CMPE 251 Data Analytics
Assignment 2: Theoretical

Question 1: Finding Nearest Neighbors

Calculate the Euclidean distance from a new point *N(3,4)* to each point and determine the two nearest neighbors.

P1(2,3)

$$d_E(N, P1) = \sqrt{(2-3)^2 + (3-4)^2}$$

$$d_E(N, P1) = \sqrt{2} = 1.4142$$

Equation 1: Euclidean distance between point N and P1.

P2(3,5)

$$d_E(N, P2) = \sqrt{(3-3)^2 + (5-4)^2}$$

$$d_E(N, P2) = 1.0$$

Equation 2: Euclidean distance between point N and P2.

P3(5,4)

$$d_E(N, P3) = \sqrt{(5-3)^2 + (4-4)^2}$$

$$d_E(N, P3) = 2.0$$

Equation 3: Euclidean distance between point N and P3.

P4(6,2)

$$d_E(N, P4) = \sqrt{(6-3)^2 + (2-4)^2}$$

$$d_E(N, P4) = \sqrt{13} = 3.6055$$

Equation 4: Euclidean distance between point N and P4.

P5(4,7)

$$d_E(N, P5) = \sqrt{(4-3)^2 + (7-4)^2}$$

$$d_E(N, P5) = \sqrt{10} = 3.1623$$

Equation 5: Euclidean distance between point N and P5.

As we can observe above the Euclidean distances which yield the smallest results are from Equation 1 and Equation 2. Therefore, we know that the two nearest neighbors to the new instances *N* are points P1 and P2.

Question 2: Impact of k Value

When implementing the k nearest neighbors' algorithm it is important to select a suitable k value as the number selected will skew the output results. Both the number of instances *n* as well as the number of classes *c* need to be considered when selecting a k value.

Considering a value of k=1

- In this case the model will only consider the very closest neighbor when making the predictions.
- The output will reflect the class of the sole nearest neighbor and therefore does not consider more complex relationships within the dataset.
- Since the model will simply be memorizing things, it will lack the ability to generalize data.
- In almost every case a k value of one is not sufficient to make reliable predictions.

Considering a value of k=3

- Depending on both the number of instances and the number of classes within a dataset, k=3 could be a sufficient option for the k value.
- For example, in a dataset comparing two categorical features, having a k value of three will provide the model with multiple data points while also always determining a majority classification.
- For a dataset with less instances this would be a good option since it will provide higher perspective to make more accurate predictions, though will also consider the sensitivity of a small dataset to a sizeable k value.

Considering a value of k=5

- Similar to assigning a k value of 3, k=5 is also a reliable option dependent on instances and classes.
- Especially in datasets with multiple categorical classes which a point can be assigned to, having more data points available for instance classification is desired. A k value of five will provide the model with several points such that if there are a number of classes the points could fall into, it's more likely that there will be a majority for the model to select.
- Furthermore, the model will be provided with more data to make an accurate prediction.

Considering a value of k=100

- A k value of 100 is rather high however, the success of the model will be completely dependent on the dataset.

- If we are considering a dataset that provides millions of instances, it would not be sufficient to set the k value as k=5 for example. This would simply not be enough data to correctly classify a data point with confidence.
- Moreover, in larger datasets there is likely a higher number of classes for which a point could be assigned to. Let's say there are 10 different categories, and we are using a k value of 5, those 5 reference points could easily all belong to different categories and we therefore we don't have nearly enough useful information to make an accurate prediction.
- However, if we are considering a dataset with *n* less than or equal to 100 then there would be an issue with selecting a k value of 100 since we are essentially setting k=n. In the case that k=n we are considering all data points within the dataset as neighbors, regardless of their similarity and closeness to the query point. Eventually all points, relevant or otherwise will be considered neighbors and the point will just be classified into the category which falls as the mode of the dataset.

In summary, the selection of a k value will be totally dependent on the both the number of instances within a dataset as well as the number of categories for which a point may be classified.

