EDS ASSIGNMENT NO. 6

Monali Babde (205)

Swarup Divekar(217)

Pranav Falke(219)


```
import pandas as pd

df = pd.read_csv('/content/salary.csv')

print(df)

df1 = df.groupby('Age').max()

print(df1)
```

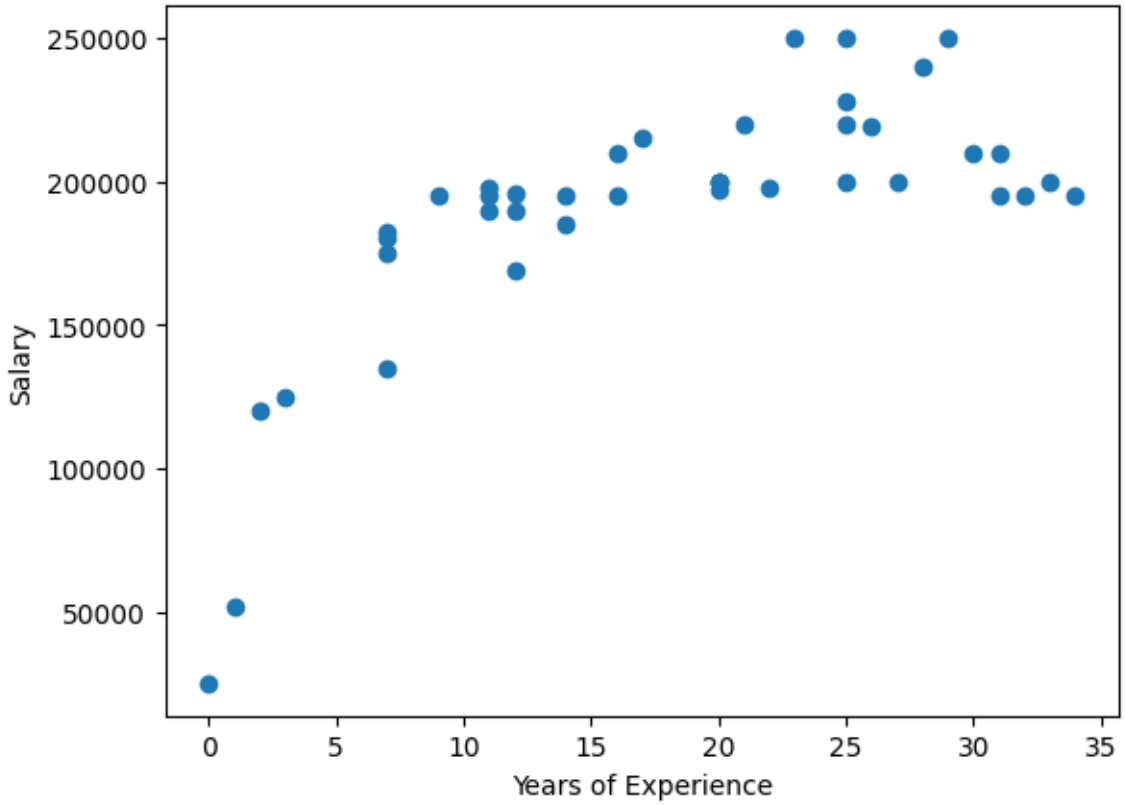
```
     Age  Gender    Education Level           Job Title  \
0    32.0    Male       Bachelor's     Software Engineer
1    28.0  Female         Master's         Data Analyst
2    45.0    Male              PhD        Senior Manager
3    36.0  Female       Bachelor's       Sales Associate
4    52.0    Male         Master's              Director
...   ...    ...              ...                   ...
6699 49.0  Female              PhD  Director of Marketing
6700 32.0    Male       High School       Sales Associate
6701 30.0  Female  Bachelor's Degree     Financial Manager
6702 46.0    Male    Master's Degree     Marketing Manager
6703 26.0  Female       High School       Sales Executive

     Years of Experience    Salary
0                    5.0   90000.0
1                    3.0   65000.0
2                   15.0  150000.0
3                    7.0   60000.0
4                   20.0  200000.0
...                  ...       ...
6699                20.0  200000.0
6700                 3.0   50000.0
6701                 4.0   55000.0
6702                14.0  140000.0
6703                 1.0   35000.0

[6704 rows x 6 columns]
     Gender                Job Title  Years of Experience    Salary
Age
21.0  Female  Junior Sales Representative                  0.0   25000.0
22.0    Male          Software Engineer                  1.0   51832.0
23.0   Other    Software Engineer Manager                  2.0  119836.0
24.0    Male              Web Developer                  3.0  125000.0
25.0   Other              Web Developer                 12.0  169159.0
26.0    Male              Web Developer                  7.0  135000.0
27.0    Male              Web Developer                  7.0  180000.0
```

```
28.0    Male            Web Developer             7.0  175000.0
29.0    Male            Web Developer             7.0  182000.0
30.0    Male            Web Developer            11.0  190000.0
31.0   Other    Software Engineer Manager            9.0  195000.0
32.0    Male            Web Developer            11.0  195000.0
33.0    Male            Web Developer            11.0  198000.0
34.0    Male            Web Developer            12.0  196000.0
35.0    Male            Web Developer            12.0  190000.0
36.0    Male            Web Developer            14.0  185000.0
37.0   Other    Software Project Manager           14.0  195000.0
38.0    Male   Software Engineer Manager           16.0  195000.0
39.0    Male        Training Specialist            16.0  210000.0
40.0    Male   Software Engineer Manager           17.0  215000.0
41.0    Male       Strategy Consultant             20.0  200000.0
42.0    Male            Web Developer            20.0  197000.0
43.0    Male       Supply Chain Manager            22.0  198000.0
44.0    Male   Software Engineer Manager           21.0  220000.0
45.0    Male   Software Engineer Manager           23.0  250000.0
46.0    Male   Software Engineer Manager           25.0  220000.0
47.0    Male         VP of Operations              25.0  200000.0
48.0    Male   Software Engineer Manager           26.0  219000.0
49.0    Male   Software Engineer Manager           25.0  228000.0
50.0    Male       Supply Chain Analyst            25.0  250000.0
51.0    Male   Software Engineer Manager           28.0  240000.0
52.0    Male   Software Engineer Manager           29.0  250000.0
53.0   Other    Software Engineer Manager           31.0  195000.0
54.0   Other    Software Engineer Manager           32.0  195270.0
55.0    Male   Software Engineer Manager           30.0  210000.0
56.0    Male   Software Engineer Manager           31.0  210000.0
57.0    Male   Software Engineer Manager           33.0  200000.0
58.0    Male   Software Engineer Manager           27.0  200000.0
60.0    Male   Software Engineer Manager           34.0  195000.0
61.0    Male   Software Engineer Manager           20.0  200000.0
62.0    Male   Software Engineer Manager           20.0  200000.0
<ipython-input-2-7c881c304abd>:5: FutureWarning: Dropping invalid columns in
DataFrameGroupBy.max is deprecated. In a future version, a TypeError will be raised. Before calling .max,
select only columns which should be valid for the function.
  df1 = df.groupby('Age').max()
```

