Dataset 1 id: 19--38-19-0 Dataset 2 id: 19--19-19-0

Part 1 – Analysis for Dataset 1

1-a) Polynomial Order and C value for Logistic Regression

The following figure is a simple scatter plot of given training dataset. The -1 class is plotted in 'red' whereas +1 class is plotted in 'green' colour. This plot helps us understand the data is separable by a polynomial curve.

Figure 1-a1: Visualization of training dataset

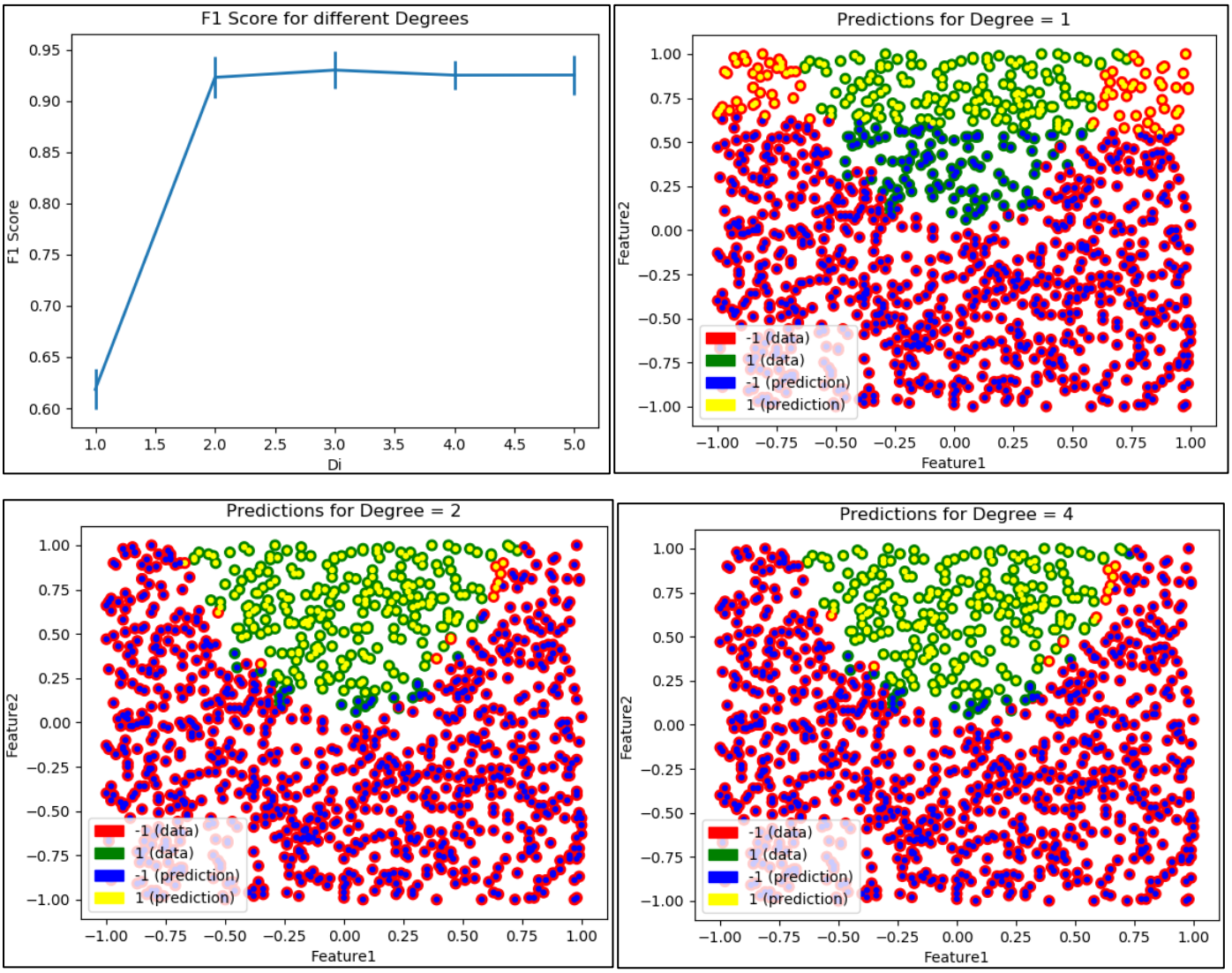
Finding Degree for Polynomial Features:

To find the degree for polynomial features function, Logistic Regression model has been trained with L2 penalty on input data set, augmented at various degrees from **1,2,3,4,5.** To capture the impact of different values of Di, the model has been trained using **5 – fold Cross Validation** and the **F1 score** of each Di is captured. **F1 score = 2TP / 2TP+FN+FP** this is helpful in measuring the effects of both false positives and false negatives in a single score. Also, this score will not approach to 1 for a dumb classifier that always predicts a single class. Higher F1 score is better.

```
for Di in D_Values:
    Xi=[]
    poly = PolynomialFeatures(Di)
    Xi = poly.fit_transform(X)

    model = linear_model.LogisticRegression(penalty='l2',solver='lbfgs')

    from sklearn.model_selection import KFold
    kf = KFold(n_splits=5)
    temp = []

    # Loop for all k-fold splits
    for train, test in kf.split(Xi):
        model.fit(Xi[train],y[train])
        ypred = model.predict(Xi[test])

        from sklearn.metrics import f1_score
        temp.append(f1_score(y[test],ypred))
```

The following figures are plots of F1 score vs different values of Degrees (Di) and plots of prediction vs training data. In the prediction plots training data is plotted in 'red' for -1 and 'green' for +1 class, whereas the predictions have been plotted in 'blue' for -1 and 'yellow' for +1 class.

