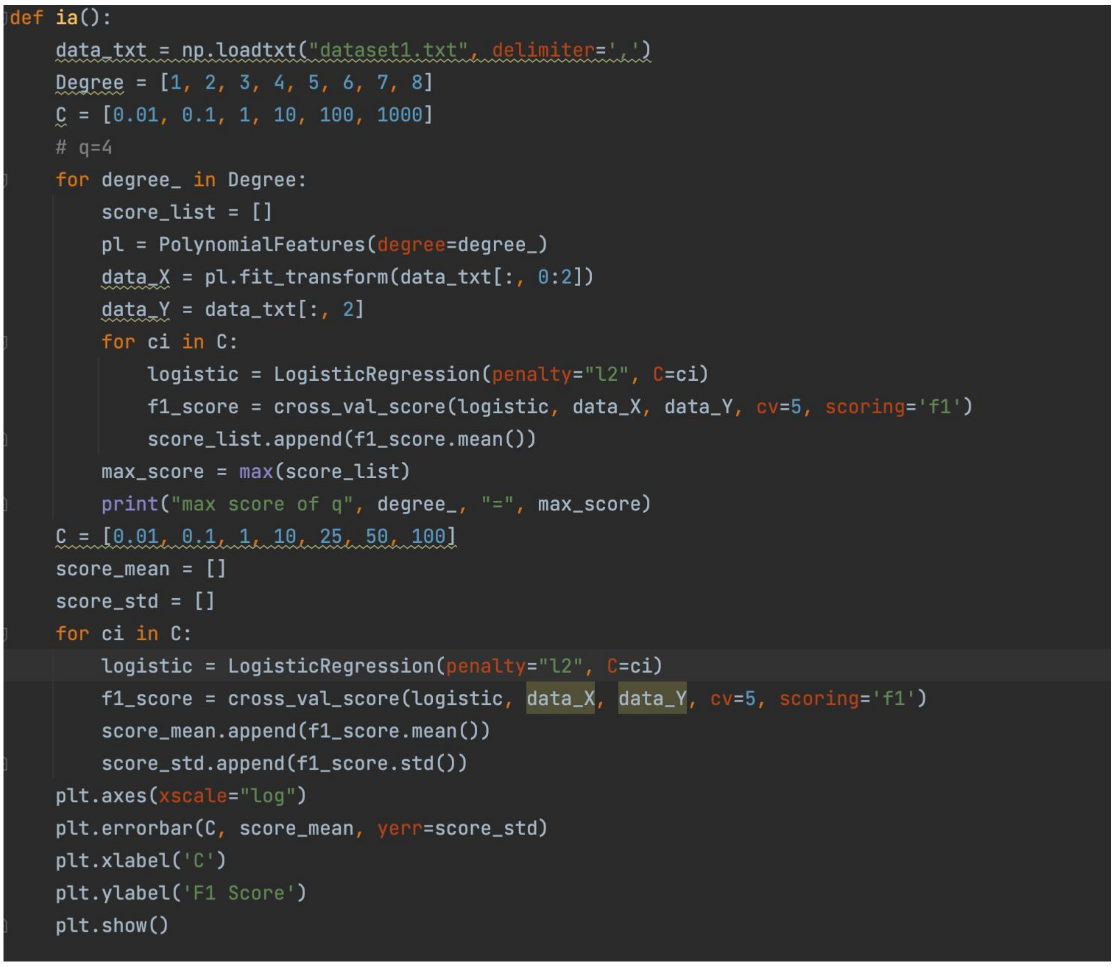
Machine Learning Week 4 Assignment

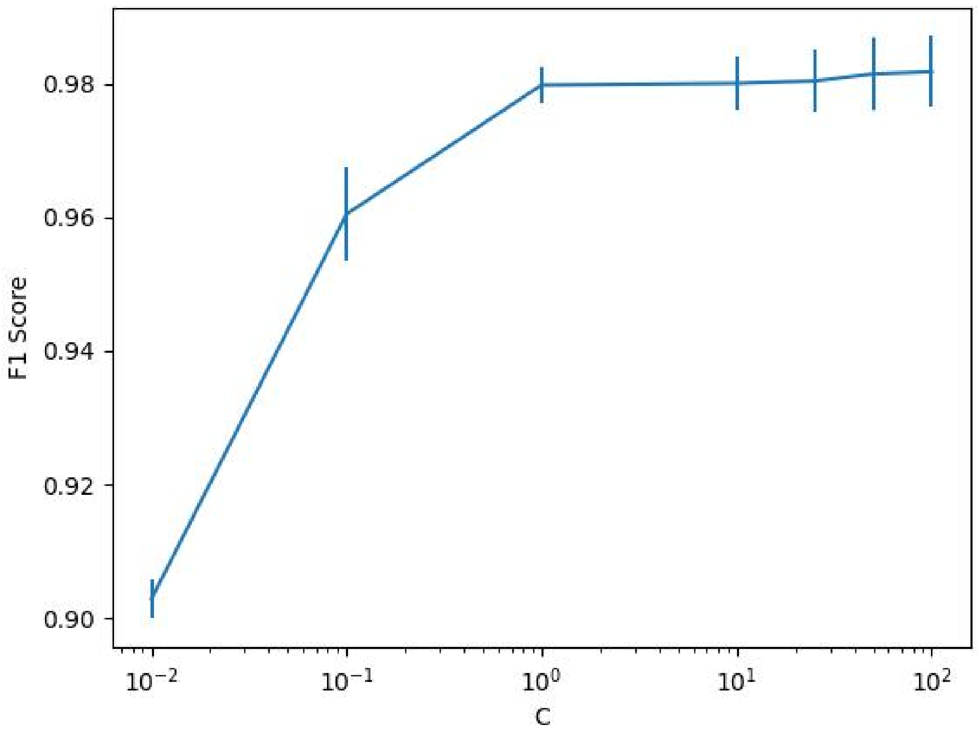
Jiaming Deng   22302794
dataset1 id:10-20--10-1   dataset2 id:10-10--10-1

i (a) Firstly, I copy the data from the web page into a txt file and use numpy's load_txt function to read it. Secondly, I try to get the best order of polynomial. I test it from 1 to 8 and use a large range of C to get the max of F1 score. The result is as the table below shows and the max F1 score is 0.9827 so I choose 4 as the best order of polynomial. Finally, I test C between 0.01 and 100 and plot a picture to get the best C, which is 100.

