CSU44061 Machine Learning
Lab 2

Adriana Hrabowych, 19304296

November 2, 2023

1 Introduction

In this assignment, the dataset used in part 'i' has the ID of 19-19-19-0 and the dataset used in part 'ii' has the ID of 19–38-19-0 .

2 Part i

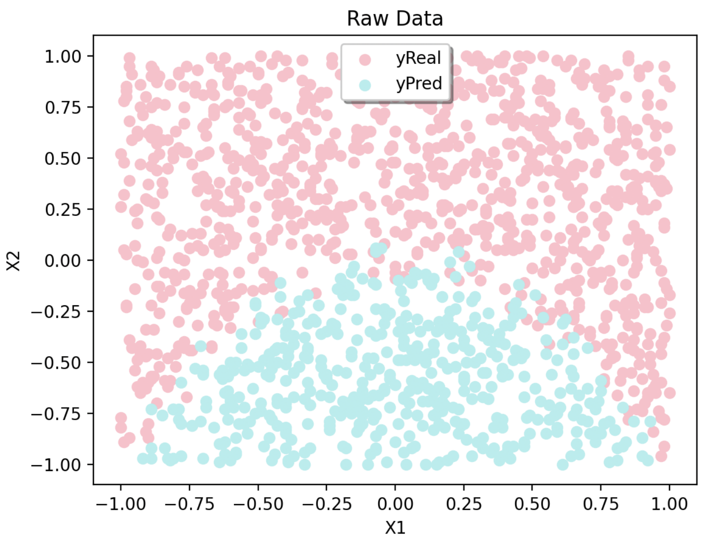

Figure 1: A scatter plot graph visualizing the first dataset.

2.1 (i)(a)

In (i)(a) we first read in the raw data and then visualize it on a graph as seen in figure one using circular markers. Each data point is placed depending on the value of its two features, with the X-axis corresponding to X1 and the Y-axis corresponding to X2. The color of the marker depends on the target value y, if the marker is red then y=1 and if blue then y=-1. As seen in figure one there is a clear decision boundary between the two classifications.

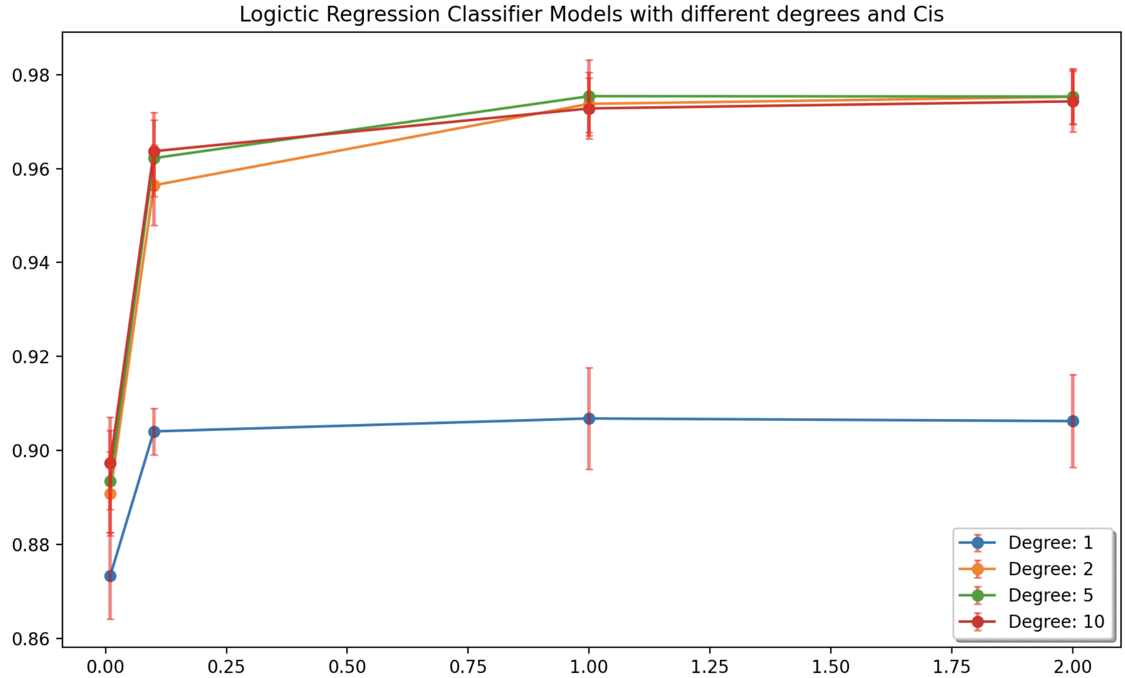

Figure 2: A plot with four lines, each representing a Logistic model trained with a different polynomial degree of X. Each line has four points with an error bar, each of them represent the f1 score mean and standard deviation of a model with a different C value.

In order to then train a logistic regression classifier on the data, we first have to use cross validation in order to choose the right polynomial degree of X and the correct weight for C. These are important to get right as the accuracy of a model can change drastically depending on the values.

During cross validation the range of polynomial degrees used was [1, 2, 5, 10]. I chose these values as they represent a good spread and represent the typical range of degrees used in logistic regression classifiers (which the addition of 1 and 10 as outliers). The range of C values used were [0.01, 0.1. 1, 2], chosen for the same reasons as the degrees. The default C used is typically 1, and then I added greater and lesser values to have a good spread.

The accuracy metric used in the following cross validation was the F1 score. The higher the score the better precision and recall a model has. The F1 score is calculated as:

P = # True Positives / (# True Positives + # False Positives)
R = # True Positives / (# True Positives + # False Negative)
F1 Score = 2 * (P * R) / (P + R)

F1 scores were chosen as it combines both precision and recall and gives good results when the dataset is imbalanced (as ours is, see figure 1). Also, MSE cannot be used as a metric for classification as it cannot compare discrete classes.

The graph in figure two represents the outcomes of cross validation. A model was trained on each combination of C values and degrees, with a total of 16 models. A five-fold cross validation is performed on each model, the f1 score of each iteration is calculated and a mean and standard deviation f1 scores taken for each model. Five-fold cross validation was chosen as it is commonly used.