Figure set 1-a2: F1 score plot and prediction plots for different values of D



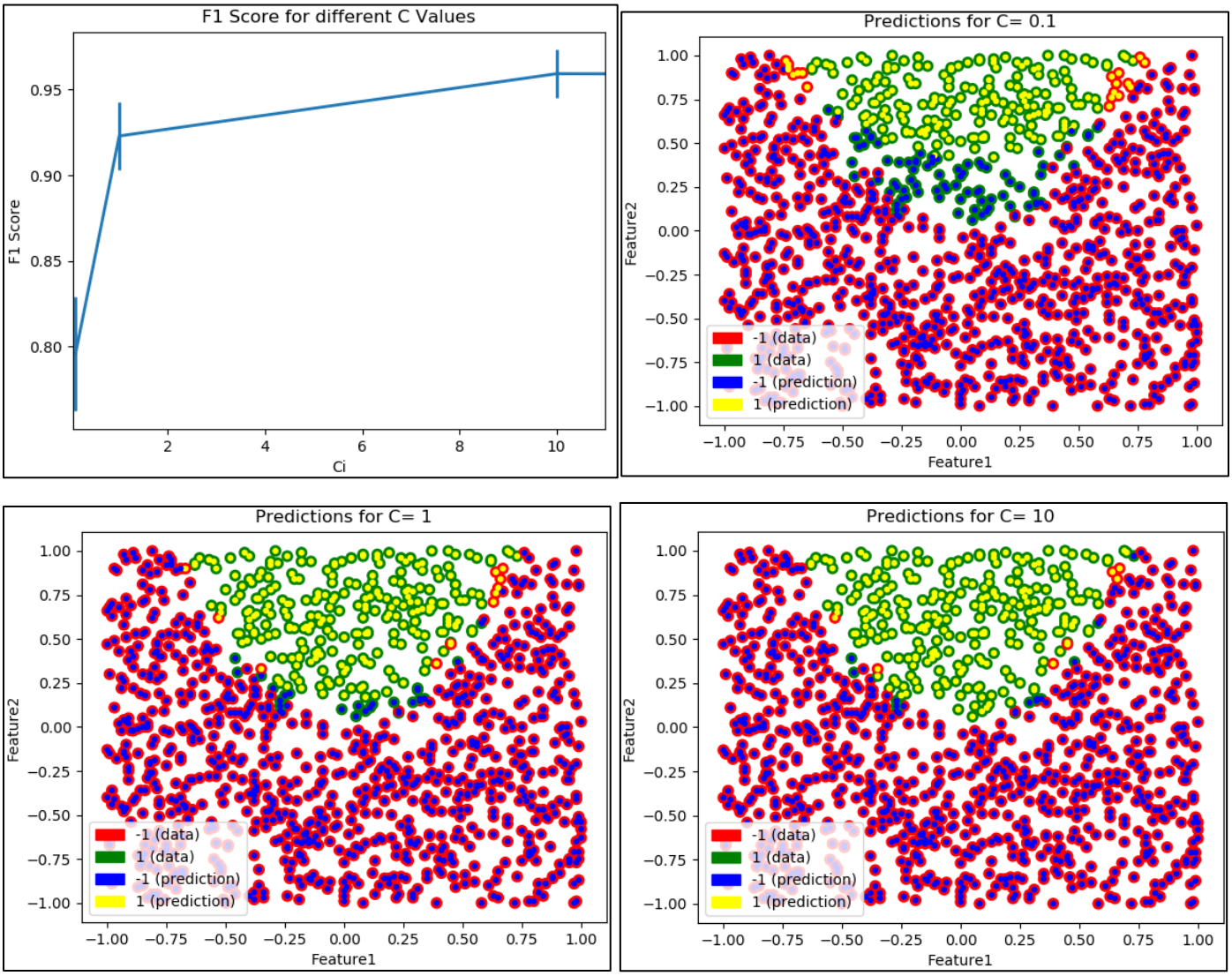
From the F1 score plot above we can see when augmented inputs of degree 2 are used the score jumps considerably when compared with degree 1. But for degree 3, 4, 5 the score remains almost same. This means degree 2 is best suited for the model. This can be confirmed by seeing at the prediction plots. When degree is 1, the model looks underfitted, but after moving to degree 2 the model is properly able to capture the curved separation of the data. Predictions for higher degrees does not seem to change much. Thus, we can finalize **Degree = 2** for this data set.

Finding value of penalty weight C:

C is the penalty weight attribute used to specify the strength of penalty used in the cost function. Small value of C results in a stronger penalty. Similar to degrees, we have used 5 – fold Cross Validation for evaluating Logistic Regression with different values of C (0.1, 1, 10, 100) by recording the F1 score for all these values.

The following figures are plots of F1 score vs different values of C and plots of prediction vs training data. In the prediction plots training data is plotted in 'red' for -1 and 'green' for +1 class, whereas the predictions have been plotted in 'blue' for -1 and 'yellow' for +1 class.

Figure set 1-a3: F1 score plot and prediction plots for different values of C (Penalty Weight)



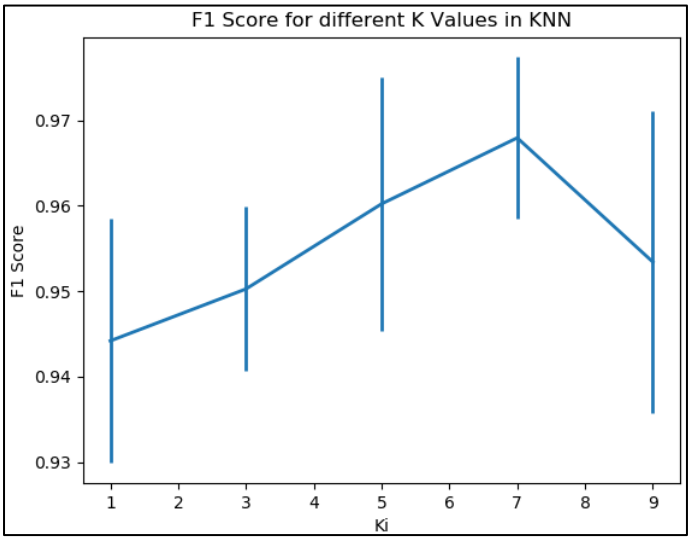
The F1 score plot for different values of C, suggests that when C=1 the F1 score is considerably higher when compared with C=0.1. Then the score increases very less and settles for higher values of C (i.e., 10 & 100). Also, the prediction plot shows signs of underfitting when C=0.1. The predictions are better when C=1, for C=10 the predictions are still good but there is a chance of overfitting. Considering all the factors above, **C=1** seems like a reasonable choice for the final value of C.