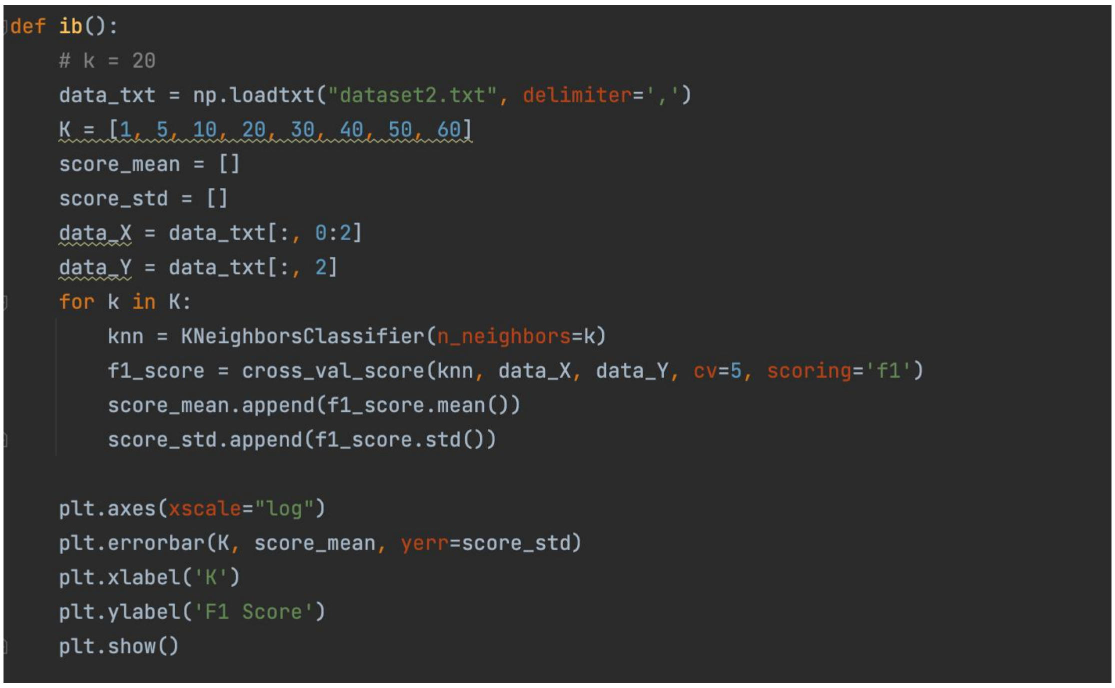


```python
def ia():
    data_txt = np.loadtxt("dataset1.txt", delimiter=',')
    Degree = [1, 2, 3, 4, 5, 6, 7, 8]
    C = [0.01, 0.1, 1, 10, 100, 1000]
    # q=4
    for degree_ in Degree:
        score_list = []
        pl = PolynomialFeatures(degree=degree_)
        data_X = pl.fit_transform(data_txt[:, 0:2])
        data_Y = data_txt[:, 2]
        for ci in C:
            logistic = LogisticRegression(penalty="l2", C=ci)
            f1_score = cross_val_score(logistic, data_X, data_Y, cv=5, scoring='f1')
            score_list.append(f1_score.mean())
        max_score = max(score_list)
        print("max score of q", degree_, "=", max_score)
    C = [0.01, 0.1, 1, 10, 25, 50, 100]
    score_mean = []
    score_std = []
    for ci in C:
        logistic = LogisticRegression(penalty="l2", C=ci)
        f1_score = cross_val_score(logistic, data_X, data_Y, cv=5, scoring='f1')
        score_mean.append(f1_score.mean())
        score_std.append(f1_score.std())
    plt.axes(xscale="log")
    plt.errorbar(C, score_mean, yerr=score_std)
    plt.xlabel('C')
    plt.ylabel('F1 Score')
    plt.show()
```

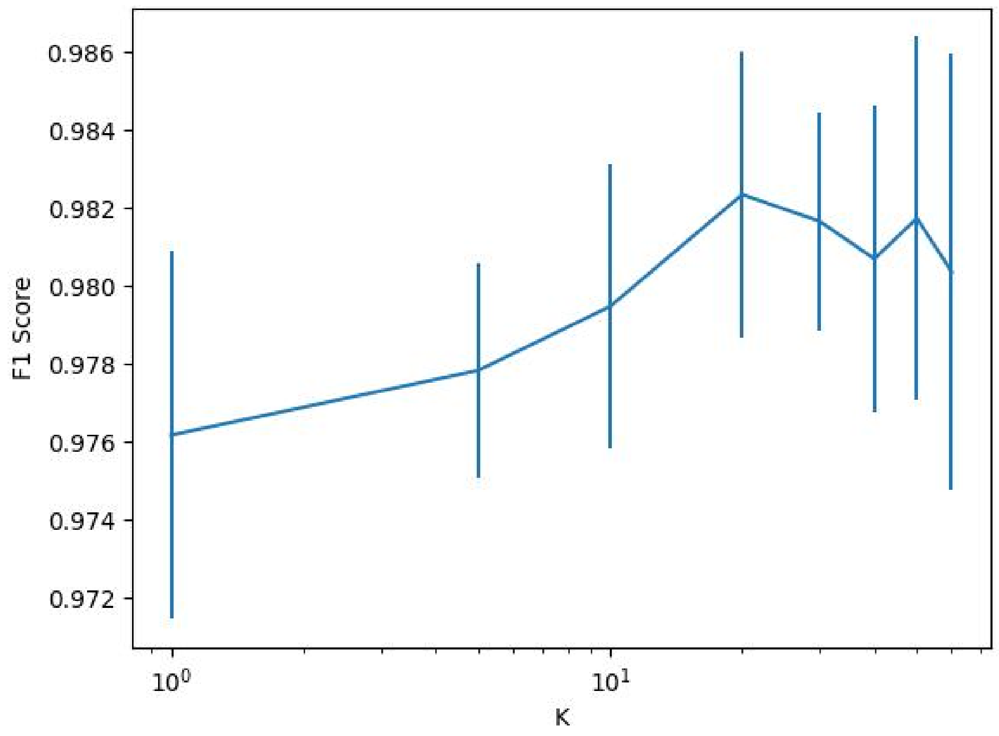
Q	Max F1 Score
1	0.8830
2	0.9817
3	0.9824
4	0.9827
5	0.9820
6	0.9823
7	0.9820
8	0.9817



