Lab-03

January 25, 2026

1 KNN as Classifier [As Regressor in another program]

[1]:
```
#Assigning Feature and local variables
#First Feature
weather = ['Sunny','Sunny',
'Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny','Rainy',
'Sunny','Overcast','Overcast','Rainy']
```

[2]:
```
#Second Feature
temp =␣
 ↪['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mi
```

[3]:
```
# Label or Target Variable
play = ['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes',
'Yes','No' ]
```

[4]:
```
# import Label Encoder
from sklearn import preprocessing
```

[5]:
```
#creating Label Encoder
le = preprocessing.LabelEncoder()
```

[6]:
```
#converting Strigns Labels to numbers
weather_encoded = le.fit_transform(weather)
```

[7]:
```
weather_encoded
```

[7]:
```
array([2, 2, 0, 1, 1, 1, 0, 2, 2, 1, 2, 0, 0, 1])
```

[8]:
```
#converting Strigns Labels to numbers
temp_encoded = le.fit_transform(temp)
label_col = le.fit_transform(play)
temp_encoded
```

[8]:
```
array([1, 1, 1, 2, 0, 0, 0, 2, 0, 2, 2, 2, 1, 2])
```

[9]:
```
#combining weather and temp into single list of tuples
features= list(zip(weather_encoded,temp_encoded))
```

```
features
```

[9]:
```
[(np.int64(2), np.int64(1)),
 (np.int64(2), np.int64(1)),
 (np.int64(0), np.int64(1)),
 (np.int64(1), np.int64(2)),
 (np.int64(1), np.int64(0)),
 (np.int64(1), np.int64(0)),
 (np.int64(0), np.int64(0)),
 (np.int64(2), np.int64(2)),
 (np.int64(2), np.int64(0)),
 (np.int64(1), np.int64(2)),
 (np.int64(2), np.int64(2)),
 (np.int64(0), np.int64(2)),
 (np.int64(0), np.int64(1)),
 (np.int64(1), np.int64(2))]
```

[10]:
```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier (n_neighbors=3)
model.fit(features,label_col)
KNeighborsClassifier(n_neighbors=3)
```

[10]:
```
KNeighborsClassifier(n_neighbors=3)
```

[11]:
```
#predict output
predicted = model.predict([[1,0]])
predicted
```

[11]:
```
array([1])
```

[12]:
```
print(predicted)
```

```
[1]
```

[ ]:
```

```

KNN Classifier

January 25, 2026

1 KNN Classifier in Iris Dataset

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪classification_report
```

[2]:
```
iris = load_iris()
X = iris.data
y = iris.target

df = pd.DataFrame(X, columns=iris.feature_names)
df['target'] = y
df.head()
```

[2]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

[3]:
```
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2,␣
↪random_state=42 )
```

[4]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[5]:
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
```