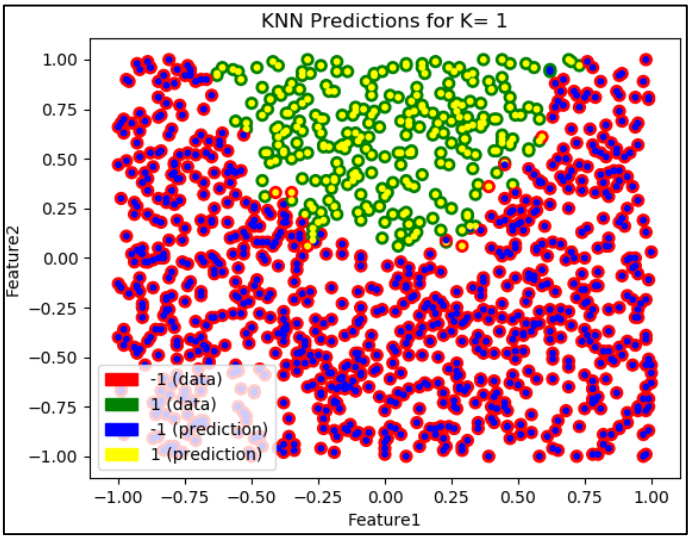
1-b) KNN Classifier

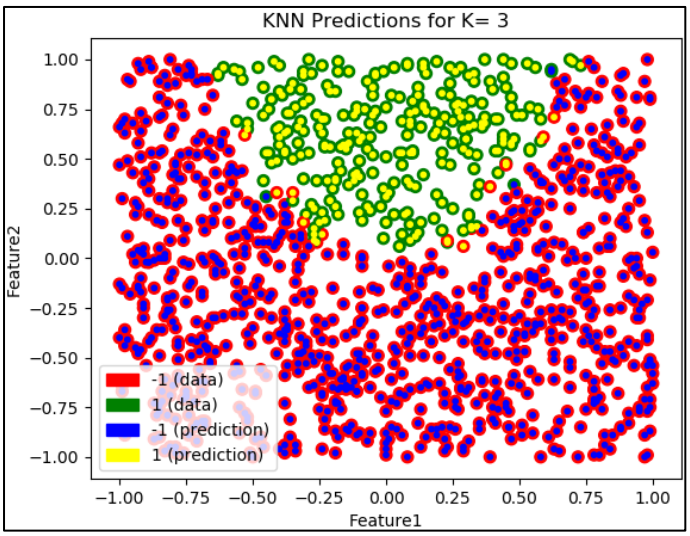
KNN model uses the class of K nearest neighbours to predict the class of the test data item. To determine the best value of K we have used 5-fold Cross Validation and recorded the F1 score for different values K (1, 3, 5, 7, 9).

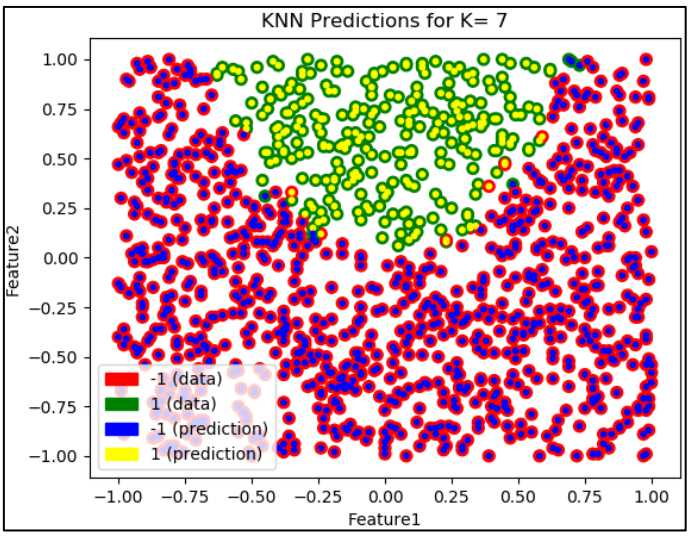
The figures below represent plot of F1 score against K values for the KNN classifier and plot of training data vs predicted data. The training data is plotted in 'red' for -1 and 'green' for +1 class, whereas the predictions have been plotted in 'blue' for -1 and 'yellow' for +1 class.

Figure set 1-b1: F1 score plot and prediction plots for different values of D









From the F1 score plot we can judge that the KNN model performs really well for all the values of K as the score is >0.94 for all of them. The prediction plot also shows good performance for all the values. We can choose **K=3,** as the F1 score is really good as well as we can avoid overfitting. This will also limit the number of distance calculations per prediction, when compared to larger values of K.

1-c) Confusion Matrices

We have trained Logistic Regression and KNN models above. Along with these we have created a baseline model that predicts the most common class of the given dataset. Consider +1 as the positive and -1 as the negative class.

Table 1-c1: Reference Confusion Matrix

	Predicted -1	**Predicted +1**
Actual -1	True Negative	False Positive
Actual +1	False Negative	True Positive

