# Question 1 - Data set week4_1.csv

## a

The plot presents the data in `week4_1.csv`. There are two features `x_1` and `x_2`, and the target values only having two values: +1 and -1. The green plus signs represent the +1 data and blue dot markers represent the -1 data. Which indicates that the data requires a classification model. And it could be observed that a clear curve boundary is between the two types of the target values, so the polynomial degree might be quadratic.

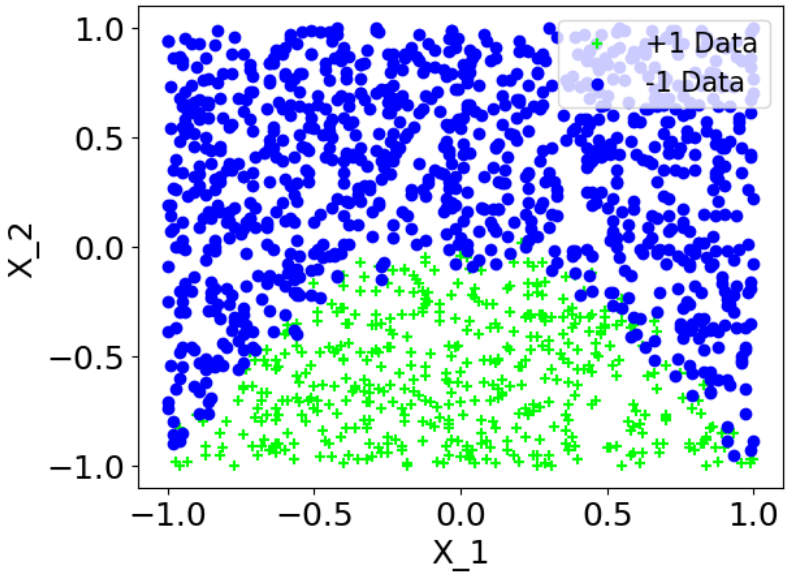



Figure 1: Plot of Data Set week4_1.csv

To find out the parameters of the most fit logistic regression model, the `f1` metrics is applied to evaluate the performance classification models, because it indicates the the model's ability at handling both types of errors: false positives and true negatives. With continuously calculating the mean and standard deviation of `f1` scores through `cross_val_scores` function, we are able to draw an `errorbar` to analyze the performance.

Firstly, evaluate the performance of the logistic model for values of polynomial degree from 1 to 15. The results are presented in the figure 2, where the x-axis is for the polynomial degree and the y-axis is for the `f1` score. The `f1` reaches its maximum(0.9442) at `degree = 2` and stays steady after that. In this case, the value 2 is the best polynomial degree for this data set, because there is no significant raise after 2 which means overfit would happen for any value greater than 2. That shows that the relationship between the features and target value is

quadratic.

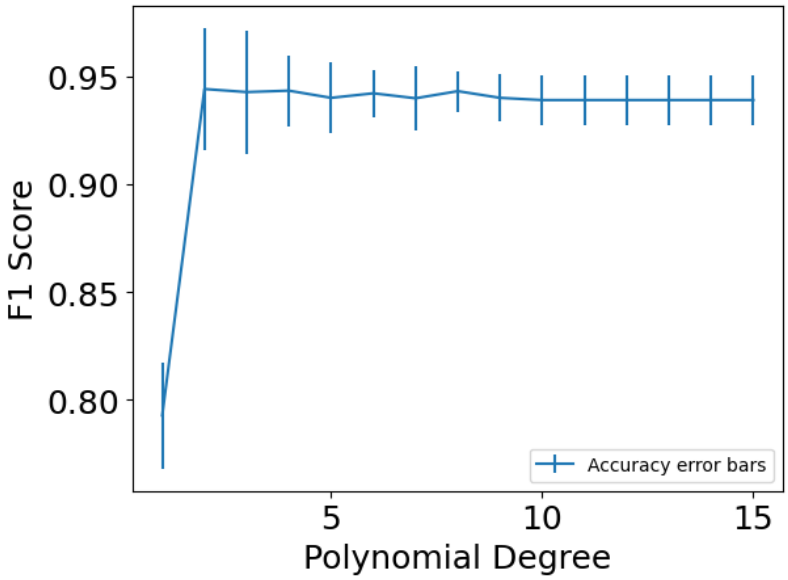



Figure 2: Plot of f1 vs different polynomial degrees on logistic regression model

Secondly, to decide the value of `C`, the same methodology is applied, which is iterating a range of `C` values while evaluating and presenting their performance calculated from `cross_val_score`. The following plots are also an `errorbar` but with x-axis changing to `C`. For the left plot, the range of `C` being tested is from 1 to $2^{11}$, and we can find that that highest `f1` score is somewhere below `c = 100`. So we shrink the range to find the best `C`. With the right plot, it can be seen that the highest `f1` score(0.9506) is found at `C = 57.36`, and that is the best value of `C`.

