếu mô hình tốt sau khi đánh giá)

In [39]:
```
# link tham khảo: https://towardsdatascience.com/knn-in-python-835643e2fb53
```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

In [2]:
```
# import some data to play with
data = pd.read_csv("Social_Network_Ads.csv")
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
User ID            400 non-null int64
Gender             400 non-null object
Age                400 non-null int64
EstimatedSalary    400 non-null int64
Purchased          400 non-null int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [3]:
```
data.shape
```

Out[3]:
```
(400, 5)
```

In [4]:
```
data.head()
```

Out[4]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [5]:
```
# data.tail()
```

In [6]:
```
# thống kê số lượng các lớp
data.groupby('Purchased').count()["Gender"]
```

Out[6]:
```
Purchased
0    257
1    143
Name: Gender, dtype: int64
```

In [7]:
```
from sklearn.preprocessing import LabelEncoder
```

In [8]:
```
le = LabelEncoder()
data['Gender_E'] = le.fit_transform(data.Gender)
```

In [9]:
```
data.head()
```

Out[9]:

	User ID	Gender	Age	EstimatedSalary	Purchased	Gender_E
0	15624510	Male	19	19000	0	1
1	15810944	Male	35	20000	0	1
2	15668575	Female	26	43000	0	0
3	15603246	Female	27	57000	0	0
4	15804002	Male	19	76000	0	1

In [10]:
```
# The columns that we will be making predictions with.
inputs = data.iloc[:,[2,3,5]]
inputs.shape
```

Out[10]:
```
(400, 3)
```

In [11]:
```
inputs.head()
```

Out[11]:

	Age	EstimatedSalary	Gender_E
0	19	19000	1
1	35	20000	1
2	26	43000	0
3	27	57000	0
4	19	76000	1

In [12]:
```
# The column that we want to predict.
outputs = data['Purchased']
#outputs = np.array(outputs)
outputs.shape
```

Out[12]:
```
(400,)
```

In [13]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, outputs,
                                                    test_size=0.30,
                                                    random_state=1)
```

In [14]:
```
# Cần phải Scale dữ liệu vì Age và EstimatedSalary có thang đo khác nhau nhiều
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

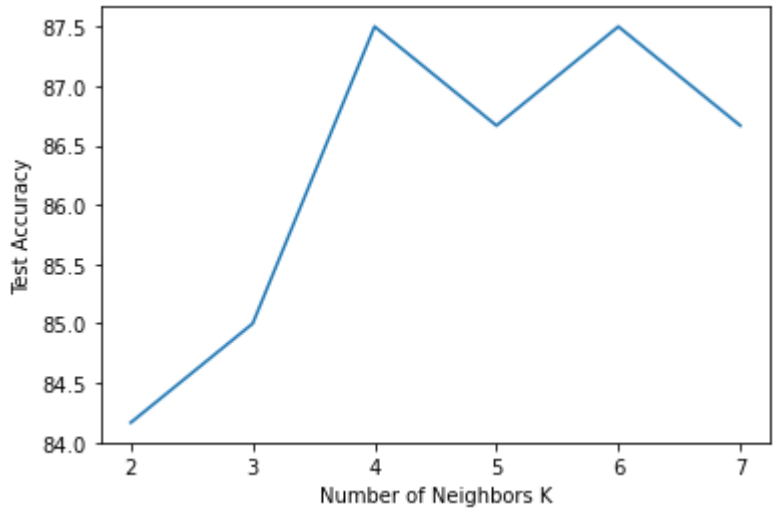
In [15]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
list_k = []
list_acc = []
for K_value in range(2,int((y_train.shape[0]**0.5)/2)):
#for K_value in range(2,10):
    list_k.append(K_value)
    neigh = KNeighborsClassifier(n_neighbors = K_value)
    neigh.fit(X_train, y_train)
    y_pred = neigh.predict(X_test)
    acc = accuracy_score(y_test,y_pred)*100
    list_acc.append(acc)
    print("k = ", K_value,": Accuracy is ", accuracy_score(y_test,y_pred))
```

```
k =  2 : Accuracy is  0.8416666666666667
k =  3 : Accuracy is  0.85
k =  4 : Accuracy is  0.875
k =  5 : Accuracy is  0.8666666666666667
k =  6 : Accuracy is  0.875
k =  7 : Accuracy is  0.8666666666666667
```

In [16]:
```
vi_tri = list_acc.index(max(list_acc))
k = list_k[vi_tri]
print("The optimal number of neighbors is", k,"with", list_acc[vi_tri])
```

```
The optimal number of neighbors is 4 with 87.5
```

