KNN

PRN: 2019BTECS00067

Aim:
Implementation of knn.

Theory:

Step 1: Calculate Euclidean Distance.
Step 2: Get Nearest Neighbors.
Step 3: Make Predictions.

```
# The k-nearest neighbor algorithm is imported from the scikit-learn
package.
# Create feature and target variables.
# Split data into training and test data.
# Generate a k-NN model using neighbors value.
# Train or fit the data into the model.
# Predict the future.
```

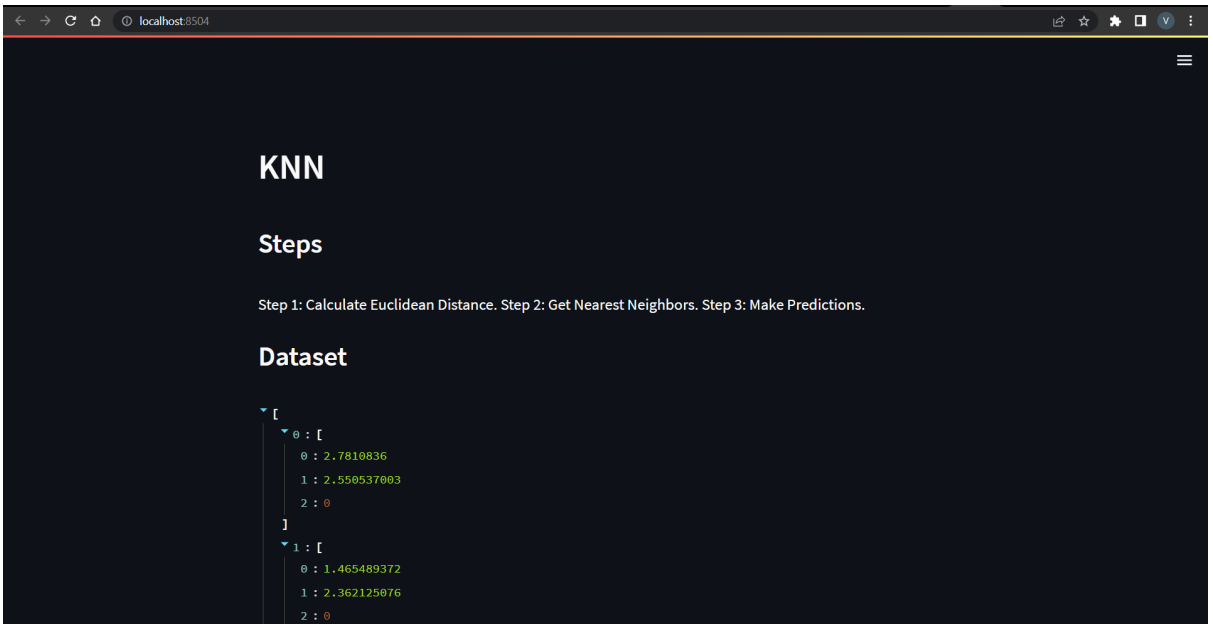
```
# Load the training data.
# Find the optimal value for K:
# Predict a class value for new data:
# Calculate distance(X, Xi) from i=1,2,3,….,n.
# where X= new data point, Xi= training data, distance as per your chosen
distance metric.
# Sort these distances in increasing order with corresponding train data.
# From this sorted list, select the top 'K' rows.
# Find the most frequent class from these chosen 'K' rows. This will be
your predicted class.
```

