KNN Classification

- **1.Collecting Data**
- **2.Preprocessing Data**
- **3.Train and Test the Data**
- **4.Model Creation**
- **5.Improve Model**

1.Collecting Data:

In [1]:

```
import pandas as pd
```

In [2]:

```
data=pd.read_csv('UniversalBank.csv')
data.head()
```

Out[2]:

nder	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securitie Accoun
F	25	1	49	91107	4	1.6	1	0	0	
F	45	19	34	90089	3	1.5	1	0	0	
M	39	15	11	94720	1	1.0	1	0	0	
M	35	9	100	94112	1	2.7	2	0	0	
M	35	8	45	91330	4	1.0	2	0	0	

2.Preprocessing

In [4]:

```
data.columns
```

Out[4]:

```
Index(['ID', 'Gender', 'Age', 'Experience', 'Income', 'ZIP Code', 'Family',
       'CCAvg', 'Education', 'Mortgage', 'Personal Loan', 'Securities Accoun
t',
       'CD Account', 'Online', 'CreditCard'],
      dtype='object')
```

In [5]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   ID                  5000 non-null   int64  
 1   Gender              5000 non-null   object 
 2   Age                 5000 non-null   int64  
 3   Experience          5000 non-null   int64  
 4   Income              5000 non-null   int64  
 5   ZIP Code            5000 non-null   int64  
 6   Family              5000 non-null   int64  
 7   CCAvg               5000 non-null   float64
 8   Education           5000 non-null   int64  
 9   Mortgage            5000 non-null   int64  
 10  Personal Loan       5000 non-null   int64  
 11  Securities Account  5000 non-null   int64  
 12  CD Account          5000 non-null   int64  
 13  Online              5000 non-null   int64  
 14  CreditCard          5000 non-null   int64  
dtypes: float64(1), int64(13), object(1)
memory usage: 586.1+ KB
```

In [6]:

```
from sklearn.preprocessing import LabelEncoder
lab=LabelEncoder()
```

In [7]:

```
data['Gender']=lab.fit_transform(data['Gender'])
data['Gender']
```

```
...
```

In [8]:

```
data.info()
```

```
...
```

In [9]:

```
data.drop('ID',axis=1,inplace=True)
```

In [10]:

```
data.head(5)
```

```
...
```

In [11]:

```
data.drop('ZIP Code',axis=1,inplace=True)
```

In [12]:

```
data.info()
```

...

In [13]:

```
data.shape
```

Out[13]:

```
(5000, 13)
```

In [15]:

```
data.columns
```

Out[15]:

```
Index(['Gender', 'Age', 'Experience', 'Income', 'Family', 'CCAvg', 'Educatio
n',
       'Mortgage', 'Personal Loan', 'Securities Account', 'CD Account',
       'Online', 'CreditCard'],
      dtype='object')
```

In [16]:

```
X=data[['Gender', 'Age', 'Experience', 'Income', 'Family', 'CCAvg', 'Education',
       'Mortgage', 'Personal Loan', 'Securities Account', 'CD Account',
       'Online']]
y=data['CreditCard']
```

3.Train and Test the data

In [17]:

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

In [22]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)
```

In [23]:

```
X_train.shape
```

Out[23]:

```
(3500, 12)
```

In [25]:

```
X_test.shape
```

Out[25]:

```
(1500, 12)
```

4.MOdel Creation

In [26]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [27]:

```
#create object for Model
model=KNeighborsClassifier()
```

In [28]:

```
model.fit(X_train,y_train)
```

Out[28]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
```

In [31]:

```
pred=model.predict(X_test)
pred
```

Out[31]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

To Find accuracy score

In [29]:

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

In [33]:

```
accuracy_score(y_test,pred)*100
```

Out[33]:

```
63.733333333333334
```

ToFind Confusion matrix

In [34]:

```
from sklearn.metrics import confusion_matrix
```

In [35]:

```
confusion_matrix(y_test,pred)
```

Out[35]:

```
array([[894, 148],
       [396,  62]], dtype=int64)
```

In [37]:

```
369+148
```

Out[37]:

```
517
```

In [38]:

```
894+62
```

Out[38]:

```
956
```

In [46]:

```
data.sample(3)
```

...

In [47]:

```
model.predict([[2,56,31,48,2,2.10,3,0,0,0,0,0]])
```

Out[47]:

```
array([0], dtype=int64)
```

In [49]:

```
data.corr()
```

...

In [64]:

```
input_data=data[['CD Account','Experience']]
```

In [65]:

```
output_data=data['CreditCard']
```

In [66]:

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

In [67]:

```
X_tr,X_te,y_tr,y_te=train_test_split(input_data,output_data,test_size=0.3)
```

In [68]:

```
X_tr.shape
```

Out[68]:

```
(3500, 2)
```

In [69]:

```
X_te.shape
```

Out[69]:

```
(1500, 2)
```

In [70]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [102]:

```
model=KNeighborsClassifier(n_neighbors=5)
```

In [103]:

```
model.fit(X_tr,y_tr)
```

Out[103]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
```