In [17]:
```
plt.plot(list_k, list_acc)
plt.xlabel('Number of Neighbors K')
plt.ylabel('Test Accuracy')
plt.show()
```



In [18]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [19]:
```
for i in (4,6):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    print("k=", i, ": The Train prediction accuracy is: ",
          knn.score(X_train,y_train)*100,"%")
    print("------ The Test prediction accuracy is: ",
          knn.score(X_test,y_test)*100,"%")
    #.../.../
```

```
k= 4 : The Train prediction accuracy is:  93.57142857142857 %
------ The Test prediction accuracy is:  87.5 %
k= 6 : The Train prediction accuracy is:  93.21428571428572 %
------ The Test prediction accuracy is:  87.5 %
```

In [20]:
```
knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(X_train, y_train)
```

Out[20]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=6, p=2,
                     weights='uniform')
```

In [21]:
```
# Kiểm tra độ chính xác
print("The Train prediction accuracy is: ",
      knn.score(X_train,y_train)*100,"%")
print("The Test prediction accuracy is: ",
      knn.score(X_test,y_test)*100,"%")
```

```
The Train prediction accuracy is:  93.21428571428572 %
The Test prediction accuracy is:  87.5 %
```

In [22]:
```
y_pred = knn.predict(X_test)
# y_pred
```

In [23]:
```
# Đánh giá model
from sklearn.metrics import confusion_matrix, classification_report
```

In [24]:
```
confusion_matrix(y_test, y_pred)
```

Out[24]:
```
array([[61, 11],
       [ 4, 44]], dtype=int64)
```

In [25]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.85      0.89        72
           1       0.80      0.92      0.85        48

    accuracy                           0.88       120
   macro avg       0.87      0.88      0.87       120
weighted avg       0.88      0.88      0.88       120
```

Quan sát kết quả và đánh giá:

- Có sự chênh lệch giữa train r_score và test r_score ~7% => có thể tạm chấp nhận
- Mô hình có độ chính xác khá cao: 87.5%

Feature Selection

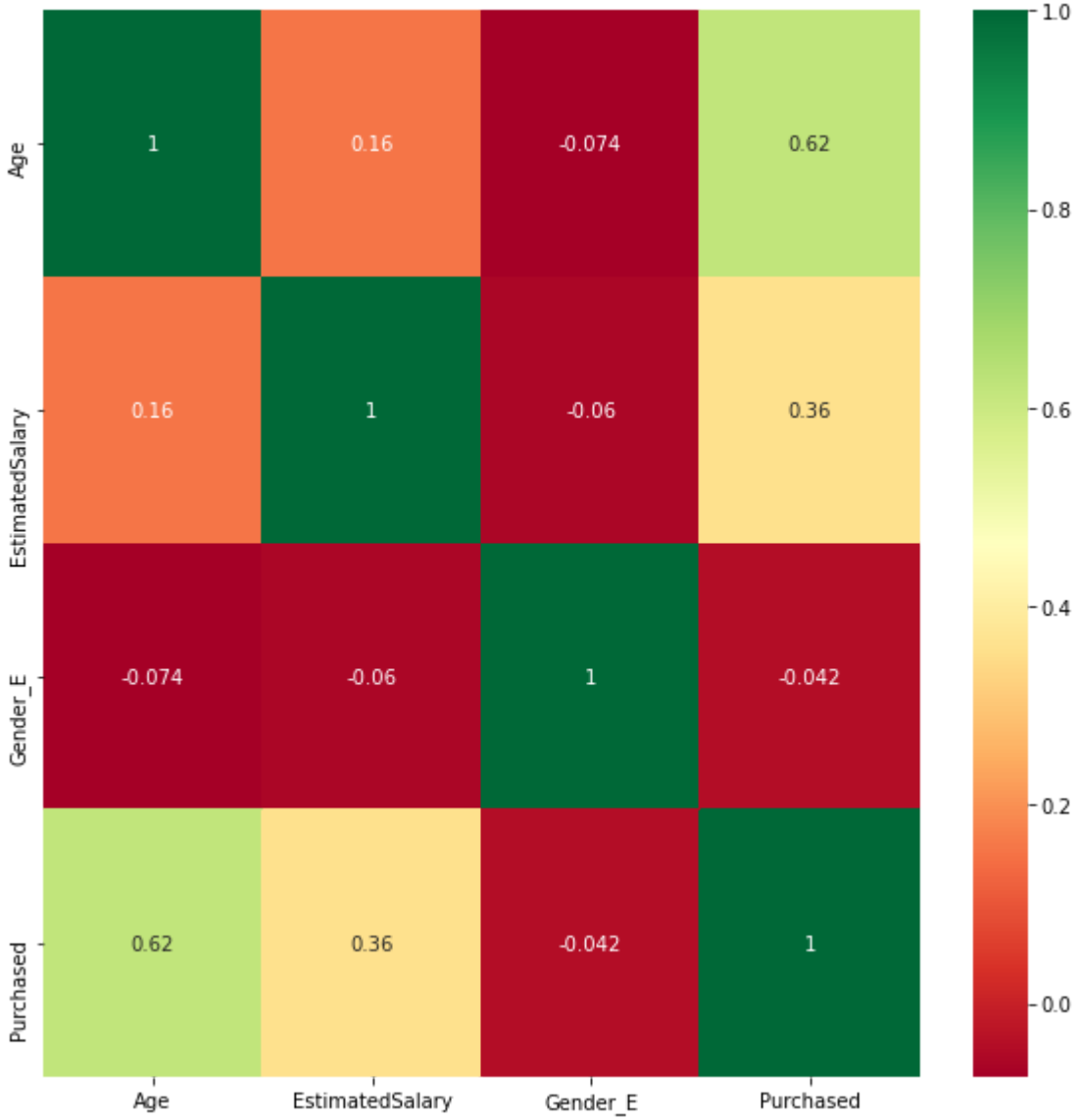
In [26]:
```
#get correlations of each features in dataset
data_sub = data.iloc[:,[2,3,5,4]]
corrmat = data_sub.corr()
top_corr_features = corrmat.index
```