Table 1-c2: Logistic Regression Confusion Matrix

	Predicted -1	**Predicted +1**
Actual -1	176	2
Actual +1	4	47

Table 1-c3: KNN Confusion Matrix

	Predicted -1	**Predicted +1**
Actual -1	177	1
Actual +1	1	50

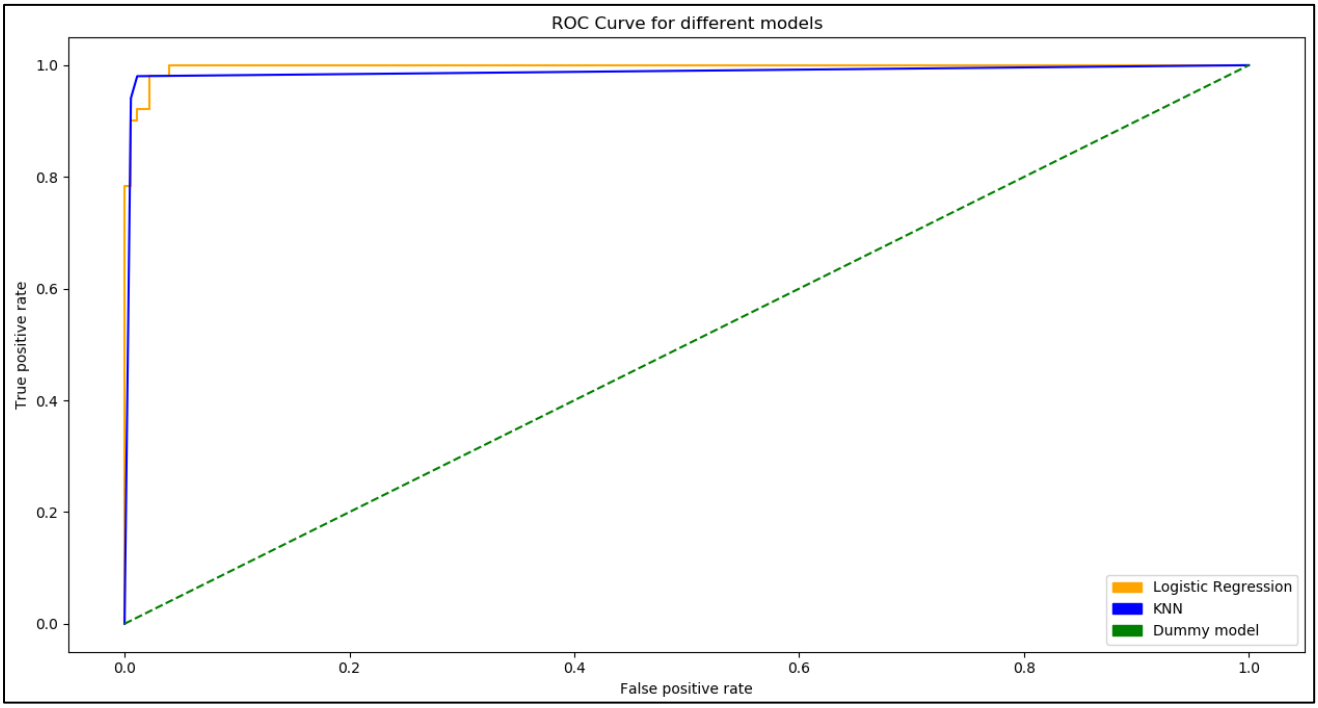
Table 1-c4: Dummy Classifier Confusion Matrix

	Predicted -1	**Predicted +1**
Actual -1	178	0
Actual +1	51	0

1-d) ROC Curve

An ROC curve is a plot of True Positive Rate on Y-axis vs False Positive Rate on X-axis. An ideal classification model should lean towards top left corner of the graph, i.e., it should have high TPR and low FPR. Following figure shows the ROC curve for the finalized Logistic Regression (orange), KNN (blue) as well as the dummy model (green).

Figure 1-d1: ROC curve plot for different models



1-e) Analysis using Confusion Matrix and ROC curve

In a confusion matrix the count of True Positive and True Negative should be high whereas the count of False Positive and False Negative Should be low. For Logistic Regression as well as for KNN the values of TP an TN are higher when compared to FP and FN, so we can say that these are good models. But for the dummy classifier predicts all points in -1 class. So, the TP count is 0 and FN count is very high, which is a sign of a bad model.

Also, the ROC curve shows the same thing, Logistic Regression and KNN are leaning towards top left, meaning they are good models where as dummy model is straight line diagonally which is not as sign of good model.

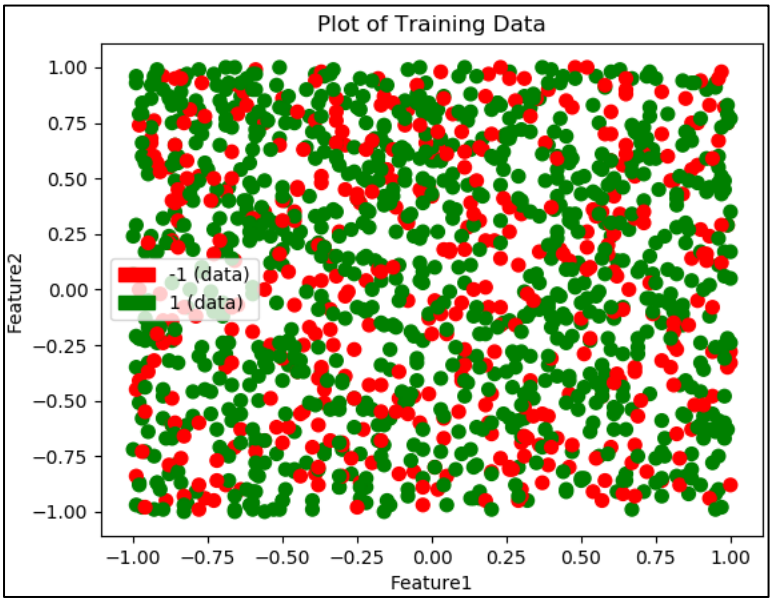
Based on these performance metrics above we can say that both, the Logistic Regression as well as the KNN provide really good classification, so either would be fine.

<u>Part 2 – Analysis for Dataset 2</u>

2-a) Polynomial Order and C value for Logistic Regression

To properly understand the data present in Training Dataset 2 we have used a simple scatter plot where -1 class is plotted in ‘red’ whereas +1 class is plotted in ‘green’ colour. After examining the plot, we can see that the data is very noisy and there doesn’t seem to be a line or a curve that can separate the two different classes in the dataset.

