Answers - KNN Questions

THEORY

Question 1:

The KNN (K-Nearest Neighbors) algorithm does not generate a model in the traditional sense, like a decision tree or a neural network. Instead, KNN is an instance-based algorithm, meaning it simply stores all the training data and makes predictions based on comparing new input data with the training set.

To deploy KNN in a real-world (production) scenario, you would need to:

1. **Store the training dataset:** Keep the data used for training available in production, as KNN needs to access this data to make predictions.
2. **Calculate distances:** When a new input arrives, calculate the distance between this input and all data in the training set.
3. **Identify the nearest neighbors:** Based on the smallest distance, select the k nearest neighbors.
4. **Make the prediction:** Classification or regression will be done based on the majority class of the k neighbors (for classification) or the average of the responses of the k neighbors (for regression).

This technique would be used in applications where simplicity and ease of interpretation are important, such as in recommendation systems or in simple classification problems.

Question 2:

In addition to Euclidean distance, several other distance metrics can be used in KNN, including:

Manhattan Distance (or L1): d(p,q)=∑∣pi−qi∣d(p, q) = \sum |p_i - q_i|d(p,q)=∑∣pi−qi∣

Minkowski Distance (generalization of Euclidean and Manhattan distances):
d(p,q)=(∑∣pi−qi∣p)1/pd(p, q) = \left(\sum |p_i - q_i|^p \right)^{1/p}d(p,q)=(∑∣pi−qi∣p)1/p

Where p=2p = 2p=2 results in Euclidean distance, and p=1p = 1p=1 results in Manhattan distance.

Chebyshev Distance: d(p,q)=max(|pi−qi|)d(p, q) = \max(|p_i - q_i|)d(p,q)=max(∣pi−qi∣)

Cosine Distance: d(p,q)=1−(p⋅q)(∣∣p∣∣ ∣∣q∣∣)d(p, q) = 1 - \frac{(p \cdot q)}{(||p|| \, ||q||)}d(p,q)=1−(∣∣p∣∣∣∣q∣∣)(p⋅q)

Where p⋅qp \cdot qp⋅q is the dot product of vectors ppp and qqq, and ∣∣p∣∣||p||∣∣p∣∣ and ∣∣q∣∣||q||∣∣q∣∣ are the norms of the vectors.

Question 3

To estimate an assumed income using KNN in a regression task, you would follow these steps:

1. **Define the training set:** Have a dataset with examples of assumed income and associated predictor variables (such as age, occupation, etc.).
2. **Choose the value of k:** Select the number of nearest neighbors to be considered in the estimation.
3. **Calculate distances:** For a new entry, calculate the distance between this entry and all entries in the training set.
4. **Select the k nearest neighbors:** Identify the k records in the training set that have the smallest distances to the new entry.
5. **Calculate the estimated income:** The assumed income would be the average of the incomes of the selected k neighbors.

This approach would be useful in cases where the relationship between the predictor variables and the income is not linear and where it is important to capture the influence of local data.

PRACTICE

Knn_1.py:

```
-------------------------------------------

Regressão Logística  - Erro de Classificação: 0.26883910386965376

Regressão Logística  - Acurácia: 0.7311608961303462

KNN                  - Erro de Classificação: 0.265173116089613

KNN                  - Acurácia: 0.734826883910387

Árvore               - Erro de Classificação: 0.25784114052953155

Árvore               - Acurácia: 0.7421588594704684

--------------------------------------------
```

Knn_2.py:

Matplotlib is building the font cache; this may take a moment.

----

k Erro de Classificação

1 0.29816700610997965

5 0.2814663951120163

10 0.2729124236252546

15 0.2725050916496945

20 0.2708757637474542

25 0.2659877800407332

30 0.265173116089613

35 0.26313645621181264

40 0.2615071283095723

45 0.2619144602851324

50 0.2606924643584521

55 0.26232179226069247

60 0.2655804480651731

65 0.26924643584521385

70 0.2680244399185336

75 0.2659877800407332

80 0.2655804480651731

85 0.26924643584521385

90 0.2704684317718941

95 0.2704684317718941

100 0.2720977596741344

105 0.27128309572301423

110 0.27128309572301423

115 0.2716904276985743

120 0.26883910386965376

125 0.2708757637474542

130 0.2708757637474542

135 0.27128309572301423

140 0.2704684317718941

145 0.2716904276985743

150 0.270061099796334

155 0.26965376782077394

160 0.2704684317718941

165 0.26883910386965376

170 0.270061099796334

175 0.26965376782077394

180 0.2708757637474542

185 0.2716904276985743

190 0.2704684317718941

195 0.27128309572301423

200 0.2720977596741344

------------------------- Gráfico -----------------------------

Question 1

After running the Knn_1.py code, the following results were obtained:

- **Logistic Regression:** Classification Error: 0.2688, Accuracy: 0.7312
- **KNN:** Classification Error: 0.2652, Accuracy: 0.7348
- **Decision Tree:** Classification Error: 0.2578, Accuracy: 0.7422

Justification: The Decision Tree model had the lowest Classification Error (0.2578) and the highest Accuracy (0.7422). Therefore, the Decision Tree model is the best for classification in this case, as a lower classification error and higher accuracy indicate a better model.

Question 2

After running the Knn_2.py code, a graph of Classification Error versus the value of k was generated. Observing the graph and the obtained values, the lowest Classification Error was for k = 50, with an error of 0.2607.

Justification: The best value of k is the one that minimizes the classification error. In the generated graph, k = 50 showed the lowest classification error, being 0.2607. Therefore, this is the best k for the test sample.