As seen in figure two, the polynomial degree which consistently gives the higher f1 scores is five, and the C value which does the same is 1. Both of these outcomes make sense as these values are commonly the default for a logistic regression classifier.

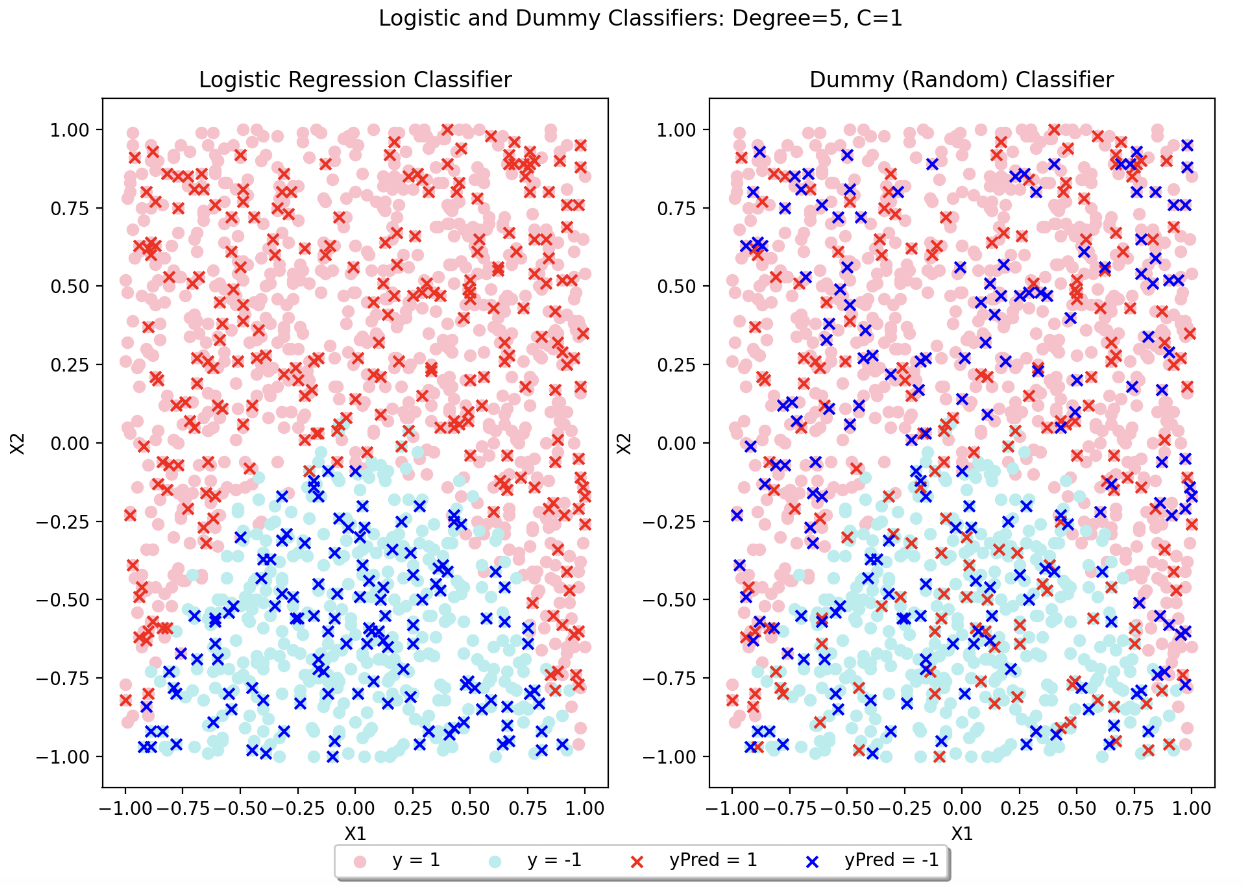

Figure 3: A figure with two scatter plots, the left one representing the predictions of the logistic regression classifier and the right being a random dummy model.

The data is then split between training and testing and the features are brought to an order of five. A logistic regression model (with a C of 1) and a dummy model (using an uniform random classification) are then trained and told to predict the testing data. The outcome of this is seen in figure 3. The LRC performed much better than the dummy classifier, with the f1 scores of each being 0.97899 and 0.58095 respectively.

2.2 (i)(b)

We then train a K-nearest neighbors classifier on the data. Knn, a point is predicted based on the most frequent classification of the points around it. The parameter 'k' defines the number of neighbors to use, we will use cross validation to choose this value. It is important to choose the correct k as too small of a k leads to overfitting while a too big k leads to under-fitting.

The values of k chosen for cross validation were [2, 5, 9, 14]. This spread was chosen as it represents both high, low, and 'typical' k's. Once again I used f1 scores and 5-fold cross validation. As there was only one value to cross validate instead of 2, there are only 4 models total. The outcome is shown in figure 4, from this graph we can see that a k of 5 and 9 produced the highest f1 scores, but 5 had a much lower standard deviation. As such we will be using a k of five in our optimal model.

The data is then split between training and testing sections. A knn model (with a k of 5) and a dummy model (using an uniform random classification) are then trained and told to predict the testing data. The outcome of this is seen in figure 5. The knn classifier performed much better than the dummy classifier, with the f1 scores of each being 0.97991 and 0.607305 respectively.