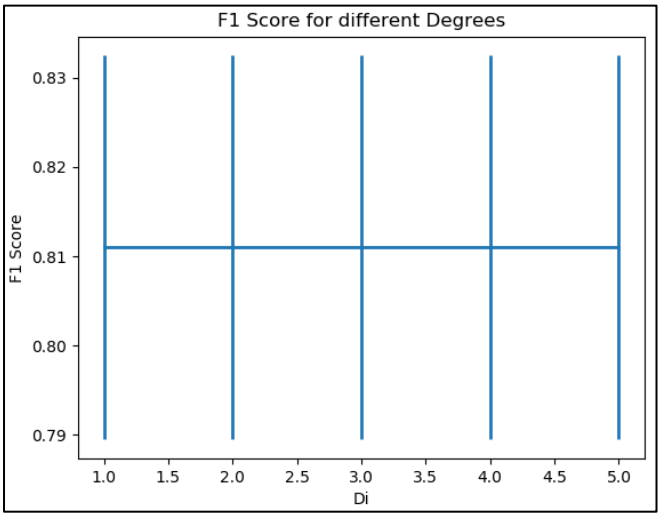
Figure 2-a1: Visualization of training dataset (Dataset 2)

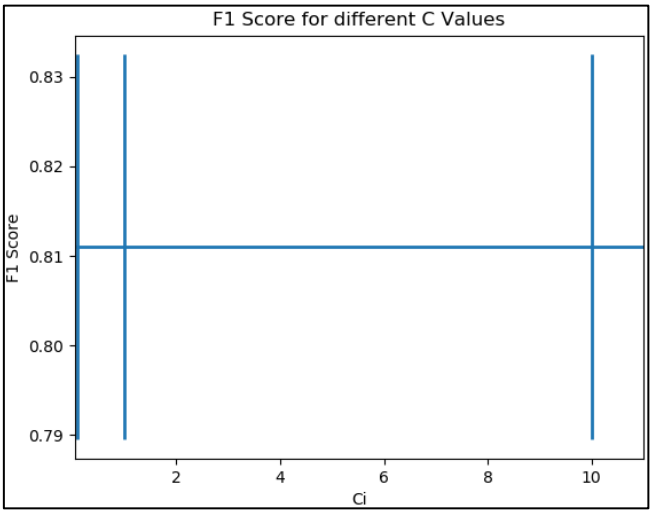


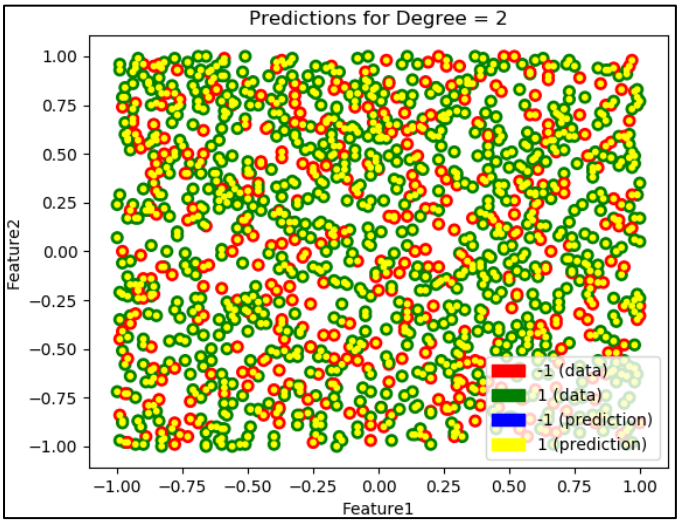
Degree of Polynomial Features and Penalty Weight C

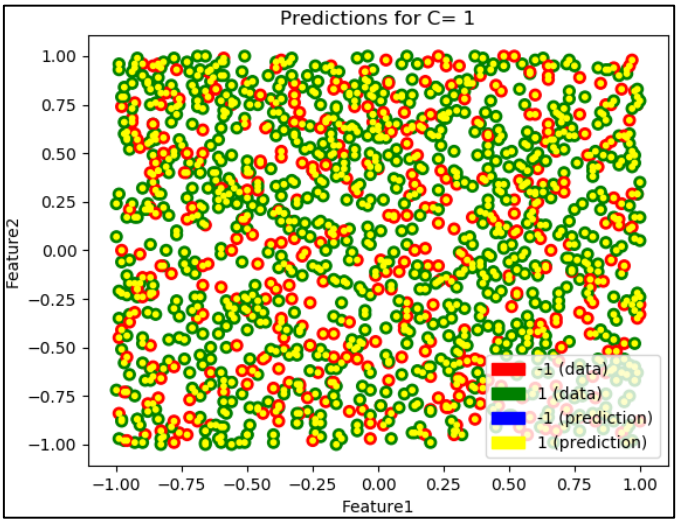
Similar to methods above, different values of Degree and C (Penalty Weight) were tried for the logistic regression, but the Logistic Regression model did not show any sign of good performance.

Figure set 2-a2: F1 Score plots and Prediction plots for different Degree and C value







