Day-4 | Salary Estimation | K-NEAREST NEIGHBOUR model

Importing Libraries

In [ ]:

```
import pandas as pd #useful for loading the dataset
import numpy as np #to perform array
```

Choose Dataset from Local Directory

In [ ]:

```
from google.colab import files
uploaded = files.upload()
```

Load Dataset

In [ ]:

```
dataset = pd.read_csv('salary.csv')
```

Summarize Dataset

In [ ]:

```
print(dataset.shape)
print(dataset.head(5))
```

Mapping Salary Data to Binary Value

In [ ]:

```
income_set = set(dataset['income'])
dataset['income'] = dataset['income'].map({'<=50K': 0, '>50K': 1}).astype(int)
print(dataset.head)
```

Segregate Dataset into X(Input/IndependentVariable) & Y(Output/DependentVariable)

In [ ]:

```
X = dataset.iloc[:, :-1].values
X
```

In [ ]:

```
Y = dataset.iloc[:, -1].values
Y
```

Splitting Dataset into Train & Test

In [ ]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25, ra
```

Feature Scaling

we scale our data to make all the features contribute equally to the result

Fit_Transform - fit method is calculating the mean and variance of each of the features present in our data

Transform - Transform method is transforming all the features using the respective mean and variance,

We want our test data to be a completely new and a surprise set for our model

In [ ]:

```python
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Finding the Best K-Value

In [ ]:

```python
error = []
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt

# Calculating error for K values between 1 and 40
for i in range(1, 40):
    model = KNeighborsClassifier(n_neighbors=i)
    model.fit(X_train, y_train)
    pred_i = model.predict(X_test)
    error.append(np.mean(pred_i != y_test))

plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

Training

In [ ]:

```python
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors = 2, metric = 'minkowski', p = 2)
model.fit(X_train, y_train)
```

Predicting, wheather new customer with Age & Salary will Buy or Not

In [ ]:

```
age = int(input("Enter New Employee's Age: "))
edu = int(input("Enter New Employee's Education: "))
cg = int(input("Enter New Employee's Captital Gain: "))
wh = int(input("Enter New Employee's Hour's Per week: "))
newEmp = [[age,edu,cg,wh]]
result = model.predict(sc.transform(newEmp))
print(result)

if result == 1:
  print("Employee might got Salary above 50K")
else:
  print("Customer might not got  Salary above 50K")
```

Prediction for all Test Data

In [ ]:

```
y_pred = model.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test
```

Evaluating Model - CONFUSION MATRIX

In [ ]:

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)

print("Confusion Matrix: ")
print(cm)

print("Accuracy of the Model: {0}%".format(accuracy_score(y_test, y_pred)*100)
```