Week 4 assignment

Name: Tianze Zhang
Student number: 19334401

First Dataset

(i)(a) The overall visualization is presented as figure 1.0, we can see that the overall data is spread evenly where +1 labels are mainly presented at the top-middle of the graph, and the rest are -1. Therefore, no removals of outliers and standardization done.

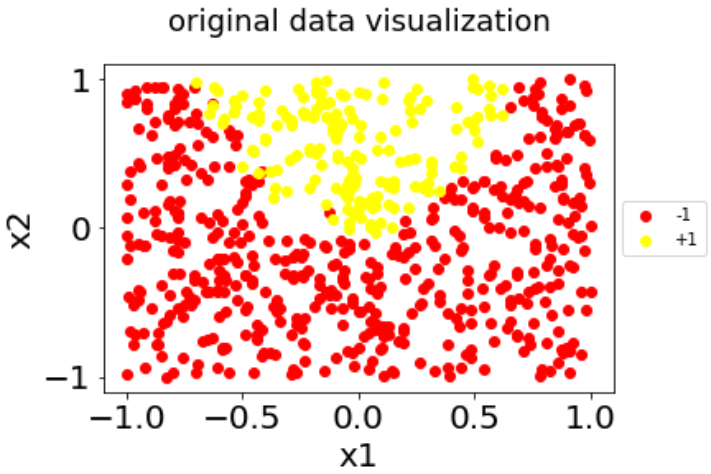


Figure 1.0

The maximum order of polynomial to use is found using hyperparameter tuning and cross validation. The penalty parameter c and the number of features are cross validated based on the f1 score. The f1 score which is 2TP/2TP+FN+FP, when tuning model, using F1 score as a single value, F1 is the overall effects of both false positives and false negatives (wrong prediction), combining precision and recall as simple value. Since we want to minimize the FP and FN, we want to maximize the F1 score. Therefore, the value of q and C is choose based on the higher F1 score.

Figure 1.1 shows cross validation of increasing features from 1 to 6. As you can see the f1 score rises significantly from q=1 to q=2, and then stays relatively the same when q>=2. Choosing q=1 which corresponds to a small f1 scores results in underfitting. When the f1 score is stable while the q increases, however we want to choose the minimum q possible to avoid overfitting, which result in good in training, but bad in testing. The trade-off is the right balance between them, then achieve the good generalisation. Therefore, a q value of 2 is chosen for the logistic classifier.

Figure 1.3 shows cross validation of increasing penalty parameter C in a range of 0 to 100. As you can see the f1 score rises significantly from C=0 to C=1 and then stays relatively the same when C>=1. Choosing c<1 may result in underfitting. However, since C is the largest at 1 and to avoid overfitting a C value of 1 is chosen.

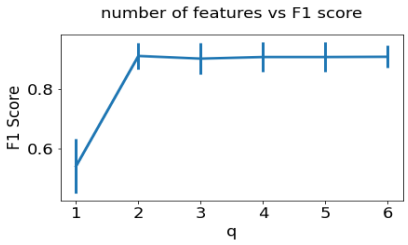


Figure 1.1

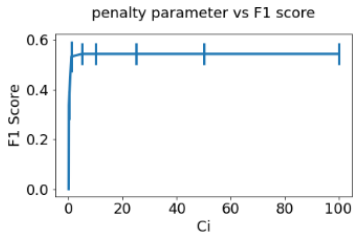


Figure 1.2

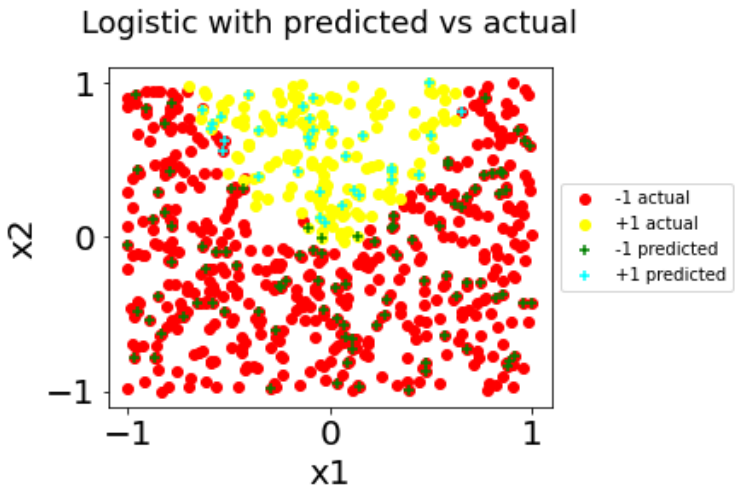


Figure 1.3

As Figure 1.3, the prediction on the test set looks mostly accurate visually. Also, this can be evaluated by different metrics for example, using confusion matrix with rates accuracy, is measurement of overall prediction correctness. True positive rate or recall, measuring when actually positive, how good to predict positive. False positive rate or specificity, measuring when actually is negative, how often predict positive, type I error.