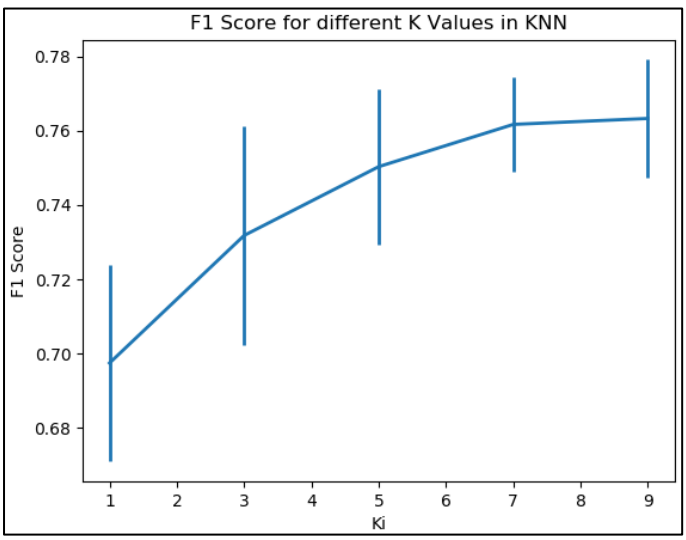


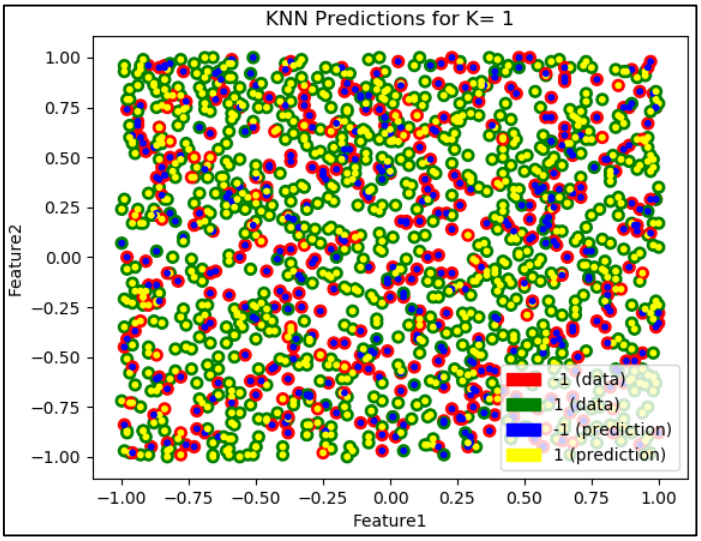
Based on the F1 score plots we can see that changing the Degree or changing the C value does not have any impact on the model. Also, the prediction plots shows that the model is predicting +1 class for almost all the data points, which means the Logistic Regression is as good as a dumb classifier that always predicts +1. Based on these observations logistic regression does not seem to be a good classifier for the given data set.

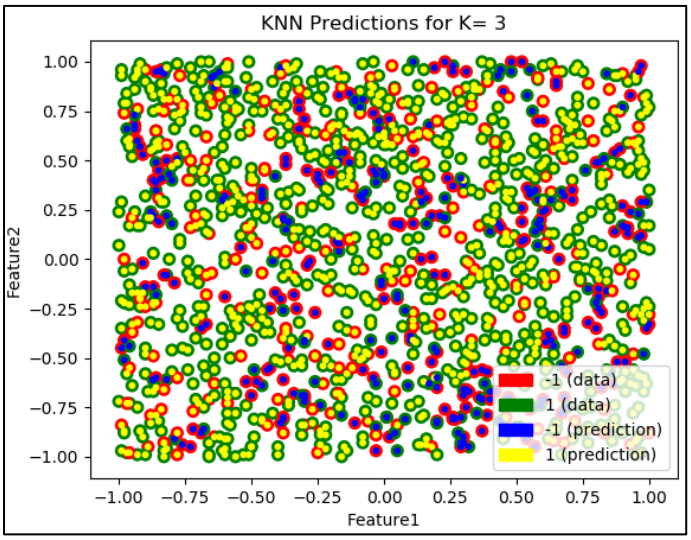
1-b) KNN Classifier

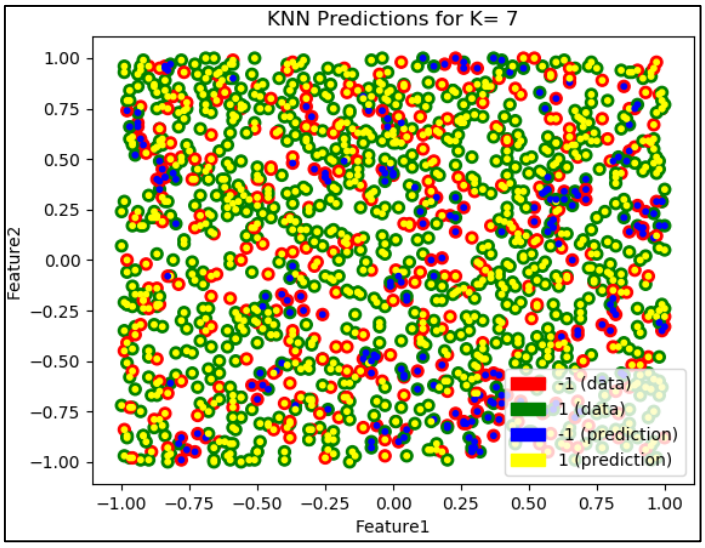
Similar to dataset 1, we try to find out performance of KNN for this dataset by varying the values of K.

Figure set 2-b2: F1 Score plot and Prediction plots for different K values (Dataset 2)









From the plots above we can see that F1 score increases slightly as we increase K value but is still in range of (0.7 to 0.76) which is not a very good score. The prediction plots also show that a KNN might be as good as dumb classifier that assigns a class randomly to the test data point. This shows that KNN also fails to find a good classification scheme for the given data set (Dataset 2).

2-c) Confusion Matrices

We have trained Logistic Regression and KNN models above. Along with these we have created a baseline model that predicts the most common class of the given dataset. In Dataset 2 the common class is +1. Consider +1 as the positive and -1 as the negative class.

Table 2-c1: Reference Confusion Matrix

	Predicted -1	**Predicted +1**
Actual -1	True Negative	False Positive
Actual +1	False Negative	True Positive