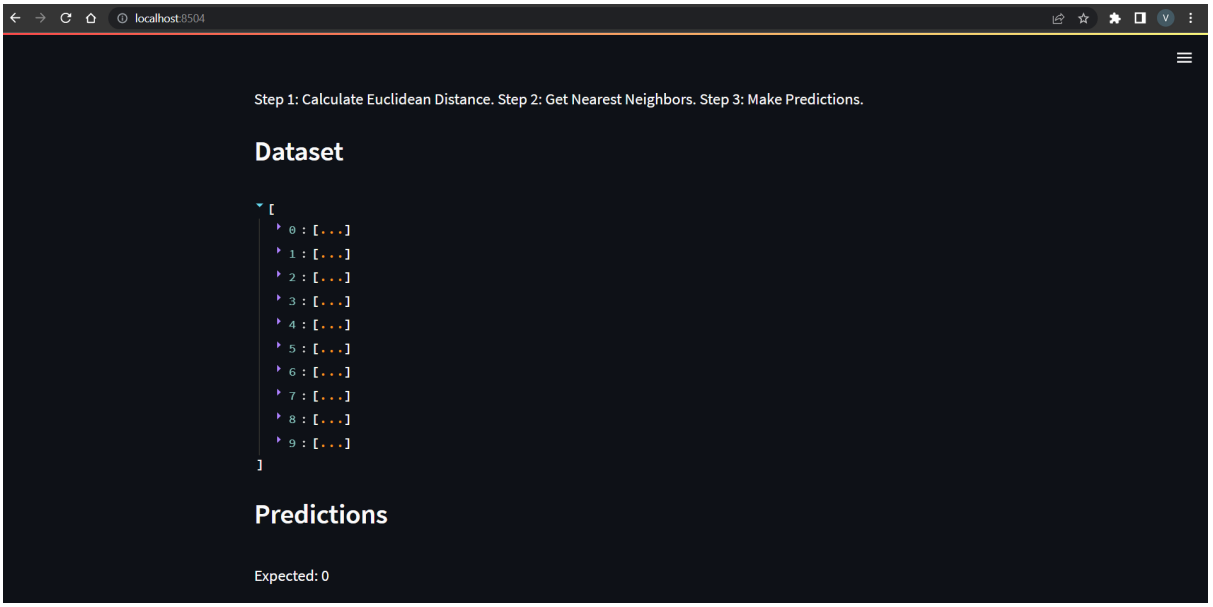
```
# calculate the Euclidean distance between two vectors
def euclidean_distance(row1, row2):
  distance = 0.0
  for i in range(len(row1)-1):
      distance += (row1[i] - row2[i])**2
  return sqrt(distance)
```

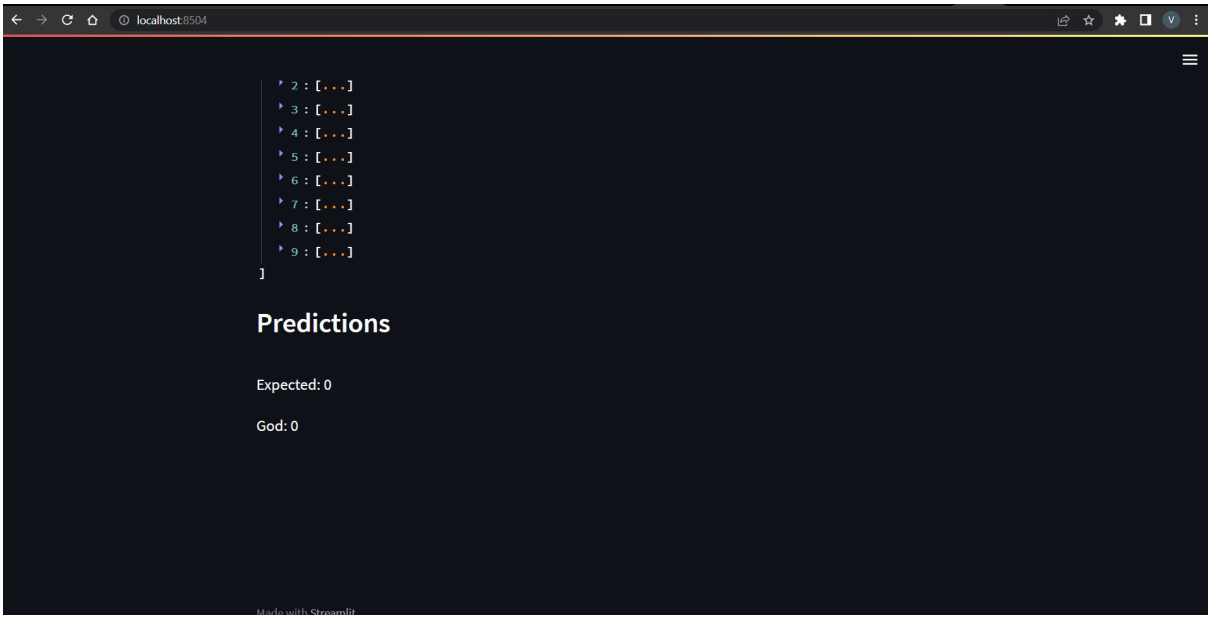
```python
# Locate the most similar neighbors
def get_neighbors(train, test_row, num_neighbors):
  distances = list()
  for train_row in train:
     dist = euclidean_distance(test_row, train_row)
     distances.append((train_row, dist))
  distances.sort(key=lambda tup: tup[1])
  neighbors = list()
  for i in range(num_neighbors):
     neighbors.append(distances[i][0])
  return neighbors
```

```python
# Make a classification prediction with neighbors
def predict_classification(train, test_row, num_neighbors):
  neighbors = get_neighbors(train, test_row, num_neighbors)
  output_values = [row[-1] for row in neighbors]
  prediction = max(set(output_values), key=output_values.count)
  return prediction
```

Result and Output snapshots:







Conclusion:

By using the knn algorithm and basic python coding knn is implemented. Predictions and Got is shown using GUI.