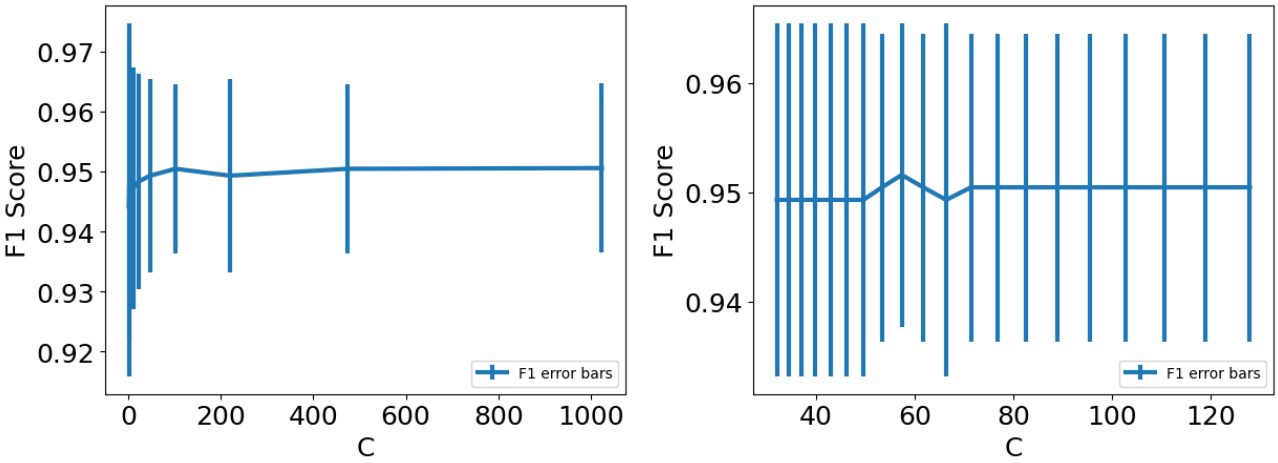



Figure 3: Error bar of Logistic Regression Model

Finally, to plot the model predictions, the following steps are taken: setting the polynomial degree to `2` to generate additional features, splitting the data into training and testing sets by `train_test_split`, instantiating a logistic regression model with `c = 57.36` and training the model on the training data, then predicting on the testing set. Now we generate the plot in Figure 5. There are upward red triangles representing +1 predictions and downward orange triangles representing -1 predictions. The predictions are quite accurate where all the red

upward triangles(+1 prediction) are located inside green plus signs(+1 data), and all the -1 predictions are inside -1 data.

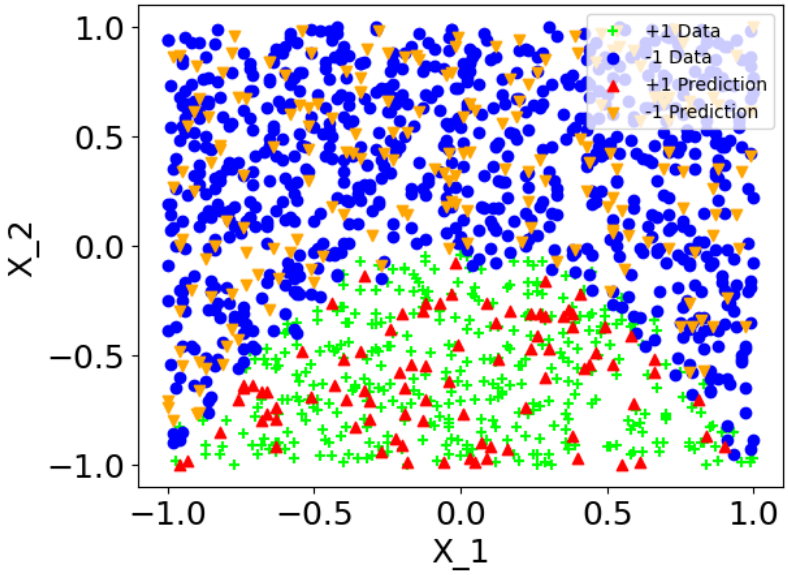
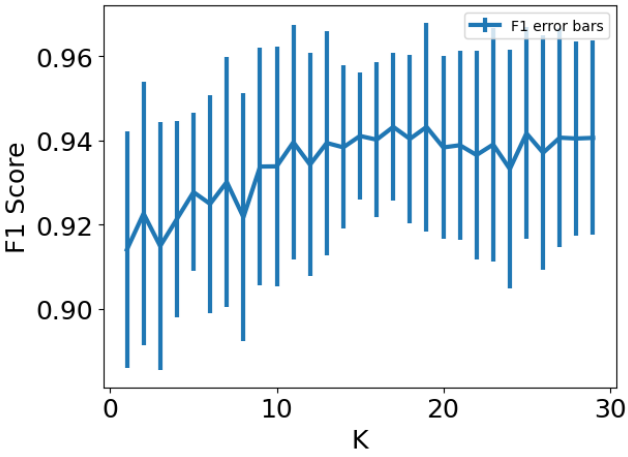



Figure 4: Plot of Original Data and Logistic Regression Model's Predictions

With the `K` set to `17`, the predictions by KNN model are shown in the following plot, all the red upward triangles(+1 prediction) locate inside the region of green plus signs(+1 data), and all the -1 predictions are in the region of -1 data.