Question 3: Weighted KNN

For the knn algorithm prediction is based off of the majority class among the k nearest neighbors. With weighted knn all the nearest neighbors are additionally assigned a weight based on their distance to the query point. The idea is to assign higher weights to the neighbors which are closest to the examined point and lower weights to those farther away. This way the neighbors nearby the point have a higher influence of the prediction made. The following formula is used for a weighted knn model, where *y* is the predicted label or value for the data point *p*.

$$y = \sum_{k \in K} \frac{1}{dist(k,p)} * k_{label}$$

The algorithm for weighted knn is as follows:

1. Find the nearest k points to the new data point.
2. Take the weighted average of their label based on the Euclidean of the k nearest neighbors.
3. Assign the label.

Weighted knn is a more complex approach to the knn algorithm and therefore will make more accurate predictions comparatively.

Question 4: Distance Metrics

The choice of distance metric used directly effects the results of the knn algorithm. The distance metric determines how the model will calculate the 'closeness' of two points. The nearest neighbors must be determined by measuring the distance between the new point to be classified and the k surrounding data points. There are several distance metrics which can be selected when implementing knn. Each of these measures are summarized below:

Euclidean distance: measures the straight-line distance between points by using the hypotenuse. It assumes that all features contribute equally to the distance. The following equation is used to calculate Euclidean distance assuming instances of *j* and *k.*

$$d_E(j,k) = \sqrt{(x_{1j} - x_{1k})^2 + (x_{2j} - x_{2k})^2 + \cdots \left(x_{pj} - x_{pk}\right)^2} = \left(\sum_{i=1}^{p} \left(x_{ij} - x_{ik}\right)^2\right)^{1/2}$$

Manhattan distance: also known as the city block distance, this method is based on absolute difference between the points. It is a measure of the distance along the axes at right angles. It can be more appropriate if the data has different ranges or if the point haven't been normally distributed. The equation for Manhattan distance is as follows:

$$d_M(j,k) = \sum_{i=1}^{p} \left|x_{ij} - x_{ik}\right|$$

Minkowski distance: this is a generalized distance measure, with the power allowed to vary. It offers more flexibility in tuning as a generalization between Euclidean and Manhattan distance. The Minkowski distance can be calculated as follows:

$$d_m(j,k) = \left(\sum_{i=1}^{p} \left|x_{ij} - x_{ik}\right|^m\right)^{1/m}$$

Cosine similarity: is used to check if vectors are pointing in the same direction. This equation measures the cosine of the angle between two vectors in a multidimensional space. The values will range from -1, completely dissimilar, to 1, which would represent two vectors pointing in the exact same direction. This distance metric can be used to compare similarity between text data with high dimensionality as well as the similarity of document vectors. The equation is as follows:

$$\cos(\emptyset) = \frac{A * B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Several issues may arise if the wrong distance metric is selected for the knn algorithm. A poor decision may directly impact the effectiveness of the predictions made. Some of these impacts are listed below.

1. Inaccurate distance calculation:

The distance metric determines the points to be considered the k nearest neighbors of the given query point *p*. Upon incorrect selection of distance metric, points that may realistically be farther apart in the space might be considered closer and therefore the wrong classes are considered when making the predictions.

2. Sensitivity to feature scaling:

An improper distance metric selected will amplify scaling issues. For example, the Euclidean distance is sensitive to scale, if the data isn't normalized a feature may disproportionately influence the calculation while the model ignores smaller scale features.

3. Inappropriate for high dimensional data:

If the metric selected does not properly reflect the high dimensional space well the effectiveness of certain metrics will be hindered. In a high dimensional space, a metric like Euclidean distance become less meaningful since points that may differ in location appear to be the same closeness from query point *p*.

Question 5: Odds and Log-Odds

In the context of logistic regression, *odds* would be considered the ratio of the probability of an event or a positive class occurring to the probability of the negative class occurring and can be represented mathematically as follows:

$$odds = \frac{(y_i = 1)}{(y_i = 0)}$$

Log odds are considered the natural log ln() of the ratio of probability that $y_i = 1$ to the probability that $y_i = 0$. We can represent the probability of the output feature as p_i therefore this metric can be calculated using the following equation:

$$g_i = \ln(\frac{p_i}{1 - p_i})$$

Log odds represented by g_i range from $-\infty$ and $+\infty$, alternate to p_i taking on the bounds of 1 to 0. Logistic regression is a binary classification model which uses a linear function to predict the log odds of a given outcome. For single input features the regression model takes the following format:

$$\ln(\frac{p_i}{1-p_i}) = w_0 + w_1 x_i$$

$$p_i = \frac{\exp\,(w_0 + w_1 x_i)}{1 + \exp\,(w_0 + w_1 x_i)}$$

The predicted values p_i from logistic regression do not correspond to the actual classes. Alternately, p_i represents the probability that the new point p has $y_i = 1$.