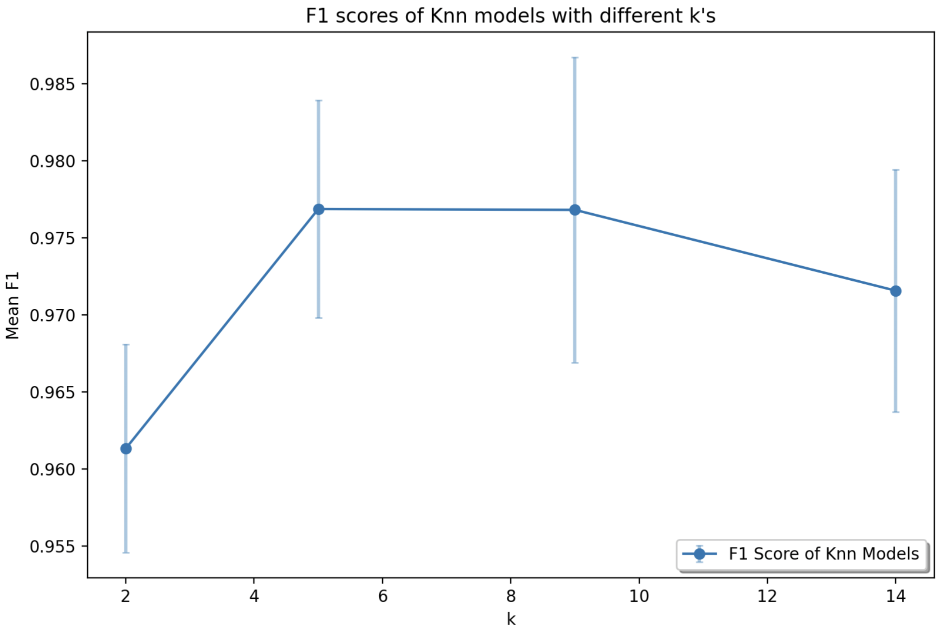

Figure 4: A plot with a singular line with four points, each point has an error bar. It represents the mean f1 scores of a knn model with different values of k.

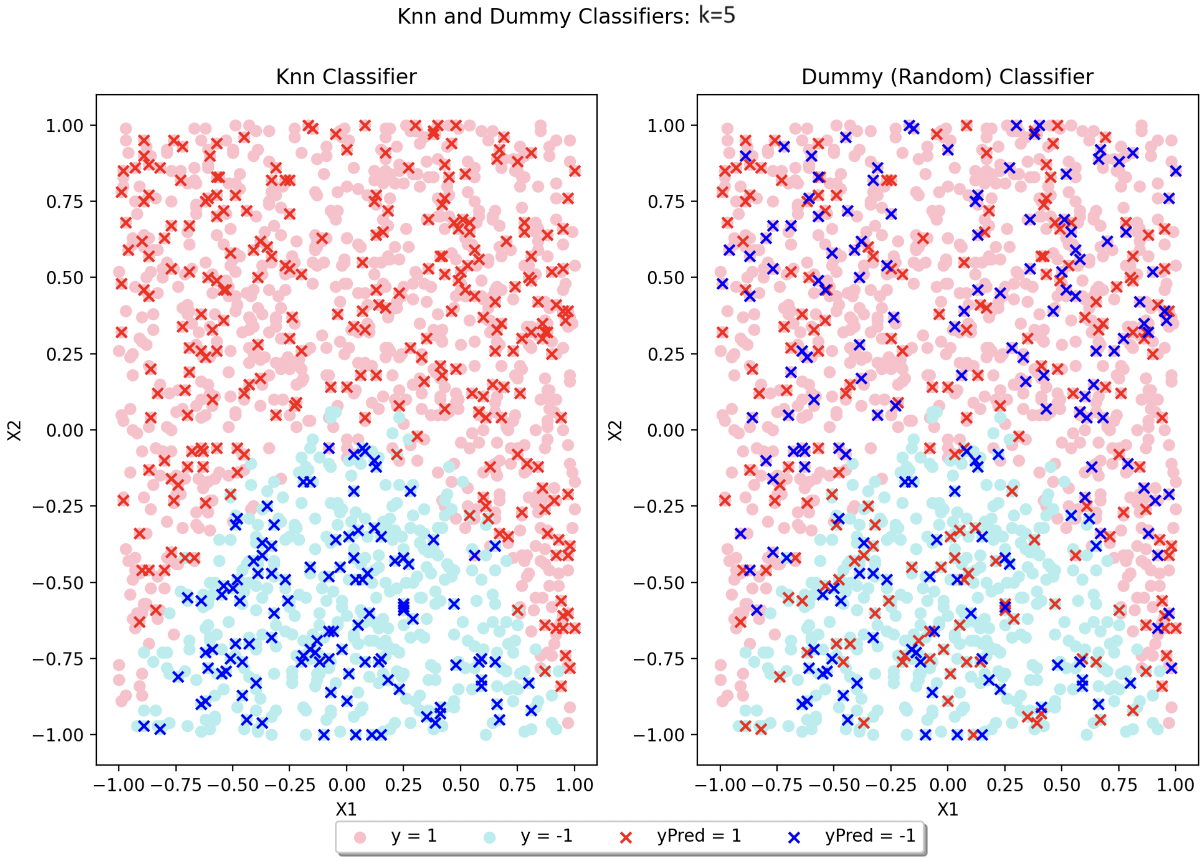

Figure 5: A figure with two scatter plots, the left one representing the predictions of the knn classifier and the right being a random dummy model.

2.3 (i)(c)

A confusion matrix is another type of performance metric that visualized the recall and precision in a table format. Data was split again between training and testing, and a LCR(C=1), Knn(k=5), and dummy (random) classifiers were all trained (with the LCR being trained with polynomial features to a degree of 5). They then made predictions based on the testing features and a confusion matrix was made by comparing those predictions to the real y values.

LCR	Predicted Positive	Predicted Negative
True Positive	238	2
True Negative	1	110

Dummy	Predicted Positiv	Predicted Negative
True Positive	109	131
True Negative	58	63

Knn	Predicted Positive	Predicted Negative
True Positive	236	4
True Negative	7	114

Figure 6: An image with 3 tables, each table being a confusion matrix for one of three models: LCR, Knn, and a dummy.

2.4 (i)(d)

An ROC curve is line that plots the true positive rate of a model against the false positive rate of a model at different classification thresholds. A classification threshold is the threshold that must be reached for a model to classify a point as positive. The formula for each rate are as follows:

$$TPR = TP \;/\; (TP + FN)$$
$$FPR = FP \;/\; (FP + TN)$$

Using the same data split and models as part C, the ROC curves of an LCR, Knn, and dummy model were plotted on the graph seen in figure 7.