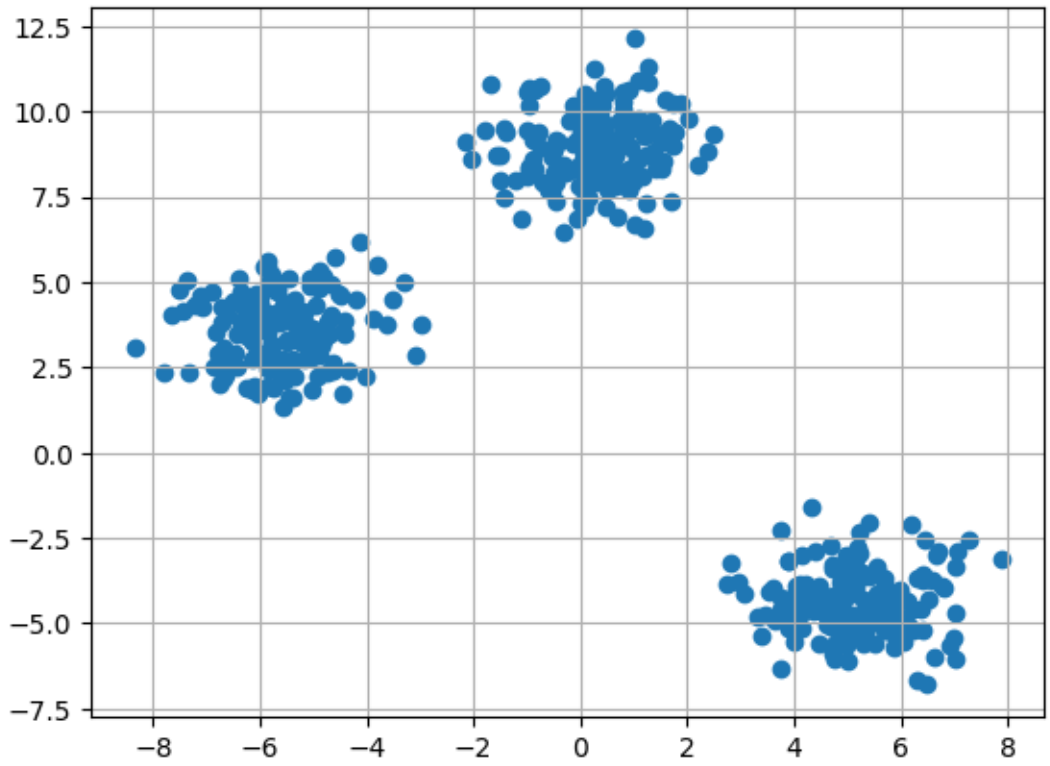
```
# Linear Regression

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression


X = df['Years of Experience']

df = df.dropna()
```

```
Y = df['Salary']


import numpy as np
X = np.array(df['Years of Experience']).reshape(-1,1)
Y = np.array(df['Salary']).reshape(-1,1)


Y = df['Salary']
# Dropping any rows with Nan Values
X_train , X_test , y_train , y_test = train_test_split(X, Y, test_size = 0.25)


#Splitting the data into training and testing data
regr = LinearRegression()
regr.fit(X_train , y_train)
print(regr.score(X_test , y_test))
# Print(df1)
plt.scatter(df1['Years of Experience'],df1['Salary'])
plt.xlabel ('Years of Experience')
plt.ylabel('Salary')
```

```
0.6639733107869471
```




```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

X,y = make_blobs(n_samples = 500,n_features = 2,centers = 3,random_state = 23)

fig = plt.figure(0)
plt.grid(True)
plt.scatter(X[:,0],X[:,1])
plt.show()
```

