INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA

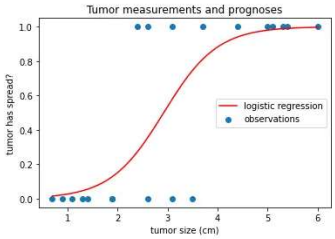
MCTA 4362: Machine Learning
(Sec. 1)
Quiz/Test #1

(LO 1)

Name: Muhamad Nurhakimie Thaqif Bin Abdullah Matric No: 2213217

Answer ALL Questions.

1) Calculate the likelihood for the following logistic regression problem. (you may estimate the values of some of the projections to the sigmoid curve since the graph here does not have a grid)



Tumor Size	Tumor Has Spread	Estimate Likelihood	Likelihood	Log(likelihood)
0.5	0	0.005	= 1- 0.005= 0.995	-0.0050
0.8	0	0.05	= 1- 0.05 = 0.95	-0.0513
1.2	0	0.07	= 1- 0.07 = 0.93	-0.0726
1.4	0	0.09	= 1- 0.09 = 0.91	-0.0943
1.5	0	0.12	= 1- 0.12 = 0.88	-0.1278
1.9	0	0.16	= 1- 0.16 = 0.84	-0.1744
2.4	1	0.30	= 0.30	-1.204
2.5	0	0.33	= 1- 0.33 = 0.67	-0.4005
2.6	1	0.35	= 0.35	-1.0498
3.0	1	0.58	= 0.58	-0.5447
3.2	0	0.59	= 1- 0.59 = 0.41	-0.8916
3.5	0	0.67	= 1- 0.67 = 0.33	-1.1087
3.7	1	0.80	= 0.80	-0.2231
4.2	1	0.90	= 0.90	-0.1054
4.9	1	0.94	= 0.94	-0.0619
5.0	1	0.95	= 0.95	-0.0513
5.2	1	0.97	= 0.97	-0.0305
5.4	1	0.98	= 0.98	-0.0202

6.0	1	1.0	= 1.0	0.0000
Total			1.995 10-3	-6.217

Therefore, The likelihood is around -6.217

2) Please refer to the following distribution of data which shows two classes of data. In a KNN algorithm what does this circle mean and what could be the impact of the diameter of this circle on your classification accuracy

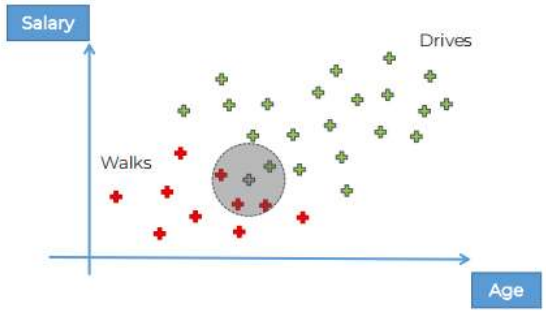


Figure 1: Walking and Driving Clusters of data.

The Circle means neighborhood around a query point, which signifies area KNN select Its nearest neighbor, the center of the circle is class that has not been classified.

The larger the area means larger K, which means more points can be considered therefore more stable as it selects more data points, and the smaller area means smaller K which means fewer points would be considered which allow to capture local points easier.

If the area is too large it would cause underfit as it would be influenced by other neighboring easier. However, if it is too small then it would be overfit as it would be easier to be influenced by noise.

3) Please refer to Figure 3 which shows three classes of data. Draw a Decision Tree which will distinguish the three classes.

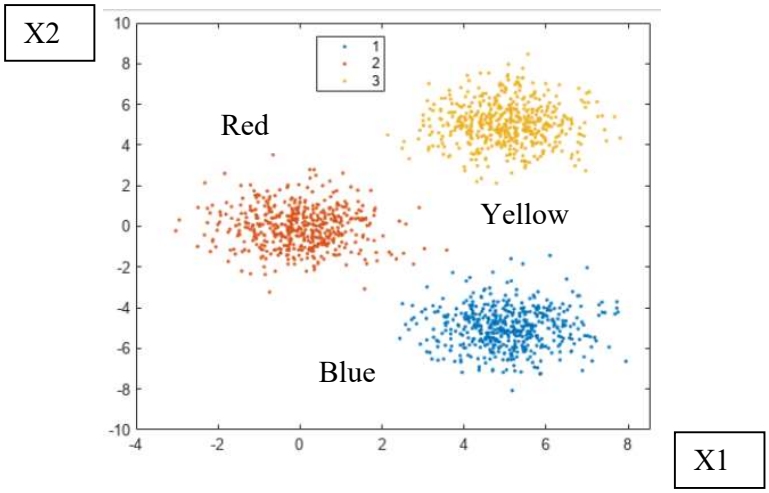
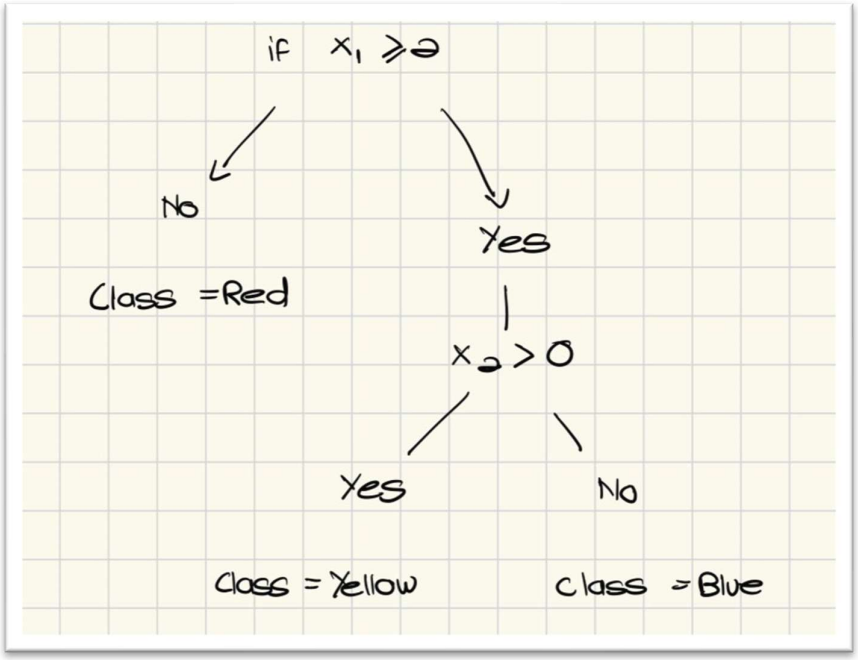