If a logistic regression coefficient w_j is 0.5, what is the effect of a one-unit increase in x_j on the odds of the positive class?

In this case, w_j represents the coefficient for a given feature value x_j. If the log reg coefficient is 0.5 and there is a one unit increase in feature value x_j, log odds of the positive class will increase by 0.5. To determine overall effect on odds we examine the impact of multiplication by e^{wj}.

$$effect\ on\ odds = exp^{w_j} = exp^{0.5} \approx 1.65$$

For every one unit increase, odds of the positive class increase by approximately 65%.

Question 6: Decision Boundary in Logistic Regression

The equation used for logistic regression is as follows:

$$P(y = 1|x) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)}}$$

- P(y=1|x) is the predicted probability of the positive class
- The coefficients are represented by w, these are the weights learned by the model
- And x values are the input features

The decision boundary is the set of points where the probability of the two classes is equal, alternately P(y=1|x) = 0.5. Subbing this back into the logistic regression equation:

$$0.5 = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)}}$$

$$2 = 1 + e^{-(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)}$$

$$1 = e^{-(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)}$$

$$0 = -(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)$$

$$0 = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

Therefore, we have derived the equation for the linear decision boundary for logistic regression.

Let's say we have an interaction term between two features. Feature x_1 and feature x_2 will have an interaction term $w_{12}x_1x_2$, w_{12} is the coefficient term. With two features and the interaction term the equation will be:

$$P(y = 1|x) = \frac{1}{1 + e^{-(w_0 + w_1x_1 + w_2x_2 + w_{12}x_1x_2)}}$$

With P(y=1|x) = 0.5 the decision boundary equation is found to be:

$$0 = w_0 + w_1x_1 + w_2x_2 + w_{12}x_1x_2$$

The inclusion of the interaction term $w_{12}x_1x_2$, causes the equation to be nonlinear. This means the resulting decision boundary will be a curve so long as $w_{12}x_1x_2$ is not equal to zero. This allows relationships of higher complexity to be captured in the model.

Question 7: LR Example

The equation for logistic regression takes the following form:

$$P(y = 1|x) = \frac{1}{1 + e^{-(w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n)}}$$

Given the example dataset:

X_1	X_2	y
2	1	0
3	2	0
4	3	1
5	4	1

We will have the following structure:

$$P(y = 1|x_1, x_2) = \frac{1}{1 + e^{-(w_0 + w_1x_1 + w_2x_2)}}$$

In order to determine the estimated weights, the sci-kit learn library provided by python was leveraged. Figure 1 displays the short program run to obtain the output which is provided in Figure 2 below.

```python
import numpy as np
from sklearn.linear_model import LogisticRegression

# Features x1 & x2
x = np.array([[2, 1], [3, 2], [4, 3], [5, 4]])
# Target variable y
y = np.array([0, 0, 1, 1])

logisticModel = LogisticRegression()
logisticModel.fit(x, y)

# intercept
w0 = logisticModel.intercept_[0]
# coefficients weights
w1, w2 = logisticModel.coef_[0]

print(f"w0: {w0}")
print(f"w1: {w1}")
print(f"w2: {w2}")
```

Figure 1: Logistic regression weight calculation code.

```
(CMPE251) laurensteel@Laurens-MacBook-Air-3 vscode % python logreg.py
w0: -4.084184387143448
w1: 0.6807217822088736
w2: 0.6806545364350418
```

Figure 2: Logistic regression weight calculation output.

Therefore, given the results we can develop the following logistic regression equation, assuming a linear combination of features:

$$P(y = 1|x_1, x_2) = \frac{1}{1 + e^{-(-4.0842 + 0.6807x_1 + 0.6806x_2)}}$$