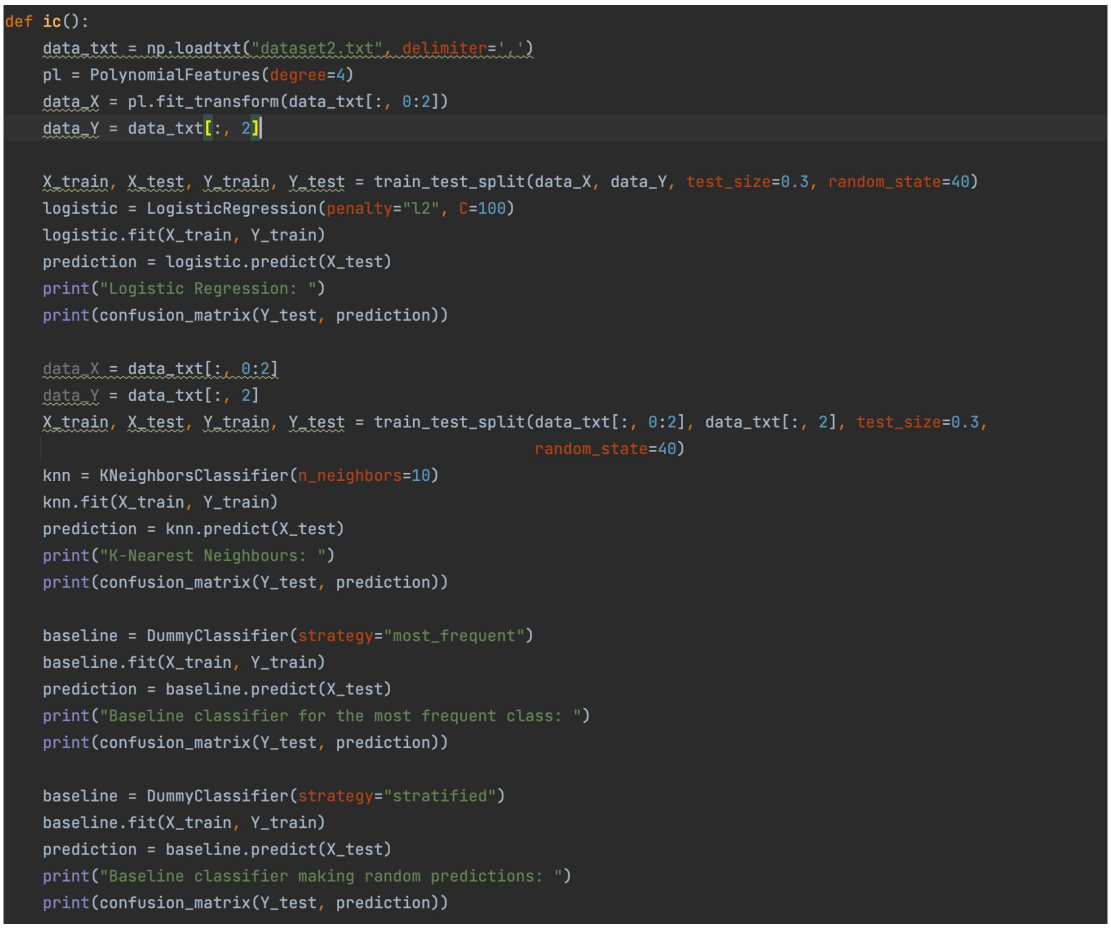
i (b) I choose K between 1 and 60 as n_neighbors argument of knn model, I get the F1 score and F1 score's standard deviation and plot the data using matplotlib's errorbar function and get a picuture below. So the best K is 20 because the F1 score is the highest and standard deviation is not worse than others.



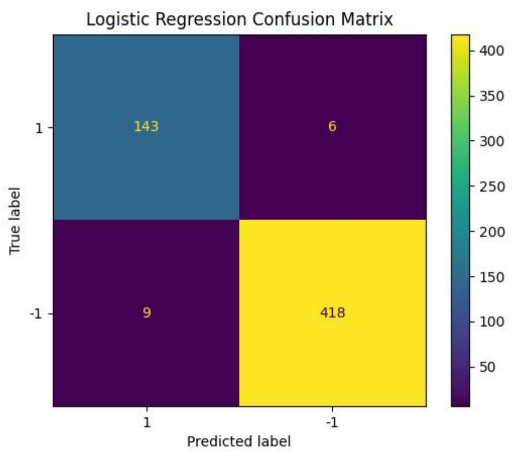
```
def ib():
    # k = 20
    data_txt = np.loadtxt("dataset2.txt", delimiter=',')
    K = [1, 5, 10, 20, 30, 40, 50, 60]
    score_mean = []
    score_std = []
    data_X = data_txt[:, 0:2]
    data_Y = data_txt[:, 2]
    for k in K:
        knn = KNeighborsClassifier(n_neighbors=k)
        f1_score = cross_val_score(knn, data_X, data_Y, cv=5, scoring='f1')
        score_mean.append(f1_score.mean())
        score_std.append(f1_score.std())

    plt.axes(xscale="log")
    plt.errorbar(K, score_mean, yerr=score_std)
    plt.xlabel('K')
    plt.ylabel('F1 Score')
    plt.show()
```

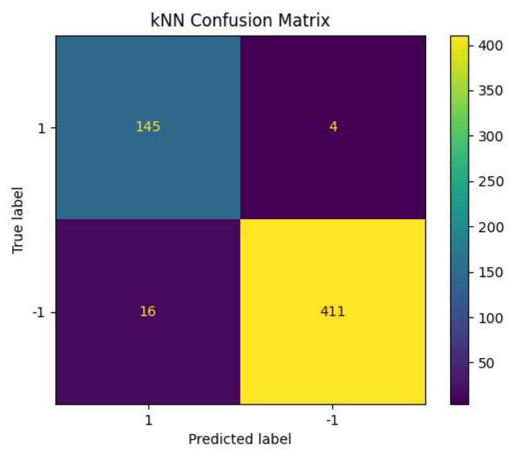


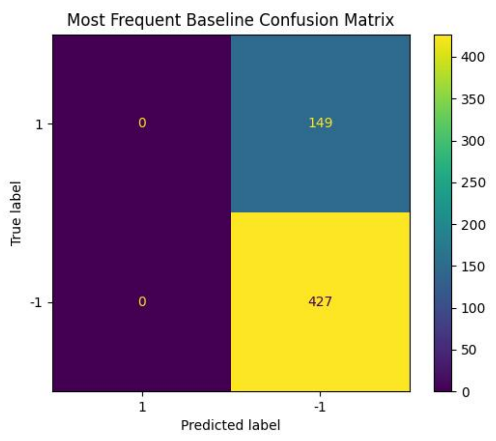
i (c) I train Logistic Regression, kNN and two baseline models. One of the baseline models always predicts the most frequent class label in the observed y in the training data and another one generates predictions uniformly at random from the list of unique class observed in y. Finally I get the confusion matrices.