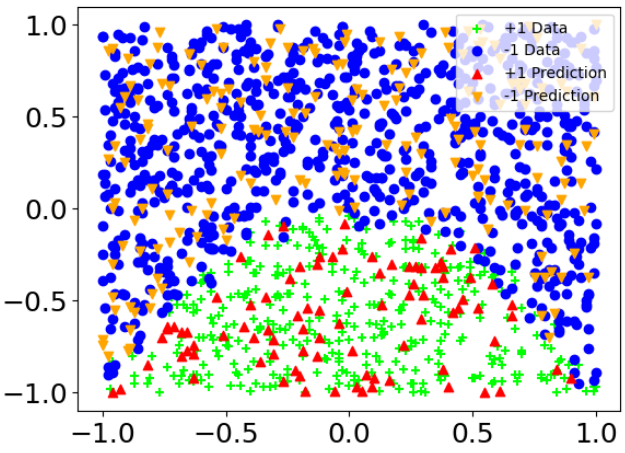




Figure 5: Plot of error bar of KNN model across a range of K

Figure 6: Plot of Original Data and kNN's Predictions

## c

To deeply analyze the performance of each model, two dummy classifier are introduced as the baselines for comparing the performance. One is trained by `most_frequent` strategy to always output the most frequent class, the other one is trained by `uniform` strategy to randomly predict results.

The confusion matrices are retrieved to form heatmaps to visualize the prediction accuracy. There are 4 heatmaps, in which two for dummy classifiers, one for the logistic regression and one for KNN. For each heatmap, the true positive is at the top-left, and true negative is at the top-right, false positive is at the bottom-left and false negative is at the bottom-right.

From the data of most frequent dummy classifier, we can find that there is 195 true data and 88 false data. Based on that, the logistic regression model predicts precisely with 191 TP and 84 FN, slightly higher than the kNN model having 189 TP and 83 FN. And the performances of them are much better than the random dummy classifier which only having around half of the predictions correct.

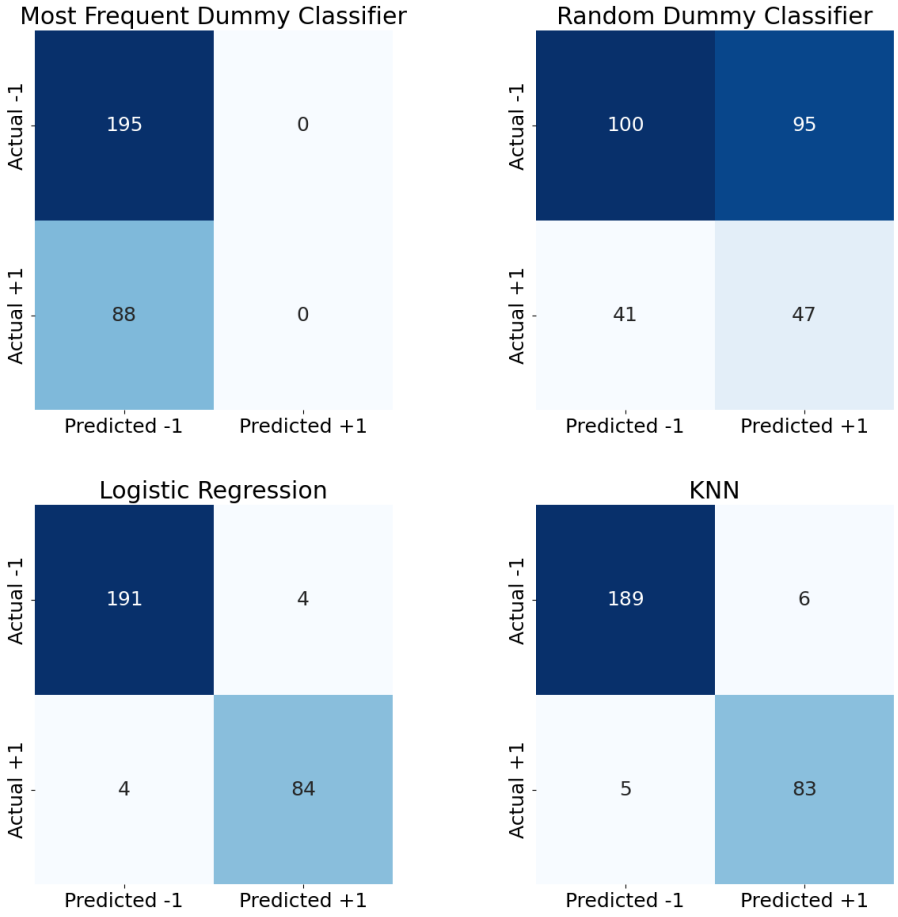



Figure 7: Heatmaps of Confusion Matrices for Models

# d

The following plot display 4 ROC curves corresponding to 4 different models from question c. ROC curve indicates the performance of a binary classifier with different decision thresholds. At the left side of plot we can see that the true positive rates of both logistic model and kNN model are above 0.8 when the false positive rate is zero. As the false positive rate raises to nearly 20%, both models can predict 100% true positive values, with a bit difference that logistic model has slightly higher TPR when FPR is close to zero. Which mean the two models are both

capable to predict true positive predictions while with very low chance of making false positive predictions. So that we can say that the two trained models does fairly well.