Table 2-c2: Logistic Regression Confusion Matrix

	Predicted -1	**Predicted +1**
Actual -1	0	91
Actual +1	0	182

Table 2-c3: KNN Confusion Matrix

	Predicted -1	**Predicted +1**
Actual -1	24	67
Actual +1	37	145

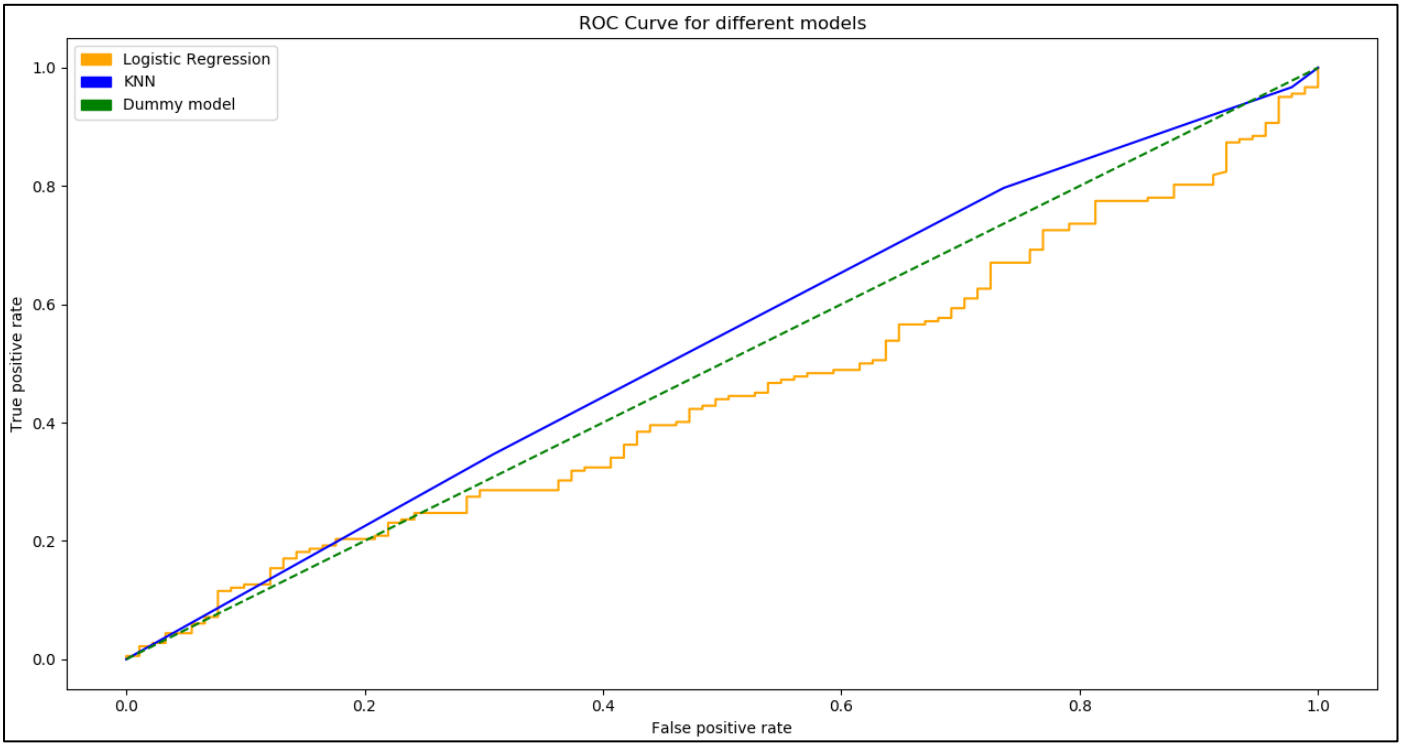
Table 2-c4: Dummy Classifier Confusion Matrix

	Predicted -1	**Predicted +1**
Actual -1	0	91
Actual +1	0	182

2-d) ROC Curve

Similar to dataset 1, we have created ROC plot for models trained on dataset 2 as well. Following figure shows the ROC curve for the finalized Logistic Regression (orange), KNN (blue) as well as the dummy model (green).

Figure 2-d1: ROC curve plot for different models



2-e) Analysis using Confusion Matrix and ROC curve

For the Logistic Regression model all prediction points belong to +1 class, which has increased the FP count whereas TN count is 0, which is not a good score. For the KNN model, the count of FN and FP is considerably high which shows the model is not able to capture the data relationships and patterns.

The ROC curve also suggests that both KNN and Logistic Regression are close to diagonal line of the dummy classifier. An ideal model should approach towards top left.

Based on the metrics above it is safe to say **NONE** of the models should be used.