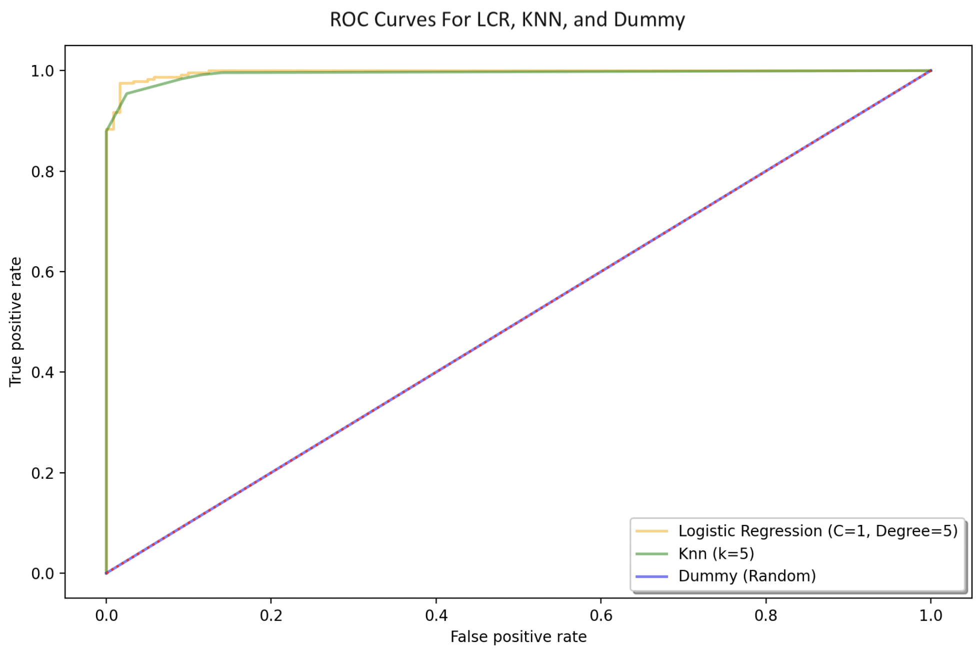

Figure 7: A graph with four lines, each representing the roc curve of a LCR, knn, and dummy model with a baseline 1-1 also plotted.

2.5 (i)(e)

The baseline dummy classifier performed far worse than both the LCR and Knn, as to be expected. It's confusion matrix had the most FPs and FNs and its ROC curve was practically the same as the baseline 1-1 line. The LCR performed slightly better than the Knn in the confusion matrix, as the number of FPs and FNS was only three, while Knn had 11 total. An ROC curve is considered better the closer the line is to an upside down L as it would mean that when the classification threshold is 1 the false positive rate is 0. The LCR again performs slightly better as it gets closer to the upper left corner than the Knn line does.

The f1 scores of each model are as follows: LCR = 0.97863, Knn = 0.97844, Dummy = 0.56937.

I would not be able to recommend a model for classification between the choice of LCR and Knn as the difference in the two in this example is not signifigant enough to be able to say for certain which is inherently better.

3 Part ii

3.1 (ii)(a)

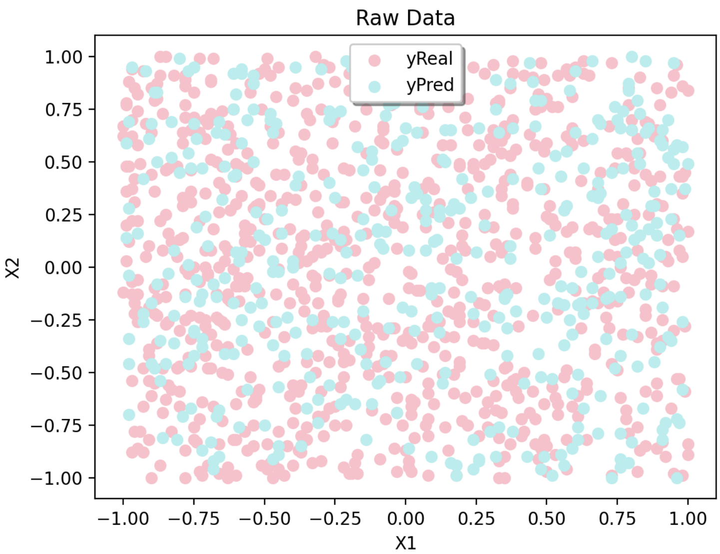

Figure 8: A scatter plot graph visualizing the second dataset.

In (ii)(a) we first read in the raw data and then visualize it on a graph as seen in figure one using circular markers. Each data point is placed depending on the value of its two features, with the X-axis corresponding to X1 and the Y-axis corresponding to X2. The color of the marker depends on the target value y, if the marker is red then y=1 and if blue then y=-1. As seen in figure 8, there is no clear decision boundary between the two classifications.

As in part (i)(a) we use the same form of cross validation with the same range of polynomial degrees and C values. The graph in figure 9 shows the outcomes of this cross validation. It is obvious from looking at the graph that this dataset does not seem to capture the same relationship as part i. Contrary to part (i) where the optimal degree was 5, for this dataset it seems to be 2. Though just barely as every combination of degree and C have extremely high standard deviations and have lower means than those in part (i)(a). The optimal C is either 0.1 or 1, we will use 1 as it is typically standard.

In figure 10 we can see the outcome of a LCR model with a C of 1 and degree 2 against a baseline random dummy model. It is interesting to note that the LCR model only ever predicted positive. The f1 scores were as follows: LCR = 0.81904, Dummy = 0.59893. The LCR still performed better than the dummy, but was much worse than in part i.