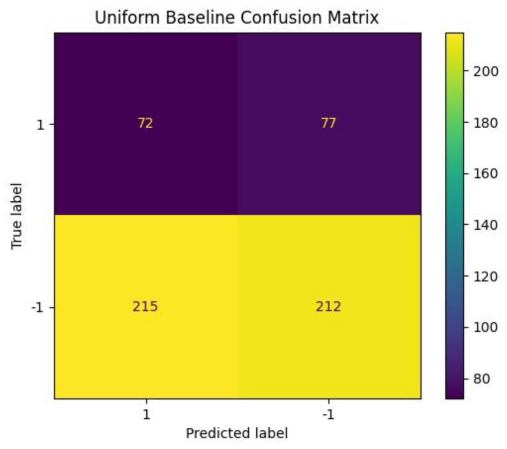


```
def ic():
    data_txt = np.loadtxt("dataset2.txt", delimiter=',')
    pl = PolynomialFeatures(degree=4)
    data_X = pl.fit_transform(data_txt[:, 0:2])
    data_Y = data_txt[:, 2]

    X_train, X_test, Y_train, Y_test = train_test_split(data_X, data_Y, test_size=0.3, random_state=40)
    logistic = LogisticRegression(penalty="l2", C=100)
    logistic.fit(X_train, Y_train)
    prediction = logistic.predict(X_test)
    print("Logistic Regression: ")
    print(confusion_matrix(Y_test, prediction))

    data_X = data_txt[:, 0:2]
    data_Y = data_txt[:, 2]
    X_train, X_test, Y_train, Y_test = train_test_split(data_txt[:, 0:2], data_txt[:, 2], test_size=0.3,
                                                        random_state=40)
    knn = KNeighborsClassifier(n_neighbors=10)
    knn.fit(X_train, Y_train)
    prediction = knn.predict(X_test)
    print("K-Nearest Neighbours: ")
    print(confusion_matrix(Y_test, prediction))

    baseline = DummyClassifier(strategy="most_frequent")
    baseline.fit(X_train, Y_train)
    prediction = baseline.predict(X_test)
    print("Baseline classifier for the most frequent class: ")
    print(confusion_matrix(Y_test, prediction))

    baseline = DummyClassifier(strategy="stratified")
    baseline.fit(X_train, Y_train)
    prediction = baseline.predict(X_test)
    print("Baseline classifier making random predictions: ")
    print(confusion_matrix(Y_test, prediction))
```

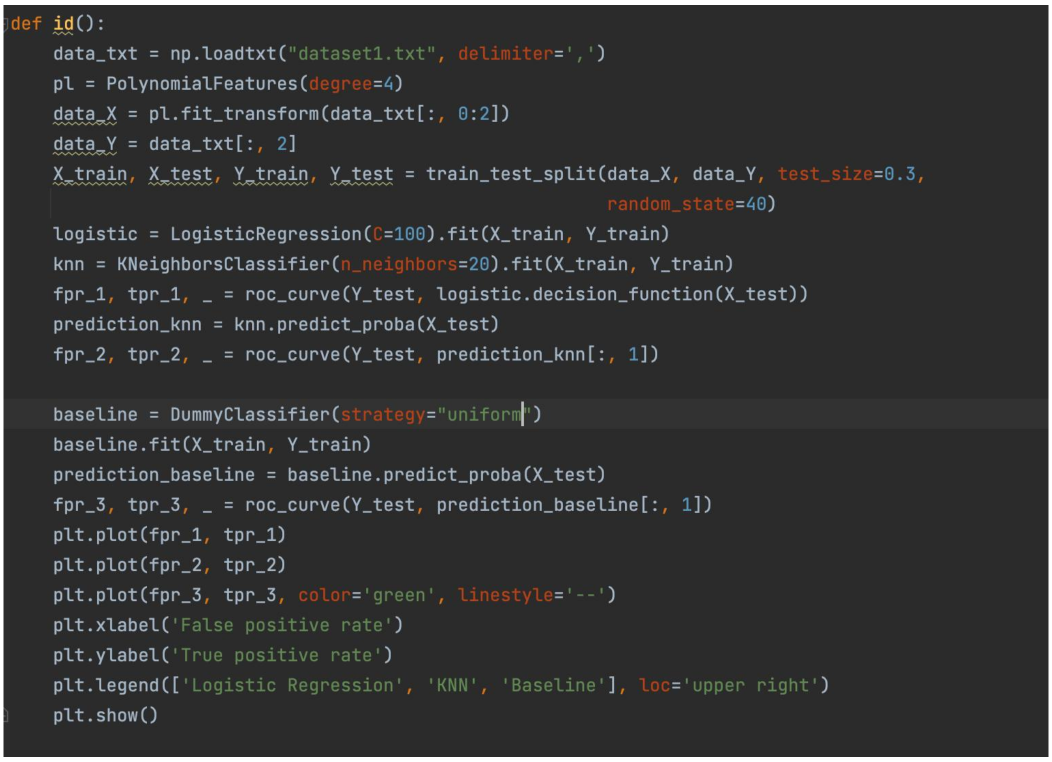




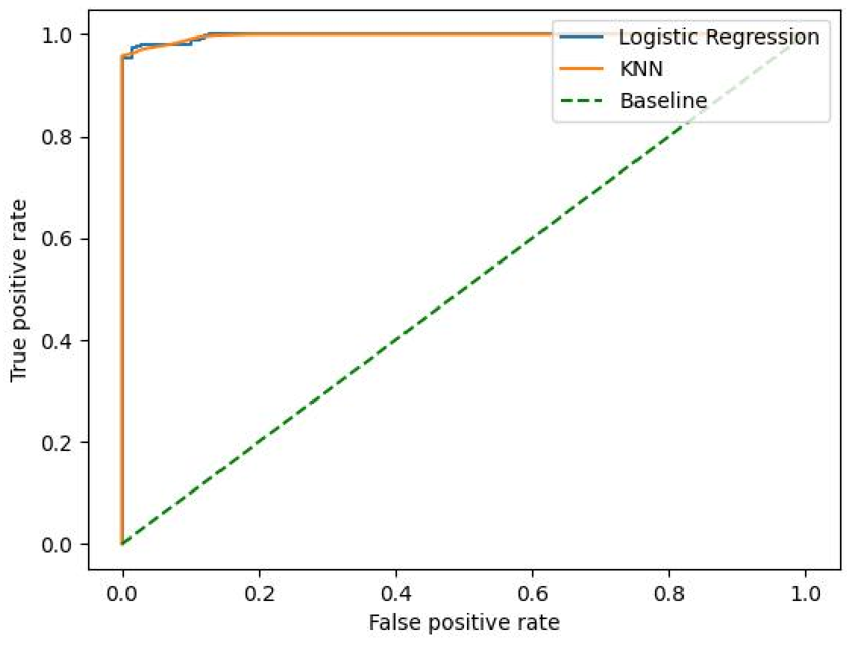




i (d) I train Logistic Regression, kNN and baseline model and get ROC curves by roc_curve function.



