Machine Learning Lab

Practical - 05

Name : Prathamesh Rajbhoj

Roll : 53

Batch : A-2

5(a) Implementation of KNN algorithm for regression as per given dataset.

```
# Number of neighbors (k)
k = 5


# Dataset
length = [10.0, 11.0, 12.0, 10.0, 7.0, 9.0, 8.0]
weight = [15.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0]
cost = [45, 150, 200, 250, 300, 350, 400]


# Test dataset
test_length = 4.7
test_weight = 2.2


# Function to calculate Euclidean distance
def euclidean_distance(point1, point2):
    return ((point1[0] - point2[0]) ** 2 + (point1[1] - point2[1]) ** 2) ** 0.5


# Calculate distances to all data points
distances = [(euclidean_distance((test_length, test_weight), (length[i], weight[i])), cost[i]) for i in range(len(length))]


# Sort the distances
sorted_distances = sorted(distances, key=lambda x: x[0])


# select the top k neighbors
k_nearest_neighbors = sorted_distances[:k]


# Calculate the weighted average of the costs of the k nearest neighbors
weighted_cost_sum = sum(neighbor[1] / neighbor[0] for neighbor in k_nearest_neighbors)
weighted_distance_sum = sum(1 / neighbor[0] for neighbor in k_nearest_neighbors)
predicted_cost = weighted_cost_sum / weighted_distance_sum



print(f"Predicted cost for (length={test_length}, weight={test_weight}): {predicted_cost:.2f}")
```

```
Predicted cost for (length=4.7, weight=2.2): 303.73
```

5(b) Implementation of KNN algorithm for classification using Iris dataset.

```
# Import libraries
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix


# Load the Iris dataset
iris = datasets.load_iris()


X = iris.data
y = iris.target
```

```
# Data Splitting
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


# Create a KNN classifier
k = 5
knn_classifier = KNeighborsClassifier(n_neighbors=k)


# Model fitting
knn_classifier.fit(X_train, y_train)


# Prediction
y_pred = knn_classifier.predict(X_test)
```

```
# Model Evaluation
confusion_mat = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print("Confusion Matrix : ", end="\n\n")
print(confusion_mat, end="\n\n\n\n")
print("\nClassification Report : ", end="\n\n")
print(classification_rep)
```

```
Confusion Matrix : 

[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]




Classification Report : 

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

```
# Testing on New data
new_data = np.array([[5.1, 3.5, 1.4, 0.2]])


predicted_class = knn_classifier.predict(new_data)

predicted_class_label = iris.target_names[predicted_class[0]]


print(f"The predicted class for new_data is: {predicted_class_label}")
```

```
The predicted class for new_data is: setosa
```