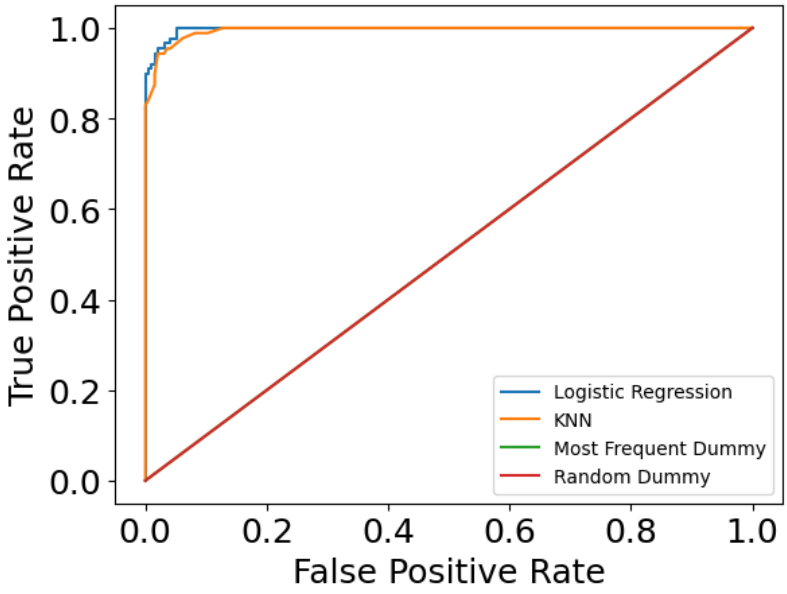



Figure 8: Plot of ROC on multiple models

# Question 2 - Data set week4_2.csv

## a

The plot from Figure 9 shows the data from data set week4_2.csv. There are two features involved, and two types of target values existed. The green plus sign is for the +1 data and the blue dot maker is for -1 data. The data of both two classes is quite discrete and no decision boundary could be found, so there might not be able to find a model that can fit the data well.

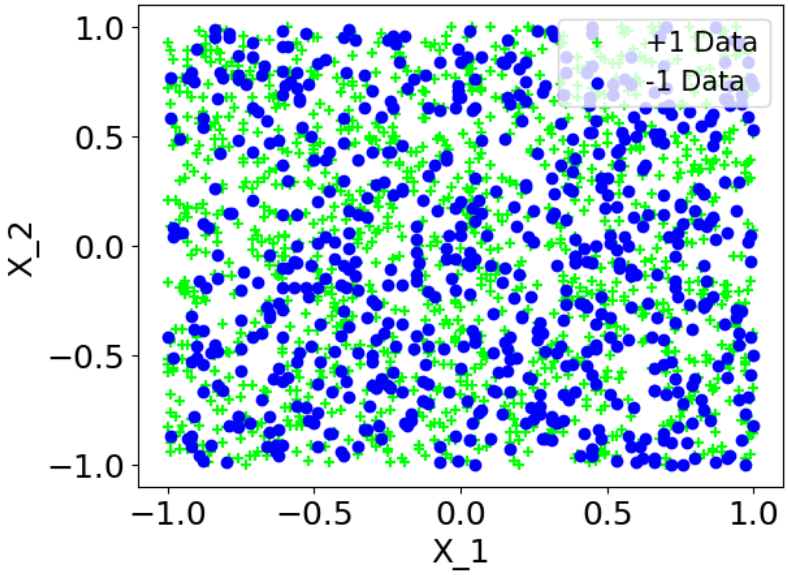



Figure 9: Plot of week4_2.csv data

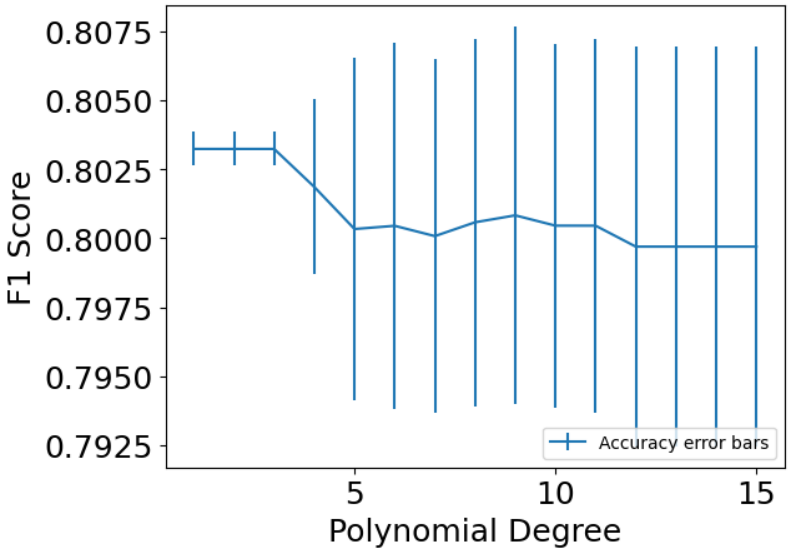



Figure 10: Plot of F1 scores of logistic Regression Model vs Polynomial Degree

The following plot is used for finding the `C` value for the logistic model in range from $2^{-5}$ to $2^{5}$, the `f1` scores barely change, and the maximum value is found at `C = 0.0312`. By which we conclude that the most suitable value of `C` is 0.0312.