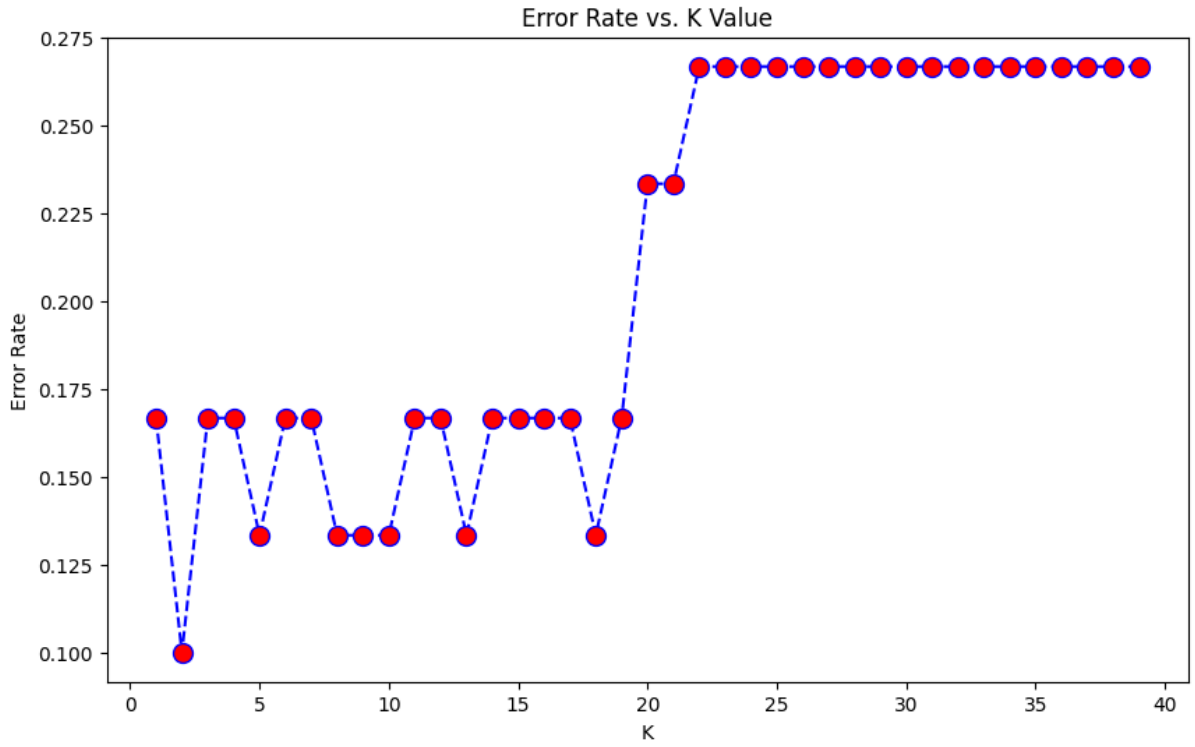



```
k = 3

clusters = {}
np.random.seed(23)

for idx in range(k):
        center = 2*(2*np.random.random((X.shape[1],))-1)
        points = []
        cluster = {
                'center' : center,
                'points' : []
        }

        clusters[idx] = cluster

clusters
```