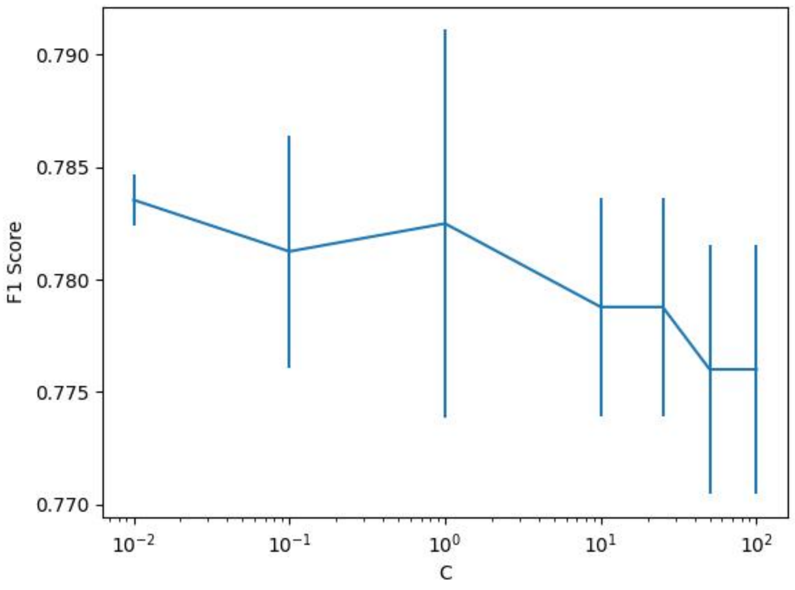
```python
def id():
    data_txt = np.loadtxt("dataset1.txt", delimiter=',')
    pl = PolynomialFeatures(degree=4)
    data_X = pl.fit_transform(data_txt[:, 0:2])
    data_Y = data_txt[:, 2]
    X_train, X_test, Y_train, Y_test = train_test_split(data_X, data_Y, test_size=0.3,
                                                        random_state=40)
    logistic = LogisticRegression(C=100).fit(X_train, Y_train)
    knn = KNeighborsClassifier(n_neighbors=20).fit(X_train, Y_train)
    fpr_1, tpr_1, _ = roc_curve(Y_test, logistic.decision_function(X_test))
    prediction_knn = knn.predict_proba(X_test)
    fpr_2, tpr_2, _ = roc_curve(Y_test, prediction_knn[:, 1])

    baseline = DummyClassifier(strategy="uniform")
    baseline.fit(X_train, Y_train)
    prediction_baseline = baseline.predict_proba(X_test)
    fpr_3, tpr_3, _ = roc_curve(Y_test, prediction_baseline[:, 1])
    plt.plot(fpr_1, tpr_1)
    plt.plot(fpr_2, tpr_2)
    plt.plot(fpr_3, tpr_3, color='green', linestyle='--')
    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.legend(['Logistic Regression', 'KNN', 'Baseline'], loc='upper right')
    plt.show()
```



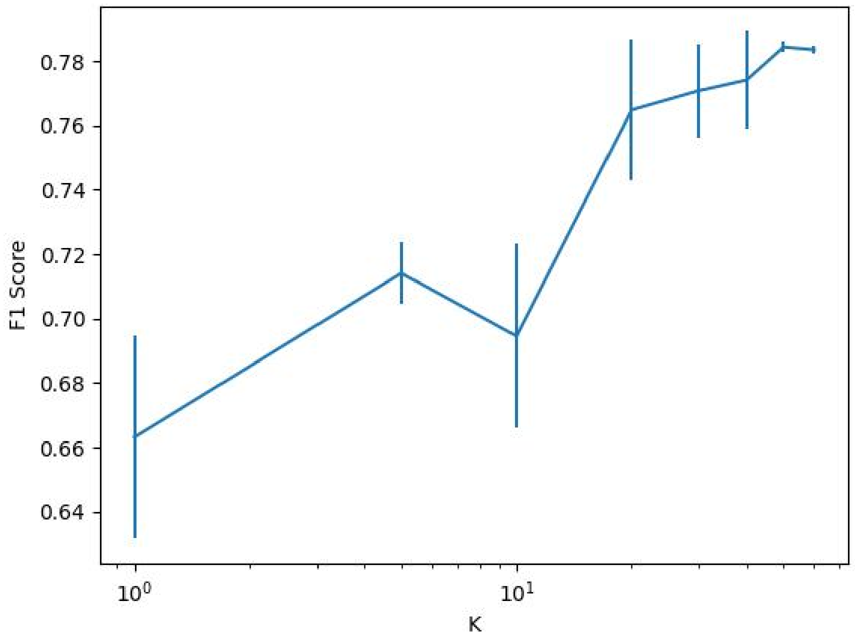
i (e) According to the confusion matrix, Logistic Regression and kNN model are much better than baseline models, but the difference between Logistic Regression model and kNN model is very small. According to the ROC curves, the ROC curve of Logistic Regression model and that of kNN are almost same. The ROC curve of random baseline is a line but Logistic Regression and kNN's ROC curve are curves. For this dataset, both of the Logistic Regression and kNN model are recommended by me. Because according to the confusion matrix and ROC curve, these two models show good performance.

ii (a) The explanations of code and choice are the same as question 1. The result is as the table below shows and the max F1 score is 0.7841 so I choose 5 as the best order of polynomial. I test C between 0.01 and 100 and plot a picture to get the best C, which is 0.01.

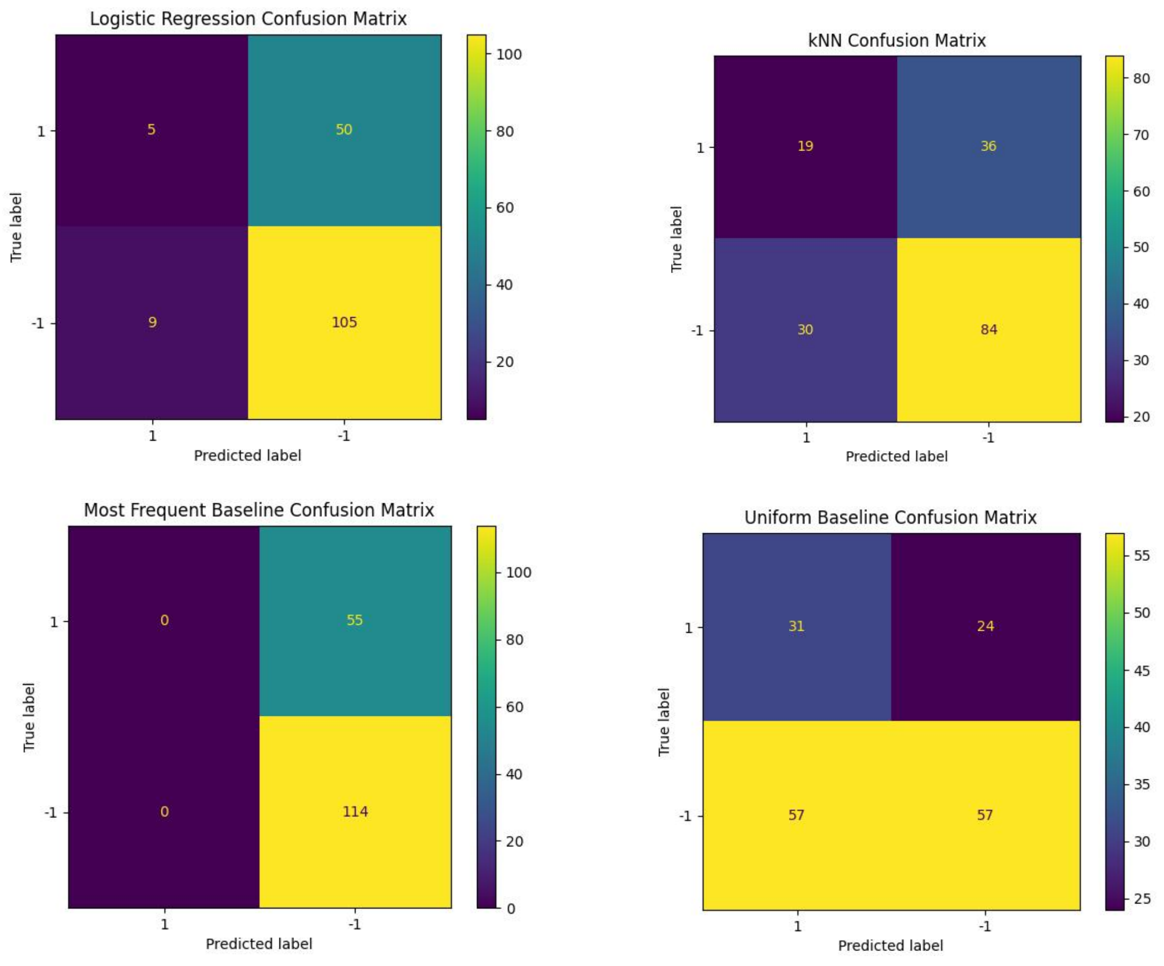
Q	Max F1 Score
1	0.7835
2	0.7835
3	0.7835
4	0.7835
5	0.7841
6	0.7834
7	0.7835
8	0.7835