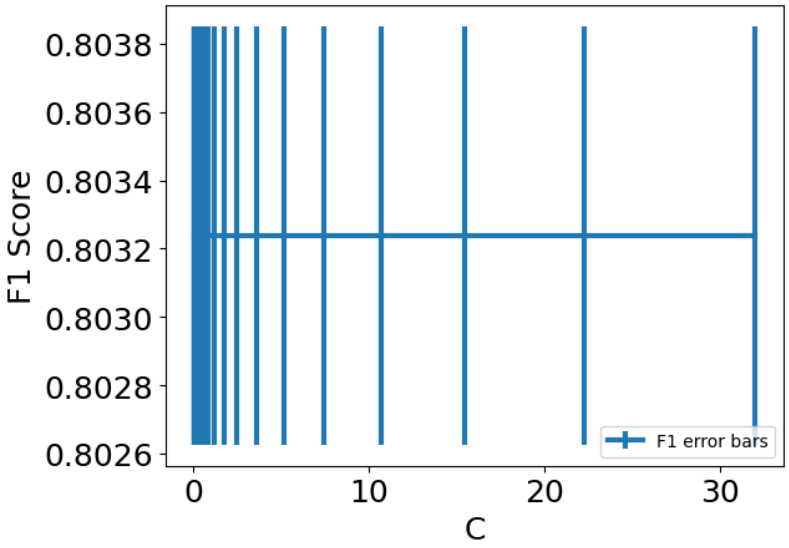



Figure 11: Plot of F1 scores of logistic Regression Model vs C

After calculating the most appropriate values of polynomial degree and C, we are able to train the logistic model and make predictions. The following plot shows the predictions of the trained model. The +1 predictions are discrete all over the plot, almost no -1 predictions could be found, and there is no clear decision boundary. This model does not fit the data well.

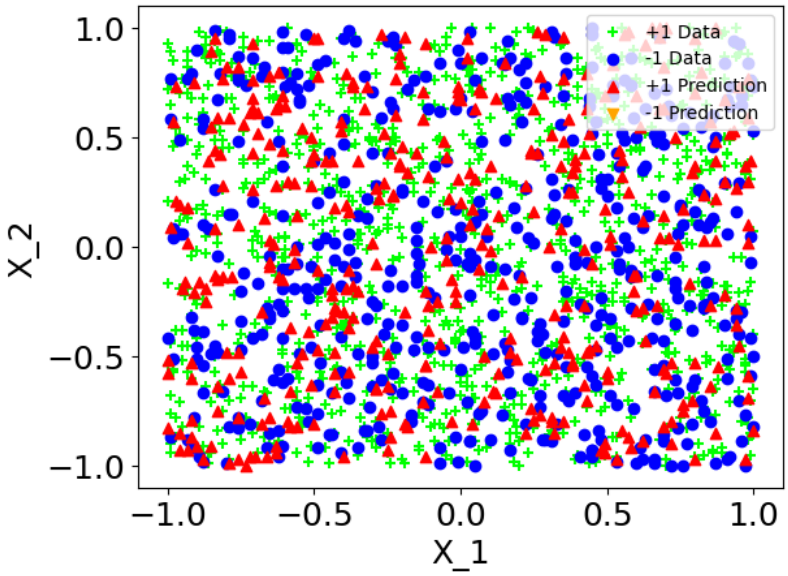
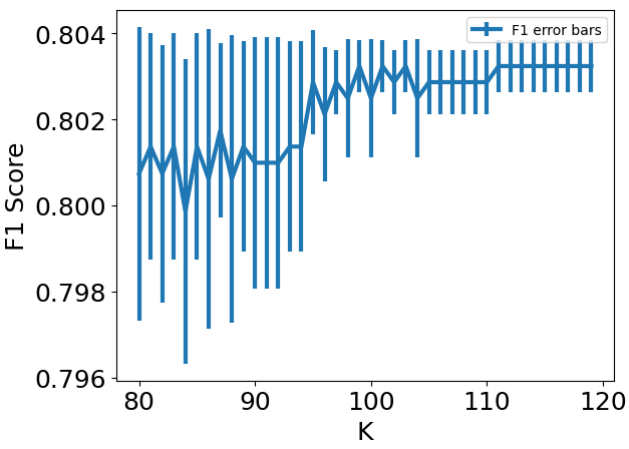



Figure 12: Plot of Original Data and Logistic Model Predictions

To find the correct value of k, evaluating the kNN models with different k values from 80 to 120. From the following error bar, it can be found that the maximum f1 score is at `K = 99`. Which means the kNN model would have the best predictions when each prediction is effected by the 99 nearest neighbors.

Plotting the predictions from the kNN model on the testing data and the original training data. It can be seen that the same phenomenon happens for kNN model, +1 predictions are scattered all over the graph while no -1 predictions could be seen, and also no clear decision boundary. Which means the most fitting kNN model could not capture the data set well.



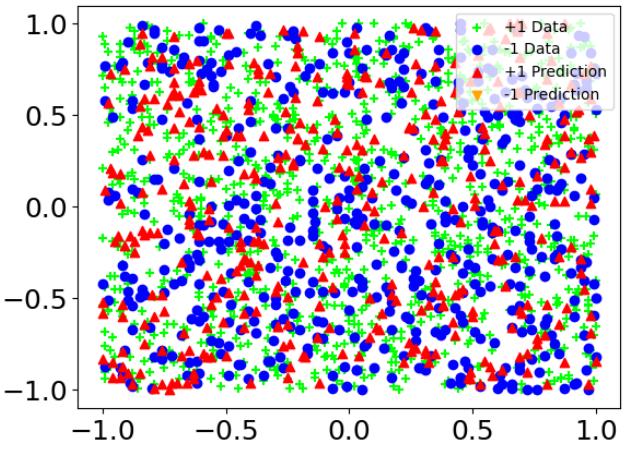




Figure 13: Plot of kNN model with multiple K values

Figure 14: Plot of training data and kNN predictions

From the following heatmaps, it can be seen that logistic model and the kNN model predicts exactly same as the most frequent dummy classifier. It indicates that the two models could not capture the feature of +1 data, they perform terribly as a dummy classifier.