Figure 3: Three classes of data



4) The following dataset contains the characteristics of patients diagnosed with cancer. The dataset contains a unique ID for each patient, the type of cancer (diagnosis), the visual characteristics of the cancer and the average values of these characteristics. By using KNN, train a model to classify the two different classes.

Library

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

[32] ✓ 0.0s Python

Import Main Library

```
missing_values = dataset.isnull().sum() # count of missing values in each column
print(missing_values)
```

[35] ✓ 0.0s Open 'missing_values' in Data Wrangler

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
...
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64
```

Check for missing Value it looks like none

```
for i, col in enumerate(dataset.columns):
    print(f"{i}: {col}")
```

[36] ✓ 0.0s

```
0: id
1: diagnosis
2: radius_mean
3: texture_mean
4: perimeter_mean
5: area_mean
6: smoothness_mean
7: compactness_mean
8: concavity_mean
9: concave points_mean
10: symmetry_mean
11: fractal_dimension_mean
12: radius_se
13: texture_se
14: perimeter_se
15: area_se
16: smoothness_se
17: compactness_se
18: concavity_se
19: concave points_se
20: symmetry_se
21: fractal_dimension_se
22: radius_worst
23: texture_worst
24: perimeter_worst
...
28: concavity_worst
29: concave points_worst
30: symmetry_worst
31: fractal_dimension_worst
```

Get the Index Id in for checking if we want to use ID instead of Name Of header

```
print(dataset.dtypes)
```
[37] ✓ 0.0s

```
id                        int64
diagnosis                object
radius_mean             float64
texture_mean            float64
perimeter_mean          float64
area_mean               float64
smoothness_mean         float64
compactness_mean        float64
concavity_mean          float64
concave points_mean     float64
symmetry_mean           float64
fractal_dimension_mean  float64
radius_se               float64
texture_se              float64
perimeter_se            float64
area_se                 float64
smoothness_se           float64
compactness_se          float64
concavity_se            float64
concave points_se       float64
symmetry_se             float64
fractal_dimension_se    float64
radius_worst            float64
texture_worst           float64
perimeter_worst         float64
...
concave points_worst    float64
symmetry_worst          float64
fractal_dimension_worst float64
dtype: object
```

Checking for data type to know if we want to use one hot encoding or not

```
dataset_Cleaned = dataset.drop(columns='id')
```
[38] ✓ 0.0s

Remove ID as it is useless in training

Split

```
from sklearn.model_selection import train_test_split

X = dataset_Cleaned.drop(columns='diagnosis')
y = dataset_Cleaned['diagnosis']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```
[40] ✓ 0.0s

Split Data set to training 80 % Test 20 %

Feature Scaling

```
from sklearn.preprocessing import StandardScaler

# Select only numeric features
numeric_features = X.select_dtypes(include=['int64', 'float64']).columns

# Initialize the scaler
scaler = StandardScaler()

# Fit and transform the training data
X_train_scaled = scaler.fit_transform(X_train[numeric_features])

# Transform the test data
X_test_scaled = scaler.transform(X_test[numeric_features])

# Turn into DataFrames (optional but looks sexy)
X_train_df = pd.DataFrame(X_train_scaled, columns=numeric_features, index=X_train.index) # type: ignore
X_test_df = pd.DataFrame(X_test_scaled, columns=numeric_features, index=X_test.index) # type: ignore
```
[44] ✓ 0.0s

Apply Feature Scaling To make sure no bias of bigger data value happening and kept everything in range [3,-3]

KNN

```
from sklearn.neighbors import KNeighborsClassifier

classifier_KNN = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2) # p=2 is the default value for Euclidean distance
                                                                                    # p=1 is the default value for Manhattan distance
classifier_KNN.fit(X_train_df, y_train)
```

[46] ✓ 0.3s

```
KNeighborsClassifier
KNeighborsClassifier()
```

```
y_pred_KNN = classifier_KNN.predict(X_test_df)
```

[47] ✓ 0.4s

Doing KNN with 5 Neighbor and Metric Minkowski with p=2 for Euclidean

Visualize

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

print("Accuracy:", accuracy_score(y_test, y_pred_KNN))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_KNN))
print("Classification Report:\n", classification_report(y_test, y_pred_KNN, zero_division=0))
```

[48] ✓ 0.0s

```
Accuracy: 0.956140350877193
Confusion Matrix:
 [[67  0]
 [ 5 42]]
Classification Report:
              precision    recall  f1-score   support

           B       0.93      1.00      0.96        67
           M       1.00      0.89      0.94        47

    accuracy                           0.96       114
   macro avg       0.97      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

Visualize to check performance, here we got **95.61 %**