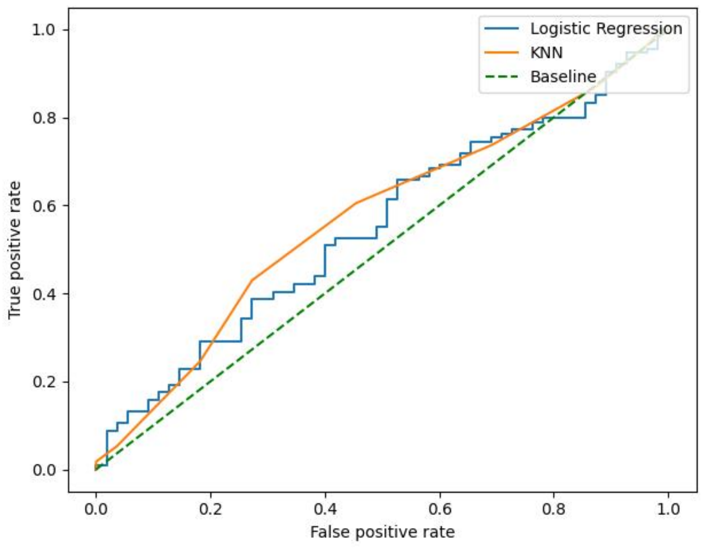
ii (b) The best K is 50 because the F1 score is the highest and standard deviation is the least.



ii (c) The confusion matrices of Logistic Regression, kNN, two baseline models are shown below.



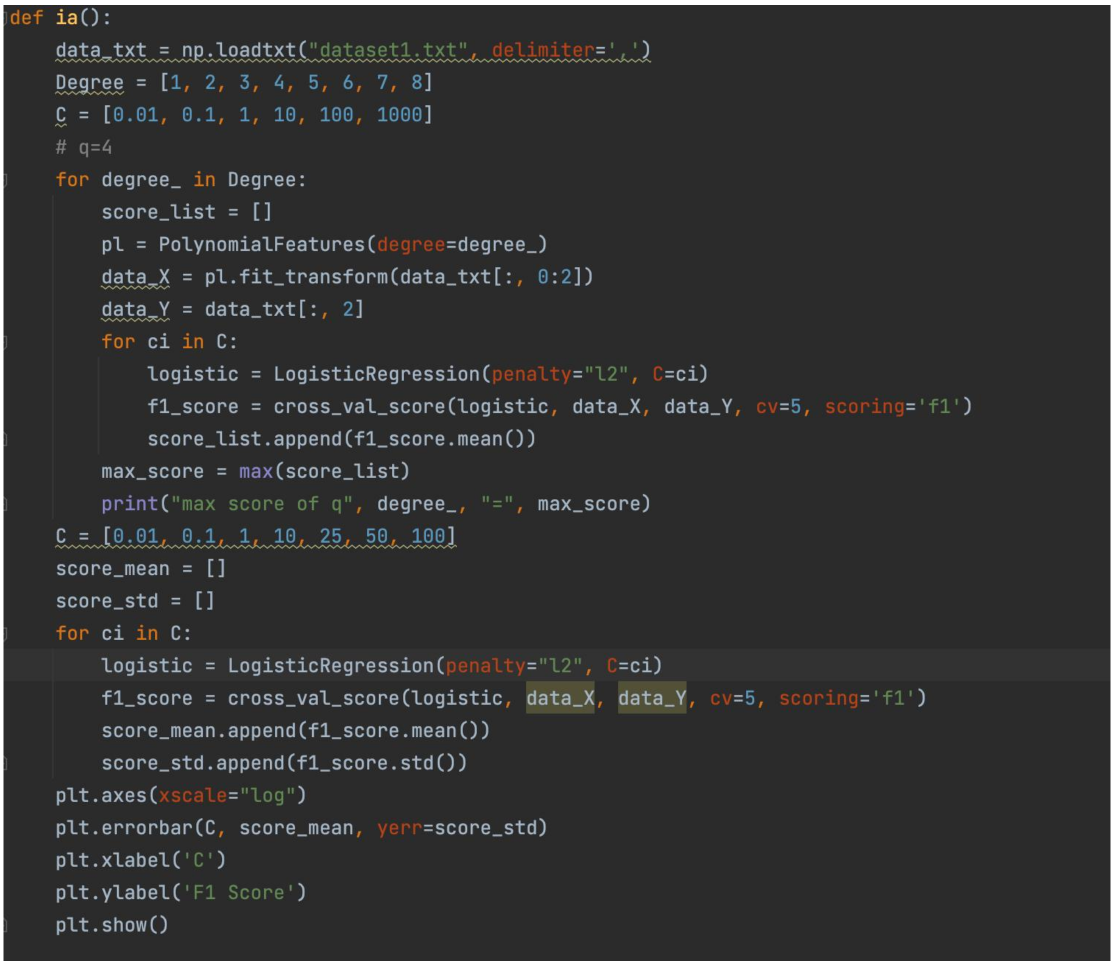
ii (d) I train Logistic Regression, kNN and baseline model and get ROC curves by roc_curve function.



ii (e) According to the confusion matrix, no classfier significantly better or worse than others. According to the ROC curves, All of these curves are not much different. The ROC curve of random baseline is a line but Logistic Regression's ROC curve presents a ladder shape and kNN's ROC curve is smooth curve. For this dataset, I recommend kNN model. Because according to the confusion matrix and ROC curve, this model shows better performance.