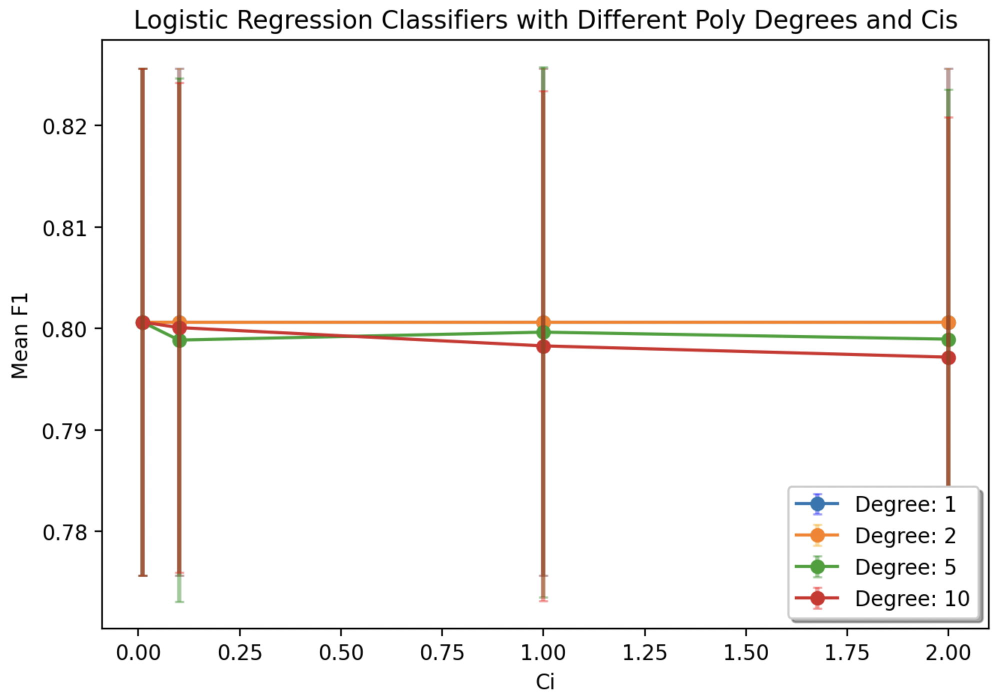

Figure 9: A plot with four lines, each representing a Logistic model trained with a different polynomial degree of X. Each line has four points with an error bar, each of them represent the f1 score mean and standard deviation of a model with a different C value.

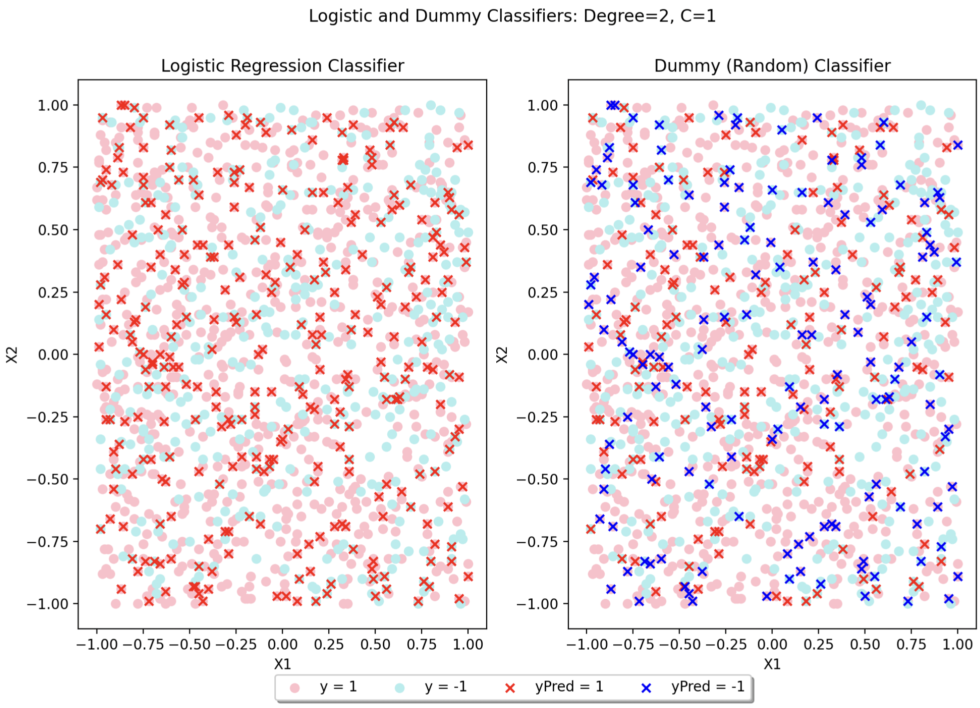

Figure 10: A figure with two scatter plots, the left one representing the predictions of the logistic regression classifier and the right being a random dummy model.

3.2 (ii)(b)

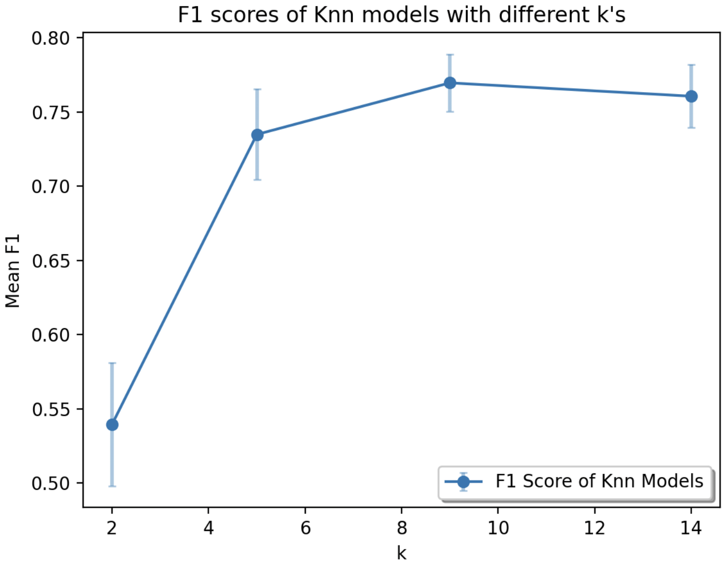

Figure 11: A plot with a singular line with four points, each point has an error bar. It represents the mean f1 scores of a knn model with different values of k.