{0: {'center': array([0.06919154]), 'points': []},
 1: {'center': array([1.78785042]), 'points': []},
 2: {'center': array([1.06183904]), 'points': []}}

```
def distance(p1,p2):
  return np.sqrt(np.sum((p1-p2)**2))

#Implementing E step
def assign_clusters(X, clusters):
  for idx in range(X.shape[0]):
    dist = []

    curr_x = X[idx]

    for i in range(k):
      dis = distance(curr_x,clusters[i]['center'])
      dist.append(dis)
    curr_cluster = np.argmin(dist)
    clusters[curr_cluster]['points'].append(curr_x)
  return clusters

#Implementing the M-Step
def update_clusters(X, clusters):
  for i in range(k):
    points = np.array(clusters[i]['points'])
    if points.shape[0] > 0:
      new_center = points.mean(axis =0)
      clusters[i]['center'] = new_center

      clusters[i]['points'] = []
  return clusters

def pred_cluster(X, clusters):
  pred = []
  for i in range(X.shape[0]):
    dist = []
    for j in range(k):
      dist.append(distance(X[i],clusters[j]['center']))
    pred.append(np.argmin(dist))
  return pred

clusters = assign_clusters(X,clusters)
clusters = update_clusters(X,clusters)
pred = pred_cluster(X,clusters)
```

```
from sklearn.metrics import classification_report,\
    confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

X_train, X_test,\
    y_train, y_test = train_test_split(scaled_features,
                                       df['Taregt'],
                                       test_size=0.30)

# Remember that we are trying to come up
# with a model to predict whether
# someone will Target or not.
# We'll start with k = 1.

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
pred = knn.predict(X_test)

# Predictions and Evaluations
# Let's evaluate our KNN model !


print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
from sklearn.metrics import classification_report,\
    confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

X_train, X_test,\
    y_train, y_test = train_test_split(scaled_features,
                                       df['Taregt'],
                                       test_size=0.30)

# Remember that we are trying to come up
# with a model to predict whether
# someone will Target or not.
# We'll start with k = 1.

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
pred = knn.predict(X_test)

# Predictions and Evaluations
# Let's evaluate our KNN model !
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```




```
# FIRST A QUICK COMPARISON TO OUR ORIGINAL K = 1
knn = KNeighborsClassifier(n_neighbors = 1)

knn.fit(X_train, y_train)
pred = knn.predict(X_test)

print('WITH K = 1')
print('Confusion Matrix')
print(confusion_matrix(y_test, pred))
print('Classification Report')
print(classification_report(y_test, pred))
```

```
WITH K = 1
Confusion Matrix
[[19  3]
 [ 2  6]]
Classification Report
              precision    recall  f1-score   support

           0       0.90      0.86      0.88        22
           1       0.67      0.75      0.71         8

    accuracy                           0.83        30
```

```
   macro avg       0.79      0.81      0.79        30
weighted avg       0.84      0.83      0.84        30
```

```
# NOW WITH K = 10
knn = KNeighborsClassifier(n_neighbors = 10)

knn.fit(X_train, y_train)
pred = knn.predict(X_test)

print('WITH K = 10')
print('Confusion Matrix')
print(confusion_matrix(y_test, pred))
print('Classification Report')
print(classification_report(y_test, pred))
```

```
WITH K = 10
Confusion Matrix
[[21  1]
 [ 3  5]]
Classification Report
              precision    recall  f1-score   support

           0       0.88      0.95      0.91        22
           1       0.83      0.62      0.71         8

    accuracy                           0.87        30
   macro avg       0.85      0.79      0.81        30
weighted avg       0.86      0.87      0.86        30
```