In [27]:
```
data_sub.corr()
```

Out[27]:

	Age	EstimatedSalary	Gender_E	Purchased
Age	1.000000	0.155238	-0.073741	0.622454
EstimatedSalary	0.155238	1.000000	-0.060435	0.362083
Gender_E	-0.073741	-0.060435	1.000000	-0.042469
Purchased	0.622454	0.362083	-0.042469	1.000000

In [28]:
```
import seaborn as sns
plt.figure(figsize=(10,10))
#plot heat map
g=sns.heatmap(data[top_corr_features].corr(), cmap="RdYlGn", annot=True)
# annot=True: nếu muốn in cả giá trị
```



In [29]:
```
# => Select Age, EstimatedSalary
```

In [30]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

In [31]:
```
#apply SelectKBest class to extract all best features
bestfeatures = SelectKBest(score_func=chi2, k='all')
fit = bestfeatures.fit(inputs,outputs)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(inputs.columns)
```

In [32]:
```
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']  #naming the dataframe columns
print(featureScores.nlargest(4,'Score'))  #print 4 best features
```

```
             Specs          Score
1  EstimatedSalary  872013.169231
0              Age     451.155226
2         Gender_E       0.367946
```

In [33]:
```
# => select features EstimatedSalary & Age => KNN
```

In [34]:
```
inputs_new = data.iloc[:,[2,3]]
inputs_new.head()
```

Out[34]:

	Age	EstimatedSalary
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000

In [35]:
```
X_train_n, X_test_n, y_train_n, y_test_n = train_test_split(inputs_new,
                                                            outputs,
                                                    test_size=0.30,
                                                    random_state=1)
```

In [36]:
```
sc = StandardScaler()
X_train_n = sc.fit_transform(X_train_n)
X_test_n = sc.transform(X_test_n)
```

In [37]:
```
list_k = []
list_acc = []
for K_value in range(2,int((y_train.shape[0]**0.5)/2)):
#for K_value in range(2,10):
    list_k.append(K_value)
    neigh = KNeighborsClassifier(n_neighbors = K_value)
    neigh.fit(X_train_n, y_train_n)
    y_pred = neigh.predict(X_test_n)
    acc = accuracy_score(y_test_n,y_pred)*100
    list_acc.append(acc)
    print("k = ", K_value,": Accuracy is ", accuracy_score(y_test_n,y_pred))
```

```
k =  2 : Accuracy is  0.8416666666666667
k =  3 : Accuracy is  0.875
k =  4 : Accuracy is  0.8833333333333333
k =  5 : Accuracy is  0.875
k =  6 : Accuracy is  0.875
k =  7 : Accuracy is  0.8666666666666667
```

In [38]:
```
# Với k=4 có độ chính xác cao hơn khoảng 1%
```