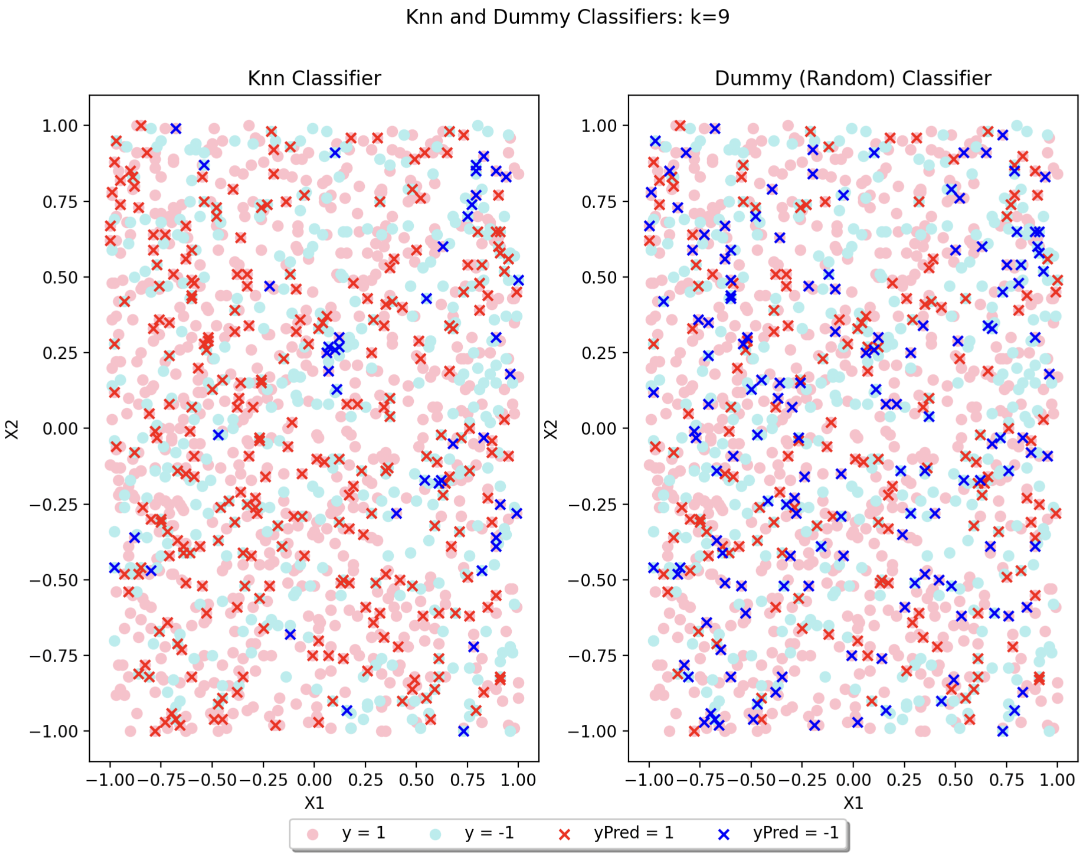

Figure 12: A figure with two scatter plots, the left one representing the predictions of the knn classifier and the right being a random dummy model.

As in part (i)(b) we use the same form of cross validation with the same range of k values. The graph in figure 11 shows the outcome. The optimal K for this dataset seems to be 9, not 5 like in the previous part. Once again we split the data and train a Knn model with a k of nine and a dummy model and then ask them to predict the test X's. The outcome of which is shown on figure 12. The f1 scores were as follows: Knn = 0.75213, Dummy = 0.5589. The Knn still performed better than the dummy, but was much worse than in part i.

3.3 (ii)(c)

Confusion matrices were made for the LCR (degree=2, C=1), Knn (k=9), and dummy models in the same way as in part i. They can be seen in figure 13.

LCR	Predicted Positive	Predicted Negative
True Positive	206	0
True Negative	104	0

Dummy	Predicted Positiv	Predicted Negative
True Positive	114	92
True Negative	49	55

Knn	Predicted Positive	Predicted Negative
True Positive	179	27
True Negative	86	18

Figure 13: An image with 3 tables, each table being a confusion matrix for one of three models: LCR, Knn, and a dummy.

3.4 (ii)(d)

Using the same data split and models as part C, the ROC curves of an LCR, Knn, and dummy model were plotted on the graph seen in figure 14.

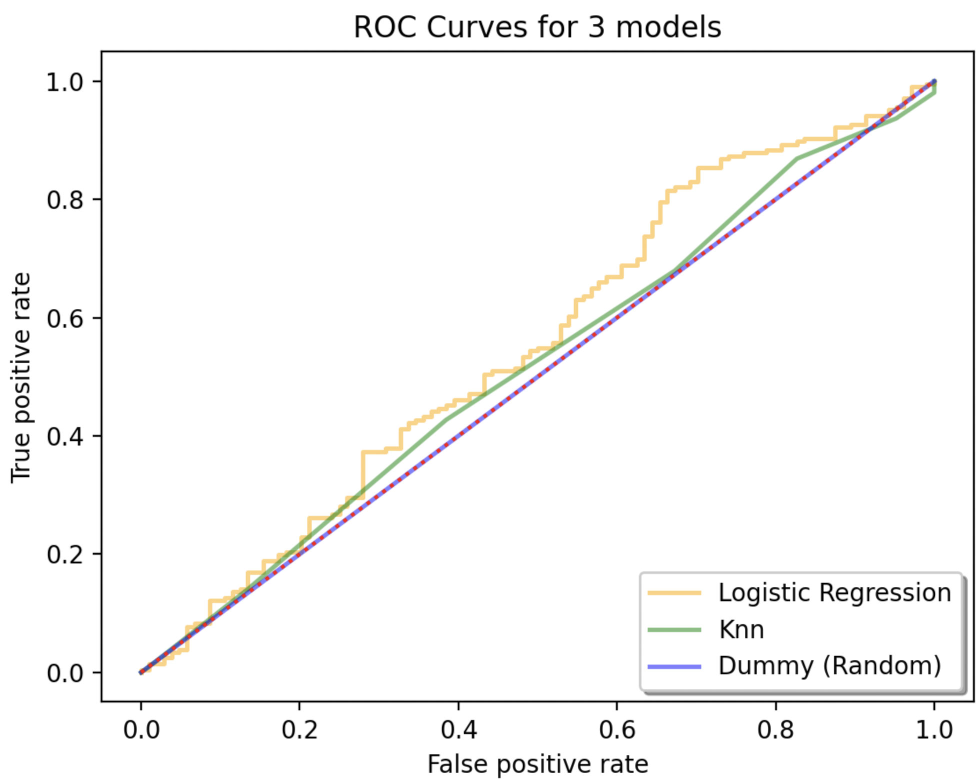

Figure 14: A graph with four lines, each representing the roc curve of a LCR, knn, and dummy model with a baseline 1-1 also plotted.