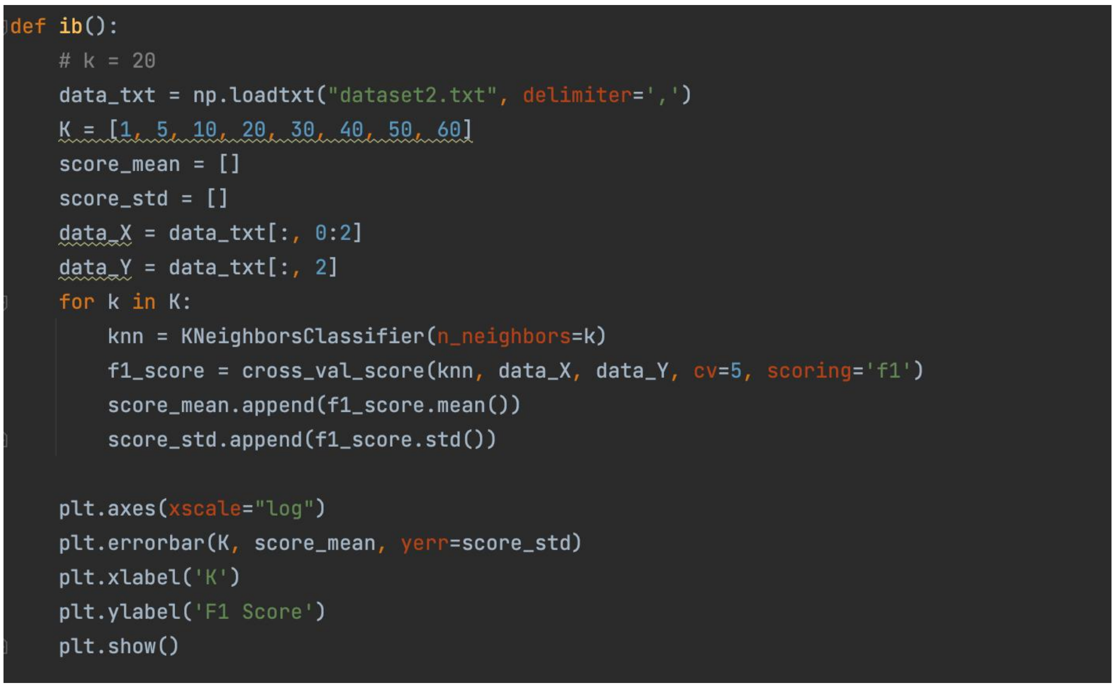
Appenix

(a)



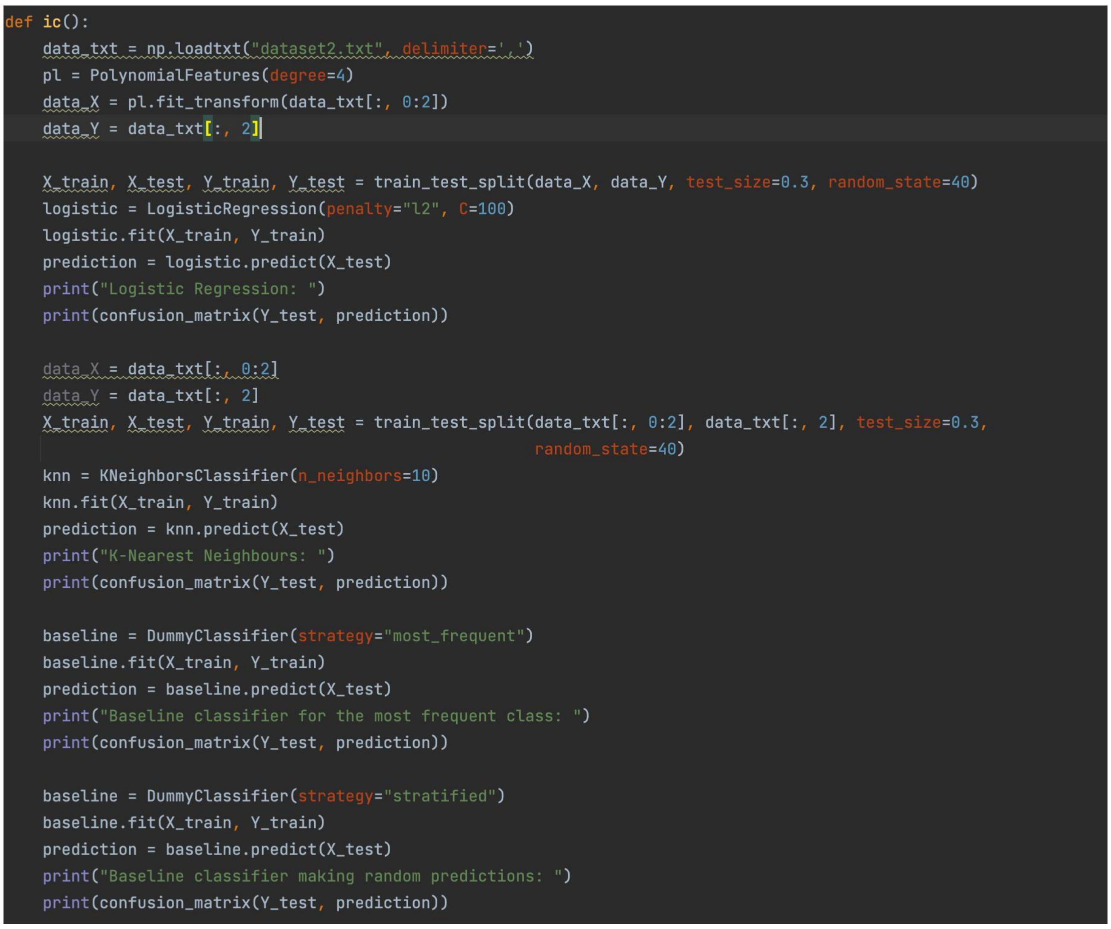
```python
def ia():
    data_txt = np.loadtxt("dataset1.txt", delimiter=',')
    Degree = [1, 2, 3, 4, 5, 6, 7, 8]
    C = [0.01, 0.1, 1, 10, 100, 1000]
    # q=4
    for degree_ in Degree:
        score_list = []
        pl = PolynomialFeatures(degree=degree_)
        data_X = pl.fit_transform(data_txt[:, 0:2])
        data_Y = data_txt[:, 2]
        for ci in C:
            logistic = LogisticRegression(penalty="l2", C=ci)
            f1_score = cross_val_score(logistic, data_X, data_Y, cv=5, scoring='f1')
            score_list.append(f1_score.mean())
        max_score = max(score_list)
        print("max score of q", degree_, "=", max_score)
    C = [0.01, 0.1, 1, 10, 25, 50, 100]
    score_mean = []
    score_std = []
    for ci in C:
        logistic = LogisticRegression(penalty="l2", C=ci)
        f1_score = cross_val_score(logistic, data_X, data_Y, cv=5, scoring='f1')
        score_mean.append(f1_score.mean())
        score_std.append(f1_score.std())
    plt.axes(xscale="log")
    plt.errorbar(C, score_mean, yerr=score_std)
    plt.xlabel('C')
    plt.ylabel('F1 Score')
    plt.show()
```