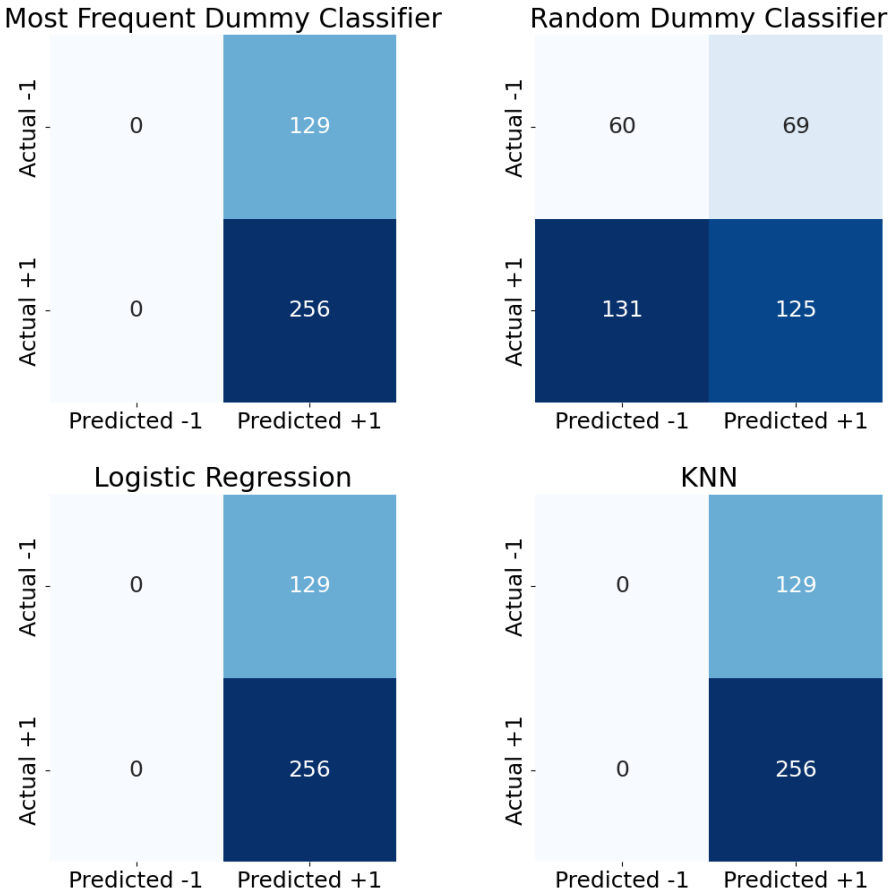



Figure 15: Heatmaps of 4 models on data set

Figure 15: Plot of ROC of different models

## e

Based on the results from question c and d, we can safely conclude that the two well refined models could not fit the data set well because this data set's relationship is hard to learn and capture, the logistic and kNN model's performance is the same as randomly guessing or always predicting the most frequent one. So no recommendation would be offered for the classifier.

# Appendix

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

from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score
```

```
from sklearn.dummy import DummyClassifier

# %%
def load_data(path):
    df = pd.read_csv(path,sep=',', comment='#', header=None)
    x1=df.iloc[:,0]
    x2=df.iloc[:,1]
    x=np.column_stack((x1,x2))
    y=df.iloc[:,2]
    return x, y

# (a) determine best poly degree and c value of penalty of Logistic regression
def FindBestPolyDegree(x, y, poly_range):
    poly = PolynomialFeatures()
    best_degree = 1
    best_penalty = 0
    best_acuracy = 0
    accuracy_mean_error=[]
    accuracy_std_error=[]
    for p in poly_range:
        # polynomial transformation
        poly.set_params(degree=p)
        x_poly = poly.fit_transform(x)


        model = LogisticRegression("l2")

        # cross validation with 5 folds
        scores = cross_val_score(model, x_poly, y, cv = 5, scoring='f1')
        mean = np.mean(scores)
        accuracy_mean_error.append(mean)
        accuracy_std_error.append(np.std(scores))

        if(mean > best_acuracy):
            best_acuracy = np.mean(scores)
            best_degree = p

    plt.errorbar(poly_range, accuracy_mean_error, yerr=accuracy_std_error,
label='Accuracy error bars')

    print(f"Maximum accuracy: {best_acuracy:.4f} at degree {best_degree}")
    plt.xlabel('Polynomial Degree')
    plt.ylabel('F1 Score')
    plt.legend(loc='lower right', fontsize = 10)
    plt.show()
    return best_degree
```

```
def FindBestC(x, y, best_degree, c_range):

    poly = PolynomialFeatures(best_degree)
    x_poly = poly.fit_transform(x)

    accuracy_mean_error=[]
    accuracy_std_error=[]
    best_penalty = 0
    best_acuracy = 0
    for c in c_range:
        model = LogisticRegression("l2", C=c)

        # cross validation with 5 folds
        scores = cross_val_score(model, x_poly, y, cv = 5, scoring='f1')
        mean = np.mean(scores)
        accuracy_mean_error.append(mean)
        accuracy_std_error.append(np.std(scores))

        if(mean > best_acuracy):
            best_acuracy = np.mean(scores)
            best_penalty = c

    plt.errorbar(c_range, accuracy_mean_error, yerr=accuracy_std_error,
linewidth=3, label='F1 error bars')

    print(f"Maximum accuracy: {best_acuracy:.4f} at degree {best_degree} with
C {best_penalty:.4f}")
    plt.xlabel('C')
    plt.ylabel('F1 Score')
    plt.legend(loc='lower right', fontsize = 10)
    plt.show()
    return best_penalty