3.5 (ii)(e)

The baseline dummy classifier performed worse than both the LCR and Knn, as to be expected. It's confusion matrix had the most FPs and FNs and its ROC curve was practically the same as the baseline 1-1 line. Both the LCR and Knn model performed far worse than in the previous parts, this is because the dataset is not useful and there wasn't any clear decision boundary between the two classifications. The LCR performed slightly better than the Knn in the confusion matrix, as the number of FPs and FNS was 104, while Knn had 113 total. Though it should be noted that the LCR strangely only predicted positive.The LCR again performs slightly better in the ROC graph as it gets closer to the upper left corner than the Knn line does.

The f1 scores of each model are as follows: LCR = 0.79844, Knn = 0.76008, Dummy = 0.572207.

I would not be able to recommend a model for classification between the choice of LCR and Knn as they both performed poorly and the difference in the two in this example is not significant enough to be able to say for certain which is better. This dataset is useless for classification and no classification model would be able to perform well.

4 Appendix

```
# id:19-19-19-0
# id:19--38-19-0
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, f1_score, roc_curve
from sklearn.model_selection import KFold, train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import PolynomialFeatures

##Function to do a-e for each dataset
def partsatoe(filename):

    if filename=="week4a.csv":
        optimalk= 5
        optimaldegree= 5
    else:
        optimalk= 9
        optimaldegree= 2

    #Part A
    ##Read in data
    df = pd.read_csv(filename)
    X1=df.iloc[:,0]
    X2=df.iloc[:,1]
    X=np.column_stack((X1,X2))
    y=df.iloc[:,2]

    ##Graph data
    plt.scatter(X1[y == 1], X2[y == 1], color='pink', marker='o')
    plt.scatter(X1[y == -1], X2[y == -1], color='paleturquoise', marker='o')

    plt.xlabel("X1")
    plt.ylabel("X2")
    plt.legend(["yReal","yPred"], loc='upper center',fancybox=True, shadow=True)
    plt.title("Raw Data")
    plt.show()