[5]:
```
KNeighborsClassifier()
```

[6]:
```
y_pred = knn.predict(X_test)
y_pred
```

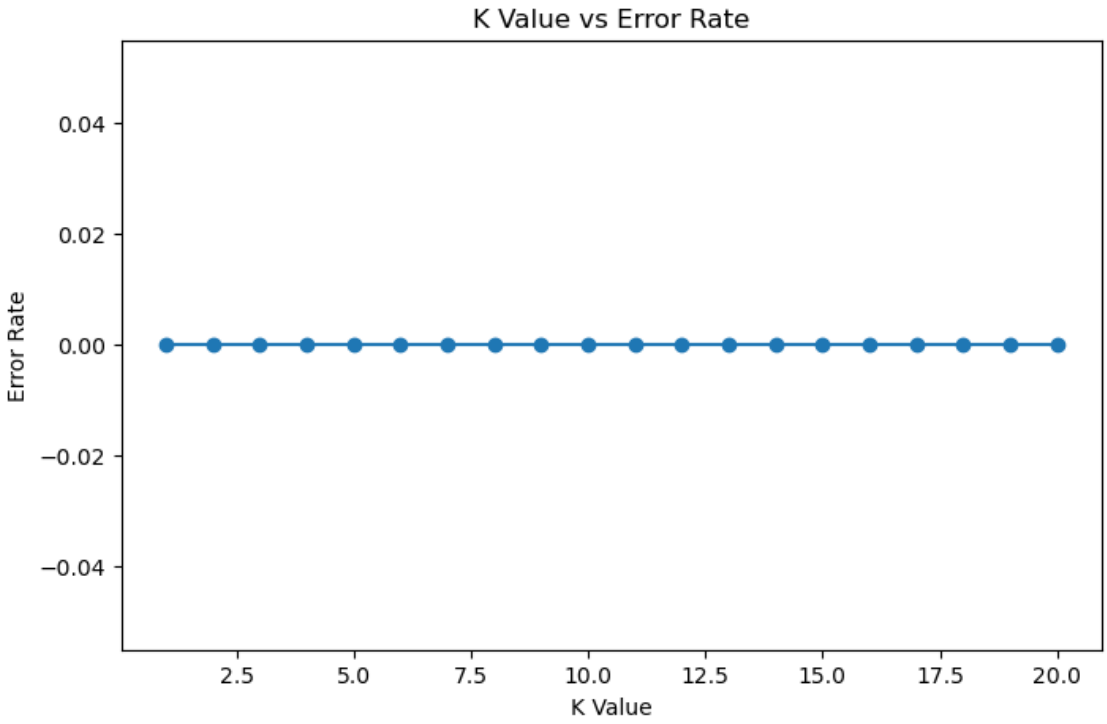
[6]:
```
array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2,
       0, 2, 2, 2, 2, 2, 0, 0])
```

[7]:
```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 1.0

Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

[8]:
```
error_rate = []

for k in range(1, 21):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    pred_k = knn.predict(X_test)
    error_rate.append(np.mean(pred_k != y_test))
```

```
plt.figure(figsize=(8,5))
plt.plot(range(1,21), error_rate, marker='o')
plt.xlabel('K Value')
plt.ylabel('Error Rate')
plt.title('K Value vs Error Rate')
plt.show()
```



[ ]:

KNN as Regrassior

January 25, 2026

0.1 KNN AS REGRESSOR

[1]:
```
#Step 1: Import Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
```

[2]:
```
#Step 2: Create the DataFrame
data = {
'id': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
'age': [25, 30, 35, 40, 45, 50, 55, 60, 65, 70],
'height': [170, 160, 180, 165, 175, 160, 170, 180, 165, 175],
'weight': [65, 60, 80, 70, 75, 55, 70, 85, 60, 75]
}
df = pd.DataFrame(data)
print(df)
```

```
   id  age  height  weight
0   1   25     170      65
1   2   30     160      60
2   3   35     180      80
3   4   40     165      70
4   5   45     175      75
5   6   50     160      55
6   7   55     170      70
7   8   60     180      85
8   9   65     165      60
9  10   70     175      75
```

[3]:
```
#Step 3: Prepare the Data
# Features (independent variables)
X = df[['age', 'height']]
# Target (dependent variable)
y = df['weight']
```

[4]:
```
#Step 4: Split the Data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

[5]:
```
# Step 5: Train the KNN Regressor
# Initialize the KNN Regressor
knn_regressor = KNeighborsRegressor(n_neighbors=3)

# Train the model
knn_regressor.fit(X_train, y_train)
KNeighborsRegressor(n_neighbors=3)
```

[5]:
```
KNeighborsRegressor(n_neighbors=3)
```

[6]:
```
#Step 6: Make Predictions
# Predict weights for the test set
y_pred = knn_regressor.predict(X_test)

# Display predictions
print("Predicted Weights:", y_pred)
```

```
Predicted Weights: [76.66666667 63.33333333]
```

[7]:
```
#Step 7: Evaluate the Model

# Calculate Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 144.44444444444454
```