(b)



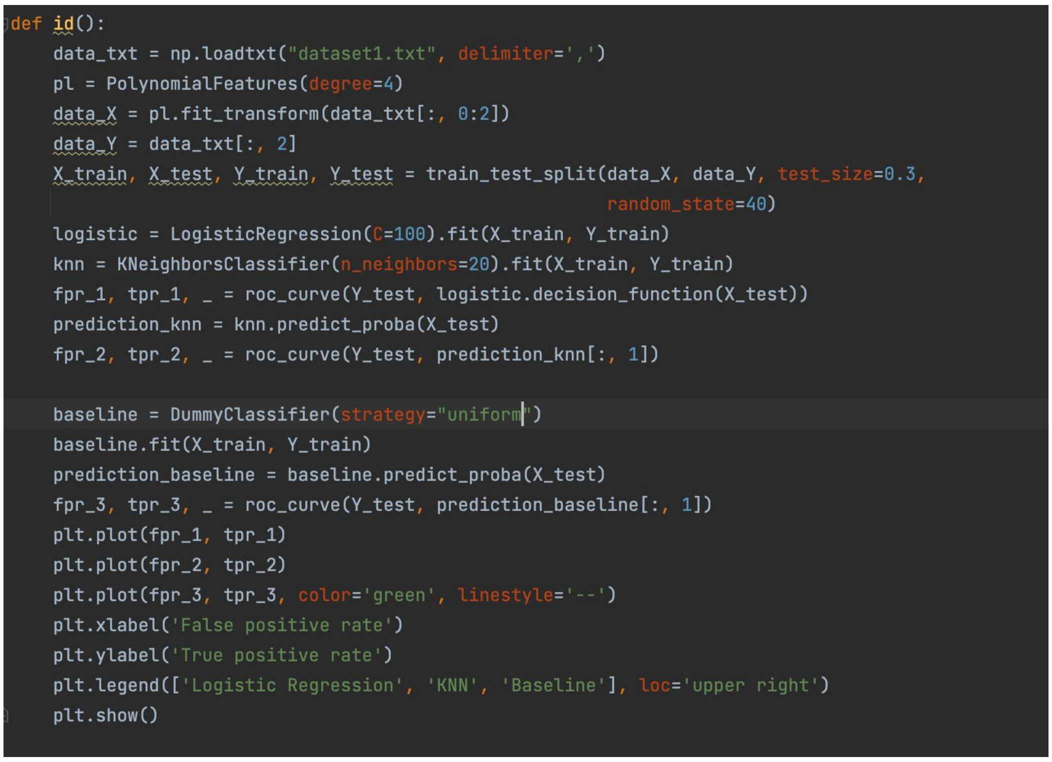
```python
def ib():
    # k = 20
    data_txt = np.loadtxt("dataset2.txt", delimiter=',')
    K = [1, 5, 10, 20, 30, 40, 50, 60]
    score_mean = []
    score_std = []
    data_X = data_txt[:, 0:2]
    data_Y = data_txt[:, 2]
    for k in K:
        knn = KNeighborsClassifier(n_neighbors=k)
        f1_score = cross_val_score(knn, data_X, data_Y, cv=5, scoring='f1')
        score_mean.append(f1_score.mean())
        score_std.append(f1_score.std())

    plt.axes(xscale="log")
    plt.errorbar(K, score_mean, yerr=score_std)
    plt.xlabel('K')
    plt.ylabel('F1 Score')
    plt.show()
```

(c)



```python
def ic():
    data_txt = np.loadtxt("dataset2.txt", delimiter=',')
    pl = PolynomialFeatures(degree=4)
    data_X = pl.fit_transform(data_txt[:, 0:2])
    data_Y = data_txt[:, 2]

    X_train, X_test, Y_train, Y_test = train_test_split(data_X, data_Y, test_size=0.3, random_state=40)
    logistic = LogisticRegression(penalty="l2", C=100)
    logistic.fit(X_train, Y_train)
    prediction = logistic.predict(X_test)
    print("Logistic Regression: ")
    print(confusion_matrix(Y_test, prediction))

    data_X = data_txt[:, 0:2]
    data_Y = data_txt[:, 2]
    X_train, X_test, Y_train, Y_test = train_test_split(data_txt[:, 0:2], data_txt[:, 2], test_size=0.3,
                                                        random_state=40)
    knn = KNeighborsClassifier(n_neighbors=10)
    knn.fit(X_train, Y_train)
    prediction = knn.predict(X_test)
    print("K-Nearest Neighbours: ")
    print(confusion_matrix(Y_test, prediction))

    baseline = DummyClassifier(strategy="most_frequent")
    baseline.fit(X_train, Y_train)
    prediction = baseline.predict(X_test)
    print("Baseline classifier for the most frequent class: ")
    print(confusion_matrix(Y_test, prediction))

    baseline = DummyClassifier(strategy="stratified")
    baseline.fit(X_train, Y_train)
    prediction = baseline.predict(X_test)
    print("Baseline classifier making random predictions: ")
    print(confusion_matrix(Y_test, prediction))
```

(d)



```
def id():
    data_txt = np.loadtxt("dataset1.txt", delimiter=',')
    pl = PolynomialFeatures(degree=4)
    data_X = pl.fit_transform(data_txt[:, 0:2])
    data_Y = data_txt[:, 2]
    X_train, X_test, Y_train, Y_test = train_test_split(data_X, data_Y, test_size=0.3,
                                                        random_state=40)
    logistic = LogisticRegression(C=100).fit(X_train, Y_train)
    knn = KNeighborsClassifier(n_neighbors=20).fit(X_train, Y_train)
    fpr_1, tpr_1, _ = roc_curve(Y_test, logistic.decision_function(X_test))
    prediction_knn = knn.predict_proba(X_test)
    fpr_2, tpr_2, _ = roc_curve(Y_test, prediction_knn[:, 1])

    baseline = DummyClassifier(strategy="uniform")
    baseline.fit(X_train, Y_train)
    prediction_baseline = baseline.predict_proba(X_test)
    fpr_3, tpr_3, _ = roc_curve(Y_test, prediction_baseline[:, 1])
    plt.plot(fpr_1, tpr_1)
    plt.plot(fpr_2, tpr_2)
    plt.plot(fpr_3, tpr_3, color='green', linestyle='--')
    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.legend(['Logistic Regression', 'KNN', 'Baseline'], loc='upper right')
    plt.show()
```