```

```
    ##Cross Validation for polynomial and C values
    plt.title("Logistic Regression Classifiers with Different Poly Degrees and Cis")

    polyRange = [1, 2, 5, 10]
    cRange = [0.01, 0.1, 1, 2]
    color = ['b', 'orange', 'g', 'r']
    counter = 0
    for i in polyRange:
        mean_f1=[]; std_f2=[]
        for c in cRange:
            model = LogisticRegression(C=c)
            temp=[]
            kf = KFold(n_splits=5)
            polyX = PolynomialFeatures(i).fit_transform(np.array(X))
            for train, test in kf.split(polyX):
                model.fit(polyX[train], y[train])
                ypred = model.predict(polyX[test])
                temp.append(f1_score(y[test], ypred))
            mean_f1.append(np.array(temp).mean())
            std_f2.append(np.array(temp).std())
        ##Graph
        markers, caps, bars = plt.errorbar(cRange, mean_f1, barsabove=True, yerr=std_f2,
    ecolor=color[counter], capsize=2, elinewidth=2, fmt="-o")
        [bar.set_alpha(0.4) for bar in bars]
        [cap.set_alpha(0.4) for cap in caps]
        counter = counter + 1

    plt.legend(["Degree: 1", "Degree: 2", "Degree: 5", "Degree: 10"], loc='lower right',
    fancybox=True, shadow=True)
    plt.xlabel("Ci")
    plt.ylabel("Mean F1")
    plt.show()

    ##Train optimal model
    polyX = PolynomialFeatures(optimaldegree).fit_transform(np.array(X))
    X_train, X_test, y_train, y_test = train_test_split(polyX, y)
    fig, (ax1, ax2) = plt.subplots(1, 2)

    logreg = LogisticRegression()
    logreg.fit(X_train, y_train)
    ypred = logreg.predict(X_test)

    ##Train dummy
    dclf = DummyClassifier(strategy = 'uniform')
    dclf.fit(X_train, y_train)
    ydummy = dclf.predict(X_test)

    ##Graph
    ax1.scatter(X1[y == 1], X2[y == 1], color='pink', marker='o')
    ax1.scatter(X1[y == -1], X2[y == -1], color='paleturquoise', marker='o')
    ax1.scatter(X_test[ypred == 1,1], X_test[ypred == 1,2], color='r', marker='x')
    ax1.scatter(X_test[ypred == -1,1], X_test[ypred == -1,2], color='b', marker='x')

    ax1.set(xlabel='X1', ylabel='X2')
    ax1.set_title("Logistic Regression Classifier")

    ax2.scatter(X1[y == 1], X2[y == 1], color='pink', marker='o')
    ax2.scatter(X1[y == -1], X2[y == -1], color='paleturquoise', marker='o')
    ax2.scatter(X_test[ydummy == 1,1], X_test[ydummy == 1,2], color='r', marker='x')
    ax2.scatter(X_test[ydummy == -1,1], X_test[ydummy == -1,2], color='b', marker='x')

    ax2.set(xlabel='X1', ylabel='X2')
    ax2.set_title("Dummy (Random) Classifier")

    if filename=="week4a.csv":
        fig.suptitle("Logistic and Dummy Classifiers: Degree=5, C=1")
    else:
        fig.suptitle("Logistic and Dummy Classifiers: Degree=2, C=1")
```

```
    fig.legend(["y = 1", "y = -1", "yPred = 1", "yPred = -1"], ncol=4, loc='lower center', fancybox=True, shadow=True)
    plt.show()

    print("Logistic Regression F1 Score: ", f1_score(y_test, ypred))
    print("Dummy (Random) F1 Score: ", f1_score(y_test, ydummy))

    #Part B
    kRange = [2, 5, 9, 14]
    mean_f1=[]; std_f2=[]
    for k in kRange:
        model = KNeighborsClassifier(n_neighbors=k)
        temp=[]
        kf = KFold(n_splits=5)
        for train, test in kf.split(X):
            model.fit(X[train], y[train])
            ypred = model.predict(X[test])
            temp.append(f1_score(y[test], ypred))
        mean_f1.append(np.array(temp).mean())
        std_f2.append(np.array(temp).std())

    ##Graph
    markers, caps, bars = plt.errorbar(kRange, mean_f1, barsabove=True, yerr=std_f2, capsize=2, elinewidth=2, fmt="-o")
    [bar.set_alpha(0.4) for bar in bars]
    [cap.set_alpha(0.4) for cap in caps]

    plt.legend(["F1 Score of Knn Models"], loc='lower right',fancybox=True, shadow=True)
    plt.title("F1 scores of Knn models with different k's")
    plt.xlabel("k")
    plt.ylabel("Mean F1")
    plt.show()

    ##Train optimal model
    X_train, X_test, y_train, y_test = train_test_split(X, y)
    fig, (ax1, ax2) = plt.subplots(1, 2)

    knn = KNeighborsClassifier(n_neighbors=optimalk)
    knn.fit(X_train, y_train)
    ypred = knn.predict(X_test)

    ##Train dummy
    dclf = DummyClassifier(strategy = 'uniform')
    dclf.fit(X_train, y_train)
    ydummy = dclf.predict(X_test)

    ##Graph
    ax1.scatter(X1[y == 1], X2[y == 1], color='pink', marker='o')
    ax1.scatter(X1[y == -1], X2[y == -1], color='paleturquoise', marker='o')
    ax1.scatter(X_test[ypred == 1,0], X_test[ypred == 1,1], color='r', marker='x')
    ax1.scatter(X_test[ypred == -1,0], X_test[ypred == -1,1], color='b', marker='x')

    ax1.set(xlabel='X1', ylabel='X2')
    ax1.set_title("Knn Classifier")

    ax2.scatter(X1[y == 1], X2[y == 1], color='pink', marker='o')
    ax2.scatter(X1[y == -1], X2[y == -1], color='paleturquoise', marker='o')
    ax2.scatter(X_test[ydummy == 1,0], X_test[ydummy == 1,1], color='r', marker='x')
    ax2.scatter(X_test[ydummy == -1,0], X_test[ydummy == -1,1], color='b', marker='x')

    ax2.set(xlabel='X1', ylabel='X2')
    ax2.set_title("Dummy (Random) Classifier")

    if filename=="week4a.csv":
        fig.suptitle("Knn and Dummy Classifiers: k=5")
    else:
         fig.suptitle("Knn and Dummy Classifiers: k=9")
    fig.legend(["y = 1", "y = -1", "yPred = 1", "yPred = -1"], ncol=4, loc='lower center', fancybox=True, shadow=True)
```

```
    plt.show()

    print("Knn F1 Score: ", f1_score(y_test, ypred))
    print("Dummy (Random) F1 Score: ", f1_score(y_test, ydummy))

    #Part C
    ##Using the models from the previous parts, test them all on the same y_test
    X_train, X_test, y_train, y_test = train_test_split(X, y)
    polyXtest = PolynomialFeatures(optimaldegree).fit_transform(np.array(X_test))
    polyXtrain = PolynomialFeatures(optimaldegree).fit_transform(np.array(X_train))

    logreg = LogisticRegression().fit(polyXtrain, y_train)
    knn = KNeighborsClassifier(n_neighbors=optimalk).fit(X_train, y_train)
    dclf = DummyClassifier(strategy='uniform').fit(X_train, y_train)

    predlog = logreg.predict(polyXtest)
    predknn = knn.predict(X_test)
    preddummy = dclf.predict(X_test)

    ##Print confusion matrices
    print("Confusion matrix for Logistic Regression Classifier")
    print(confusion_matrix(y_test, predlog))
    print("Confusion matrix for Knn Classifier")
    print(confusion_matrix(y_test, predknn))
    print("Confusion matrix for Dummy Classifier (random)")
    print(confusion_matrix(y_test, preddummy))

    #Part D

    ##Graph ROC curve for each model
    logreg = LogisticRegression().fit(polyXtrain, y_train)
    fpr, tpr, __ = roc_curve(y_test, logreg.predict_proba(polyXtest)[:, 1])
    plt.plot(fpr, tpr, color='orange', alpha=0.5, linewidth=1.8)

    knn = KNeighborsClassifier(n_neighbors=optimalk).fit(X_train, y_train)
    fpr, tpr, __ = roc_curve(y_test, knn.predict_proba(X_test)[:, 1])
    plt.plot(fpr, tpr, color='green', alpha=0.5, linewidth=1.8)

    dummy = DummyClassifier(strategy='uniform').fit(X_train, y_train)
    fpr, tpr, __ = roc_curve(y_test, dummy.predict_proba(X_test)[:, 1])
    plt.plot(fpr, tpr, color='blue', alpha=0.5, linewidth=1.8)

    plt.plot([0, 1], [0, 1], color='red', linestyle='dotted')

    ##Specify graph details
    plt.legend(['Logistic Regression', "Knn", "Dummy (Random)"], loc='lower right',fancybox=
    True, shadow=True)
    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.title("ROC Curves for 3 models")
    plt.show()

    #Part E
    ##Graph f1 scores of each model for comparison
    classifiers = ["Logistic Regression", "Knn", "Dummy (Random)"]
    f1scores = [f1_score(y_test, logreg.predict(polyXtest)), f1_score(y_test, knn.predict(
    X_test)), f1_score(y_test, dummy.predict(X_test))]
    bar_colors = ['tab:orange', 'tab:green', 'tab:blue']

    for i in range(3):
        print(classifiers[i], " - ", f1scores[i])

    plt.bar(classifiers, f1scores, color=bar_colors)
    plt.title("F1 Scores of 3 different Classifiers")
    plt.show()

partsatoe("week4a.csv")
partsatoe("week4b.csv")
```