As calculated, the confusion metrics of this model is [[94,3], [1,32]], has a precision of 0.91, which means when the model predicts positive it only misclassifies 9 data out of 100. while the model's recall is 0.97 which means when the model predicts negative it only misclassifies 3 out of 100.

(b)

As discussed in part (a), KNN classifier's K value is also selected by cross validating against the f1 score with a range of k values from 1 to 10. As you can see in Figure 2.0, when k increases from 0 to 1 there is a significant change to the f1 score, and when k further increases the f1 score decreases. Since the f1 score maximizes when k=1, then k=1 will be selected while training the model.

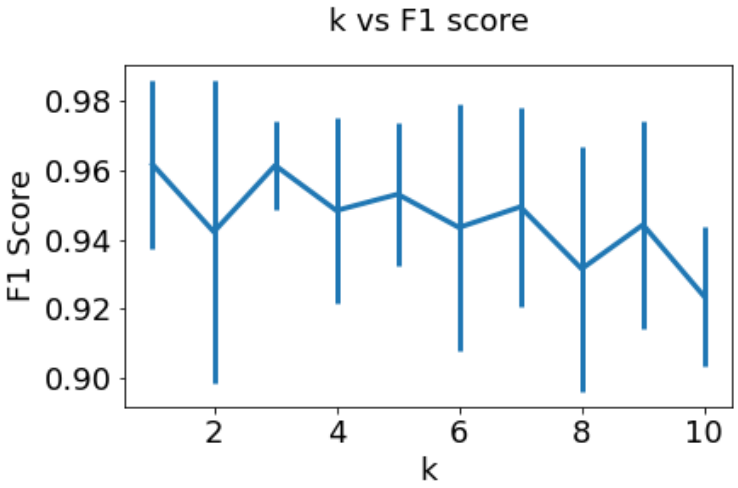


Figure 2.0

As Figure 2.1 shown, KNN also predicts accurately with the test data visually. The confusion matrix of the KNN model is [[96,1], [1,32]] with a precision of 0.97 and recall of 0.99. This means when the KNN model predicts positively it would only misclassify 3 out of 100 data, when the KNN predicts negatively the model only misclassifies 1 out of 100 data.

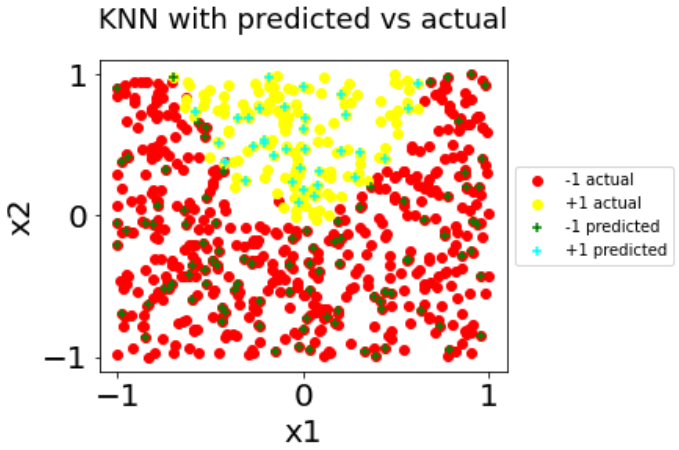


Figure 2.1

(c) As listed above, the confusion matrix for the logistic classifier is [[94,3], [1,32]] and for the KNN classifier it is [[96,1], [1,32]]. A baseline model is created using the dummy classifier from sklearn which predicts based on the most frequent class label in the observed y argument passed to fit. Therefore, it predicts the same label every time with the most occurrence. The confusion matrix of the baseline model is [[97,0], [33,0]] which still results In a accuracy f1 score of 0.75.

(d) As figure 3.3, the ROC curve is plotted with the blue line representing the logistic regression and the orange representing the KNN classifier and the dotted line which is the baseline model.