[8]:
```
#Step 8: Predict Weight for New Data

# New input data
new_data = [[32, 172]]

# Predict weight
predicted_weight = knn_regressor.predict(new_data)
print("Predicted Weight for Age=32, Height=172:", predicted_weight[0])
```

```
Predicted Weight for Age=32, Height=172: 71.66666666666667

/usr/lib/python3/dist-packages/sklearn/utils/validation.py:2749: UserWarning: X
does not have valid feature names, but KNeighborsRegressor was fitted with
feature names
  warnings.warn(
```

[ ]:
```

```

Carprice-KNN Regrassor

January 25, 2026

1 KNN Regrassor in Carprice Dataset

[1]:
```
#Step 1: Import Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
```

[2]:
```
#Step 2: Read Data
df = pd.read_csv('carprices.csv')
```

[3]:
```
df.head(16)
```

[3]:

	Car Model	Mileage	Sell Price	Age
0	BMW X5	69000	18000	6
1	BMW X5	35000	34000	3
2	BMW X5	57000	26100	5
3	BMW X5	22500	40000	2
4	BMW X5	46000	31500	4
5	Audi	59000	29400	5
6	Audi	52000	32000	5
7	Audi	72000	19300	6
8	Audi	91000	12000	8
9	Mercedez Benz	67000	22000	6
10	Mercedez Benz	83000	20000	7
11	Mercedez Benz	79000	21000	7
12	Mercedez Benz	59000	33000	5
13	Toyota	51000	42000	4
14	Toyota	65000	32000	7
15	Toyota	39000	55000	5

[4]:
```
#Step 3: Prepare the Data

# Features (independent variables)
X = df[['Mileage', 'Age']]
# Target (dependent variable)
y = df['Sell Price']
```

[5]:
```
#Step 4: Split the Data

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

[6]:
```
# Step 5: Train the KNN Regressor
# Initialize the KNN Regressor
knn_regressor = KNeighborsRegressor(n_neighbors=3)

# Train the model
knn_regressor.fit(X_train, y_train)
KNeighborsRegressor(n_neighbors=3)
```

[6]:
```
KNeighborsRegressor(n_neighbors=3)
```

[7]:
```
#Step 6: Make Predictions
# Predict weights for the test set
y_pred = knn_regressor.predict(X_test)

# Display predictions
print("Predicted Sell Price:", y_pred)
```

```
Predicted Sell Price: [20766.66666667 42166.66666667 30366.66666667
24766.66666667]
```

[8]:
```
#Step 7: Evaluate the Model

# Calculate Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 31901111.111111097
```

[9]:
```
#Step 8: Predict Sell Price for New Data

# New input data
new_data = [[48000, 3]]

# Predict weight
predicted_price = knn_regressor.predict(new_data)
print("Predicted Sell Price for Using Age=3, Mileage=48000, Sell_Price=",␣
 ↪predicted_price[0])
```

```
Predicted Sell Price for Using Age=3, Mileage=48000, Sell_Price=
35166.666666666664

/usr/lib/python3/dist-packages/sklearn/utils/validation.py:2749: UserWarning: X
does not have valid feature names, but KNeighborsRegressor was fitted with
feature names
  warnings.warn(
```

[10]:
```
df.head(17)
```

[10]:

	Car Model	Mileage	Sell Price	Age
0	BMW X5	69000	18000	6
1	BMW X5	35000	34000	3
2	BMW X5	57000	26100	5
3	BMW X5	22500	40000	2
4	BMW X5	46000	31500	4
5	Audi	59000	29400	5
6	Audi	52000	32000	5
7	Audi	72000	19300	6
8	Audi	91000	12000	8
9	Mercedez Benz	67000	22000	6
10	Mercedez Benz	83000	20000	7
11	Mercedez Benz	79000	21000	7
12	Mercedez Benz	59000	33000	5
13	Toyota	51000	42000	4
14	Toyota	65000	32000	7
15	Toyota	39000	55000	5

[ ]:
```

```