# draw the best model predictions
def DrawBestLogisticPredictions(x, y, best_degree, best_penalty):
    poly = PolynomialFeatures()
    poly.set_params(degree=best_degree)
    x_poly = poly.fit_transform(x)

    x_poly_train, x_poly_test, y_poly_train, y_poly_test =
train_test_split(x_poly, y, test_size=0.2, random_state=1)

    plt.scatter(x_poly_train[y_poly_train == 1][:, 1],
x_poly_train[y_poly_train == 1][:, 2],  color='#00ff00', marker='+', label='+1
Data')
```

```
    plt.scatter(x_poly_train[y_poly_train == -1][:, 1],
x_poly_train[y_poly_train == -1][:, 2], color='#0000fd', marker='o', label='-1
Data')

    best_logistic = LogisticRegression("l2", C=best_penalty)
    best_logistic.fit(x_poly_train, y_poly_train)
    y_best_logistic_pred = best_logistic.predict(x_poly_test)


    plt.scatter(x_poly_test[y_best_logistic_pred == 1][:, 1],
x_poly_test[y_best_logistic_pred == 1][:, 2],  color="red", marker='^',
label='+1 Prediction')
    plt.scatter(x_poly_test[y_best_logistic_pred == -1][:, 1],
x_poly_test[y_best_logistic_pred == -1][:, 2], color="orange", marker='v',
label='-1 Prediction')

    # print(f"F1: {best_logistic.(x_poly_test, y_poly_test):.4f}")

    plt.rc('font', size=18)
    plt.xlabel('X_1'); plt.ylabel('X_2')
    plt.legend(loc='upper right', fontsize = 10)
    plt.show()
    return best_logistic, y_best_logistic_pred, x_poly_test, y_poly_test

# (b) determine best k value of KNN
def FindBestKForKNN(x, y, k_range):
    best_k = 1
    best_acuracy = 0
    accuracy_mean_error=[]
    accuracy_std_error=[]
    for k in k_range:
        model = KNeighborsClassifier(n_neighbors=k)
        scores = cross_val_score(model, x, y, cv = 5, scoring='f1')
        mean = np.mean(scores)
        if(mean > best_acuracy):
            best_acuracy = mean
            best_k = k
        accuracy_mean_error.append(mean)
        accuracy_std_error.append(np.std(scores))

    plt.errorbar(k_range, accuracy_mean_error, yerr=accuracy_std_error,
linewidth=3, label='F1 error bars')
    plt.xlabel('K')
    plt.ylabel('F1 Score')
    plt.legend(loc='best', fontsize = 10)
    plt.show()
```

```
    print(f"Maximum F1: {best_acuracy:.4f} at k={best_k}")
    return best_k

def DrawBestKNNPrediction(x, y, best_k):
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=1)

    plt.scatter(x_train[y_train == 1][:, 0],  x_train[y_train == 1][:, 1],
color='#00ff00', marker='+', label='+1 Data')
    plt.scatter(x_train[y_train == -1][:, 0], x_train[y_train == -1][:, 1],
color='#0000fd', marker='o', label='-1 Data')

    best_kNN = KNeighborsClassifier(n_neighbors=best_k)
    best_kNN.fit(x_train, y_train)
    y_best_kNN_pred = best_kNN.predict(x_test)

    plt.scatter(x_test[y_best_kNN_pred == 1][:, 0],  x_test[y_best_kNN_pred ==
1][:, 1],  color="red", marker='^', label='+1 Prediction')
    plt.scatter(x_test[y_best_kNN_pred == -1][:, 0], x_test[y_best_kNN_pred ==
-1][:, 1], color="orange", marker='v', label='-1 Prediction')
    plt.legend(loc='upper right', fontsize = 10)
    plt.show()
    print(f"F1: {best_kNN.score(x_test, y_test):.4f}")
    return best_kNN, y_best_kNN_pred

# (c) confusion matrices
import seaborn as sns
def ConfutionMatrix(x, y, y_best_logistic_pred, y_best_kNN_pred):
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=1)

    fig, axes = plt.subplots(2, 2, figsize=(12, 12))
    def plot_confusion_matrix(x_loc, y_loc, y_test, y_pred, title):
        conf_matrix = confusion_matrix(y_test, y_pred)
        sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
cbar=False,
                    xticklabels=["Predicted -1", "Predicted +1"],
                    yticklabels=["Actual -1", "Actual +1"], ax=axes[x_loc,
y_loc])
        axes[x_loc, y_loc].set_yticklabels(axes[x_loc,
y_loc].get_yticklabels(), rotation=90)
        axes[x_loc, y_loc].set_title(title)

        tn, fp, fn, tp = conf_matrix.ravel()

        # Calculate recall (sensitivity)
```

```
        recall = tp / (tp + fn)