In [104]:

```
pred=model.predict(X_te)
```

In [105]:

```
pred
```

Out[105]:

```
array([0, 0, 0, ..., 0, 0, 1], dtype=int64)
```

In [106]:

```
accuracy_score(y_te,pred)*100
```

Out[106]:

```
67.73333333333333
```

In [107]:

```
confusion_matrix(y_te,pred)
```

Out[107]:

```
array([[898, 171],
       [313, 118]], dtype=int64)
```

In [78]:

Out[78]:

```
1016
```

In [79]:

Out[79]:

```
484
```

To Find Best K Value

In [83]:

```
k_values=[5,10,15,47,57,89,110]
score={}
```

In [92]:

```
for k in k_values:
    model=KNeighborsClassifier(n_neighbors=k)
    model.fit(X_tr,y_tr)
    score[k]=model.score(X_tr,y_tr)
```

In [86]:

```
score
```

Out[86]:

```
{5: 0.6754285714285714,
 10: 0.732,
 15: 0.7174285714285714,
 47: 0.7048571428571428,
 57: 0.7031428571428572,
 89: 0.7031428571428572,
 110: 0.7031428571428572}
```

In [93]:

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

In [94]:

```
plt.scatter(score.keys(),score.values(),c='g')
plt.grid()
plt.show()
```

...

KNN Regression

- 1.Collect The Data
- 2.PreProcess the data
- 3.Split the data for Traing and testing Purpose
- 4.Create the Model
- 5.Improve the Model

In [108]:

```
import pandas as pd
```

In [109]:

```
df=pd.read_csv('placement.csv')
df.head()
```

Out[109]:

	Year	ECE	CSE	EEE	TotalPlacedData
0	1980	10.0	10.0	20	40.0
1	1981	50.0	50.0	25	125.0
2	1982	20.0	30.0	40	90.0
3	1983	152.0	50.0	45	247.0
4	1984	25.0	40.0	55	120.0

In [110]:

```
df.shape
```

Out[110]:

```
(41, 5)
```

2.Preprocess the Data

In [111]:

```
df.isna().sum()
```

...

In [112]:

```
df.info()
```

...

Split the data for testing and traing

In [114]:

```
df.columns
```

Out[114]:

```
Index(['Year', 'ECE', 'CSE', 'EEE', 'TotalPlacedData'], dtype='object')
```

In [141]:

```
X=df[['Year', 'ECE', 'CSE', 'EEE']]
y=df['TotalPlacedData']
```

In [142]:

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

In [143]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)
```

In [144]:

```
X_train.shape
```

Out[144]:

```
(28, 4)
```

In [145]:

```
X_test.shape
```

Out[145]:

```
(13, 4)
```

In [146]:

```
from sklearn.neighbors import KNeighborsRegressor
```

In [147]:

```
# Create object for model
model=KNeighborsRegressor()
```

In [148]:

```
model.fit(X_train,y_train)
```

Out[148]:

```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
          metric_params=None, n_jobs=None, n_neighbors=5, p=2,
          weights='uniform')
```

In [149]:

```
model.score(X_train,y_train)
```

Out[149]:

```
0.895177667653771
```

In [150]:

```
model.score(X_test,y_test)
```

Out[150]:

```
0.8905138606827467
```

In [151]:

```
df.sample()
```

Out[151]:

	Year	ECE	CSE	EEE	TotalPlacedData
25	2005	65.0	124.0	244	433.0

In [152]:

```
model.predict([[2005,65.0,124.0,244]])
```

Out[152]:

```
array([459.44])
```

In [127]:

```
df.corr()
```

...

In [154]:

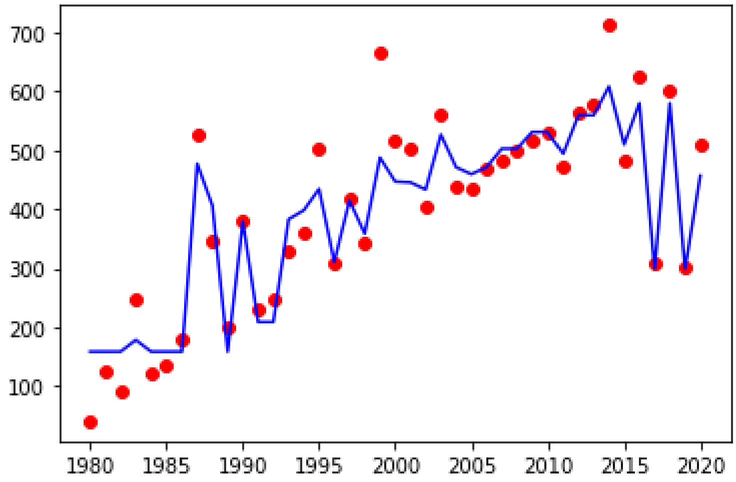
```
import matplotlib.pyplot as plt
```

In [160]:

```
plt.scatter(df['Year'],df['TotalPlacedData'],c='r')
plt.plot(df['Year'],model.predict(df[['Year', 'ECE', 'CSE', 'EEE']]),c='b')
```

Out[160]:

```
[<matplotlib.lines.Line2D at 0x22e0015cdd8>]
```



In [ ]:

```

```