Appendix

Code referred from lecture slides and official documentation of sklearn and matplotlib.

```
# Name: Omkar Pramod Padir
# Student Id: 20310203
# Dataset 1 id: 19--38-19-0
# Dataset 2 id: 19--19-19-0
# Course: Machine Learning CS7CS4
# Week 4 Assignment


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import matplotlib.patches as mpatches


# Part 1 starts here

# Load data and create arrays of input and output

# Dataset

DatasetArray=["W4_D1.csv","W4_D2.csv"]  # Same analysis for both datasets

for d in DatasetArray:

    # Load Data
    df = pd.read_csv(d)

    X1=df.iloc[:,0]
    X2=df.iloc[:,1]
    X=np.column_stack((X1,X2))
    y=df.iloc[:,2]

    # Colors for plots and legends
    cmap, norm = mcolors.from_levels_and_colors([-1, 0, 1], ['red', 'green'])
    red_patch = mpatches.Patch(color='red', label='-1 (data)')
    green_patch = mpatches.Patch(color='green', label='1 (data)')
    cmap2, norm2 = mcolors.from_levels_and_colors([-1, 0, 1], ['blue', 'yellow'])
    blue_patch = mpatches.Patch(color='blue', label='-1 (prediction)')
    yellow_patch = mpatches.Patch(color='yellow', label='1 (prediction)')
    bluecmap, norm3 = mcolors.from_levels_and_colors([-1, 0, 1], ['blue', 'blue'])
    yellowcmap, norm4 = mcolors.from_levels_and_colors([-1, 0, 1], ['yellow',
'yellow'])

    # function to plot given data
    def PlotBaselineData():
        plt.xlabel('Feature1')
        plt.ylabel('Feature2')

        plt.scatter(X1, X2, 50, y, cmap=cmap)

        plt.legend(handles=[red_patch, green_patch])


    plt.title("Plot of Training Data")
    PlotBaselineData()
    plt.show()

    # Function to plot predictions
```

```
    def PlotPredictionData(Prediction):

        # If all cases are +1 use yellow
        if (np.count_nonzero(Prediction==-1) == 0):
            plt.scatter(X1, X2, 10, Prediction, cmap=yellowcmap, marker="o")
        else:
            plt.scatter(X1, X2, 10, Prediction, cmap=cmap2, marker="o")


    from sklearn.preprocessing import PolynomialFeatures

    # this function will provide all combinations of inputs to degrree given in
parameters
    # eg. degree 3 for inputs a and b will give 1, a, b, a^2, a*b, b^2, a^3, (a^2)*b ,
a*(b^2), b^3.
    # https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html


    from sklearn import linear_model
    from sklearn.neighbors import KNeighborsClassifier

    D_Values = [1,2,3,4,5]            # Different Degree values
    C_Values = [0.1, 1, 10, 100]      # Different C values for changing penalty
strength
    K_Values = [1,3,5,7,9]            # Different K values for KNN

    mean_score=[]
    std_error=[]

    # Check performance of all Degrees
    for Di in D_Values:
        Xi=[]
        poly = PolynomialFeatures(Di)
        Xi = poly.fit_transform(X)

        model = linear_model.LogisticRegression(penalty='l2',solver='lbfgs')

        from sklearn.model_selection import KFold
        kf = KFold(n_splits=5)
        temp = []

        # Loop for all k-fold splits
        for train, test in kf.split(Xi):
            model.fit(Xi[train],y[train])
            ypred = model.predict(Xi[test])

            from sklearn.metrics import f1_score
            temp.append(f1_score(y[test],ypred))

        mean_score.append(np.array(temp).mean())
        std_error.append(np.array(temp).std())

        # Plot Baseline and Predictions
        PlotBaselineData()
        PlotPredictionData(model.predict(Xi))
        plt.legend(handles=[red_patch,green_patch,blue_patch,yellow_patch])
        plt.title("Predictions for Degree = "+str(Di))
        plt.show()

    # Plot F1 scores
    plt.errorbar(D_Values,mean_score,yerr=std_error,linewidth=2)
    plt.xlabel('Di'); plt.ylabel('F1 Score')
    plt.title("F1 Score for different Degrees")
```

```
    plt.show()

    # Check performance for different C values:
    mean_score=[]
    std_error=[]

    for Ci in C_Values:
        Xi=[]
        poly = PolynomialFeatures(2)    # 2 chosen from f1 score analysis above
        Xi = poly.fit_transform(X)

        model = linear_model.LogisticRegression(penalty='l2',C=Ci,solver='lbfgs')
        from sklearn.model_selection import KFold
        kf = KFold(n_splits=5)
        temp = []

        # Loop for all k-fold splits
        for train, test in kf.split(Xi):
            model.fit(Xi[train],y[train])
            ypred = model.predict(Xi[test])

            from sklearn.metrics import f1_score
            temp.append(f1_score(y[test],ypred))

        mean_score.append(np.array(temp).mean())
        std_error.append(np.array(temp).std())

        # Plot Baseline and Predictions
        PlotBaselineData()
        PlotPredictionData(model.predict(Xi))
        plt.legend(handles=[red_patch,green_patch,blue_patch,yellow_patch])
        plt.title("Predictions for C= "+str(Ci))
        plt.show()

    # Plot F1 score
    import matplotlib.pyplot as plt
    plt.errorbar(C_Values,mean_score,yerr=std_error,linewidth=2)
    plt.xlabel('Ci'); plt.ylabel('F1 Score')
    plt.xlim((0.05,11))
    plt.title("F1 Score for different C Values")
    plt.show()

    # Loop for different K values:
    mean_score=[]
    std_error=[]

    for Ki in K_Values:
        Xi=X

        model = KNeighborsClassifier(n_neighbors=Ki)

        from sklearn.model_selection import KFold
        kf = KFold(n_splits=5)
        temp = []

        # Loop for all k-fold splits
        for train, test in kf.split(Xi):
            model.fit(Xi[train],y[train])
            ypred = model.predict(Xi[test])

            from sklearn.metrics import f1_score
            temp.append(f1_score(y[test],ypred))

        mean_score.append(np.array(temp).mean())
```

```
        std_error.append(np.array(temp).std())

        PlotBaselineData()
        PlotPredictionData(model.predict(Xi))
        plt.legend(handles=[red_patch,green_patch,blue_patch,yellow_patch])
        plt.title("KNN Predictions for K= "+str(Ki))
        plt.show()



    plt.errorbar(K_Values,mean_score,yerr=std_error,linewidth=2)
    plt.xlabel('Ki'); plt.ylabel('F1 Score')
    plt.title("F1 Score for different K Values in KNN")
    plt.show()

    # Confusion Matrix of finalized models

    from sklearn.model_selection import train_test_split

    poly = PolynomialFeatures(2)    # 2 chosen from f1 score analysis above for
Logistic Regression
    Xi = poly.fit_transform(X)

    LR_X_train, LR_X_test, LR_y_train, LR_y_test = train_test_split(Xi, y,
test_size=0.2, random_state=42)
    KNN_X_train, KNN_X_test, KNN_y_train, KNN_y_test = train_test_split(X, y,
test_size=0.2, random_state=42)


    # Finalize LR model where C=1
    LR_Final =
linear_model.LogisticRegression(penalty='l2',C=1,solver='lbfgs').fit(LR_X_train,LR_y_tr
ain)
    # Finalize KNN model where K=2
    KNN_Final = KNeighborsClassifier(n_neighbors=3).fit(KNN_X_train,KNN_y_train)


    from sklearn.dummy import DummyClassifier
    dummy_clf = DummyClassifier(strategy="most_frequent")    # Dummy classifier that
predicts most frequent class
    dummy_clf.fit(KNN_X_train, KNN_y_train)

    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import roc_curve

    print("Confusion Matrix for Final LR Model")
    print(confusion_matrix(LR_y_test, LR_Final.predict(LR_X_test)))

    print("Confusion Matrix for Final KNN Model")
    print(confusion_matrix(KNN_y_test, KNN_Final.predict(KNN_X_test)))

    print("Confusion Matrix for Dummy Model")
    # Using KNN split because it represents input without augmented features i.e.
Polyfeatures() NOT applied
    print(confusion_matrix(KNN_y_test, dummy_clf.predict(KNN_X_test)))


    # Roc Curve for all finalized models

    LR_fpr, LR_tpr, _ = roc_curve(LR_y_test,LR_Final.decision_function(LR_X_test))
    plt.plot(LR_fpr,LR_tpr, color= 'orange')

    KNN_fpr, KNN_tpr, _ =
roc_curve(KNN_y_test,KNN_Final.predict_proba(KNN_X_test)[:,1])
```

```
    plt.plot(KNN_fpr,KNN_tpr, color='blue')

    D_fpr, D_tpr, _ = roc_curve(KNN_y_test,dummy_clf.predict_proba(KNN_X_test)[:,1])
    plt.plot(D_fpr,D_tpr , color='green',linestyle='dashed')

    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    orange_roc = mpatches.Patch(color='orange', label='Logistic Regression')
    blue_roc = mpatches.Patch(color='blue', label='KNN')
    green_roc = mpatches.Patch(color='green', label='Dummy model')
    plt.legend(handles=[orange_roc,blue_roc,green_roc])
    plt.title('ROC Curve for different models')
    plt.show()
```