        # Calculate specificity
        specificity = tn / (tn + fp)

        # Calculate precision
        precision = tp / (tp + fp)

        # Calculate F1 score
        f1_score = 2 * (precision * recall) / (precision + recall)

        # Display the results
        print(f"Recall (Sensitivity): {recall:.2f}")
        print(f"Specificity: {specificity:.2f}")
        print(f"F1 Score: {f1_score:.2f}")


    print("Most Frequent Dummy Classifier Confution Matrix")
    most_frequent_dummy =
DummyClassifier(strategy='most_frequent').fit(x_train, y_train)
    ydummy = most_frequent_dummy.predict(x_test)
    plot_confusion_matrix(0, 0, y_test, ydummy, "Most Frequent Dummy
Classifier")

    print("Random Dummy Classifier Confution Matrix")
    random_dummy = DummyClassifier(strategy='uniform',
random_state=42).fit(x_train, y_train)
    ydummy = random_dummy.predict(x_test)
    plot_confusion_matrix(0, 1, y_test, ydummy, "Random Dummy Classifier")


    print("Logistic Regression Confution Matrix")
    plot_confusion_matrix(1, 0, y_test, y_best_logistic_pred, "Logistic
Regression")

    print("KNN Confution Matrix")
    plot_confusion_matrix(1, 1, y_test, y_best_kNN_pred, "KNN")

    plt.subplots_adjust(wspace=0.4, hspace=0.25)  # Increase space between
plots
    plt.show()

    return most_frequent_dummy, random_dummy

# (d) ROC curve
def DrawROCCurve(x, y, x_poly_test, y_poly_test, best_logistic, best_kNN,
```

```
most_frequent_dummy, random_dummy):
    fpr, tpr, thresholds = roc_curve(y_poly_test,
best_logistic.decision_function(x_poly_test))
    plt.plot(fpr, tpr, label='Logistic Regression')
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=1)
    fpr, tpr, thresholds = roc_curve(y_test, best_kNN.predict_proba(x_test)
[:,1])
    plt.plot(fpr, tpr, label='KNN')
    fpr, tpr, thresholds = roc_curve(y_test,
most_frequent_dummy.predict_proba(x_test)[:,1])
    plt.plot(fpr, tpr, label='Most Frequent Dummy')
    fpr, tpr, thresholds = roc_curve(y_test,
random_dummy.predict_proba(x_test)[:,1])
    plt.plot(fpr, tpr, label='Random Dummy')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.legend(loc='lower right', fontsize = 10)
    plt.show()
    print("Logistic Regression AUC: ", roc_auc_score(y_poly_test,
best_logistic.decision_function(x_poly_test)))
    print("KNN AUC: ", roc_auc_score(y_test, best_kNN.predict_proba(x_test)
[:,1]))
    print("Most Frequent Dummy AUC: ", roc_auc_score(y_test,
most_frequent_dummy.predict_proba(x_test)[:,1]))
    print("Random Dummy AUC: ", roc_auc_score(y_test,
random_dummy.predict_proba(x_test)[:,1]))


# %%
# (i) data set week4_1.csv
print("Running on week4_1.csv:")
x, y = load_data('week4_1.csv')

plt.scatter(x[y == 1][:, 0],  x[y == 1][:, 1],  color='#00ff00', marker='+',
label='+1 Data')
plt.scatter(x[y == -1][:, 0], x[y == -1][:, 1], color='#0000fd', marker='o',
label='-1 Data')
plt.xlabel('X_1'); plt.ylabel('X_2')
plt.legend(loc='upper right', fontsize = 15)
plt.show()

# try polynomial degree from 1 to 9
best_degree = FindBestPolyDegree(x, y, range(1, 16))

# try C of penalty from 1 to 1000
```

```
best_penalty = FindBestC(x, y, best_degree, np.logspace(0, 11, 20, base=2))

# try C of penalty from 1 to 1000
best_penalty = FindBestC(x, y, best_degree, np.logspace(5, 7, 20, base=2))

best_logistic, y_best_logistic_pred, x_poly_test, y_poly_test =
DrawBestLogisticPredictions(x, y, best_degree, best_penalty)

best_k = FindBestKForKNN(x, y, range(1, 30))
best_kNN, y_best_kNN_pred = DrawBestKNNPrediction(x, y, best_k)
most_frequent_dummy, random_dummy = ConfutionMatrix(x, y,
y_best_logistic_pred, y_best_kNN_pred)
DrawROCCurve(x, y, x_poly_test, y_poly_test, best_logistic, best_kNN,
most_frequent_dummy, random_dummy)

# %%
# (ii)
print("Running on week4_2.csv")
x, y = load_data('week4_2.csv')

plt.scatter(x[y == 1][:, 0],  x[y == 1][:, 1],  color='#00ff00', marker='+',
label='+1 Data')
plt.scatter(x[y == -1][:, 0], x[y == -1][:, 1], color='#0000fd', marker='o',
label='-1 Data')
plt.xlabel('X_1'); plt.ylabel('X_2')
plt.legend(loc='upper right', fontsize = 15)
plt.show()

best_degree = FindBestPolyDegree(x, y, range(1, 16))
best_penalty = FindBestC(x, y, best_degree, np.logspace(-5, 5, 20, base=2))
best_logistic, y_best_logistic_pred, x_poly_test, y_poly_test =
DrawBestLogisticPredictions(x, y, best_degree, best_penalty)
best_k = FindBestKForKNN(x, y,  range(80, 120))
best_kNN, y_best_kNN_pred = DrawBestKNNPrediction(x, y, best_k)
most_frequent_dummy, random_dummy = ConfutionMatrix(x, y,
y_best_logistic_pred, y_best_kNN_pred)
DrawROCCurve(x, y, x_poly_test, y_poly_test, best_logistic, best_kNN,
most_frequent_dummy, random_dummy)
```