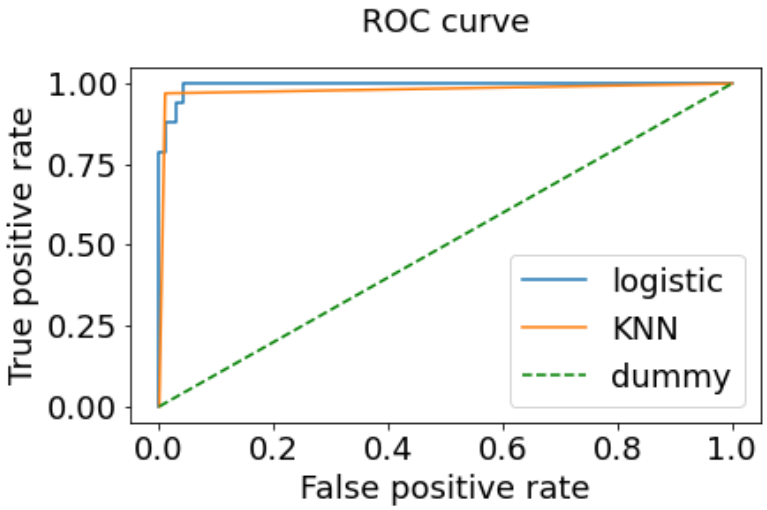


Figure 3.0

(e)I would recommend the KNN model to be use. However, the KNN is only slightly better than the logistic classifier when it comes to precision where the KNN does a better job at predicting positives than logistic, with 96 TP and 1 FP vs 94 TP and 3 FP based on the confusion matrix. Both the KNN and logistic have a similar performs when predicting negative, both 1 FN and 32 TP. However, based on the ROC curve, we can see that the KNN classifier is more to the top-left corner than the logistic classifier. Which means more true positive rates than false positive rates, resulting a greater AUC, therefore also proves the KNN model is slightly better. Both the KNN and logistic outperforms the random classifier when the data is negative because it predicts positive all the time. Although it has a higher accuracy when the data is positive, however, it only leads the KNN model by 1% accuracy but losing 97% on negative data. The ROC of the random classifier is at a 45% line, furthest to the top-left corner.

Second Dataset

As Figure 4.0, the overall of the dataset is 'noise', it is hard to discover any correlation between x1 and x2.

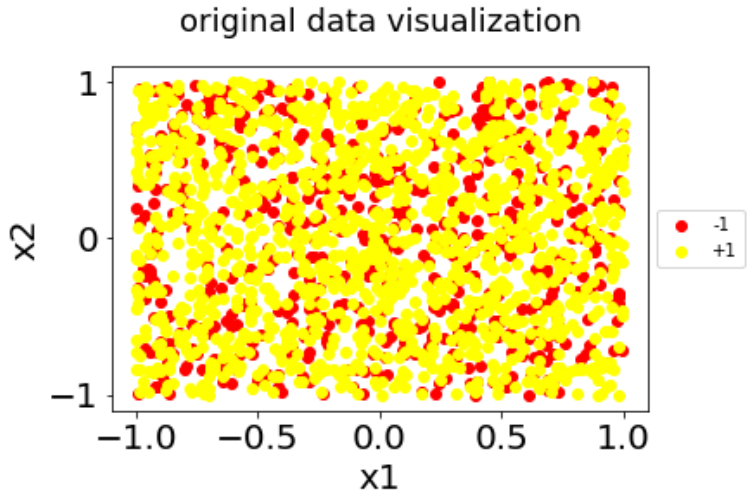


Figure 4.0

As Figure 4.1 and Figure 4.2 shown, increasing the penalty parameter and the number of features has to effect on the accuracy f1 score, since the data are uncorrelated. The f1 score stays the same.

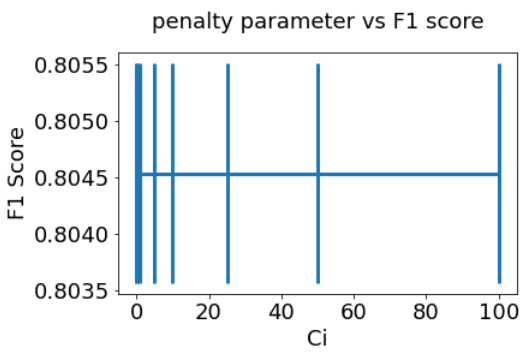


Figure 4.1

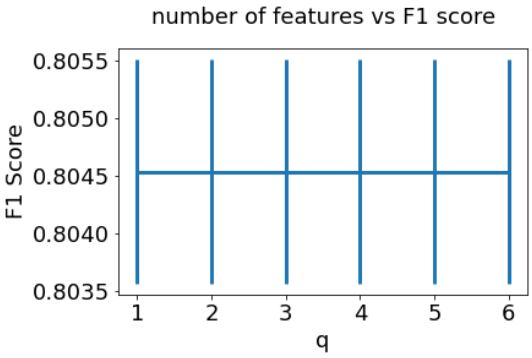


Figure 4.2

As figure 4.3 shown, if trying to train a logistic regression with C of 1 and q of 1, it does not make any reasonable prediction, predicting +1 every time. It has a precision of 0.67 and recall of 0.

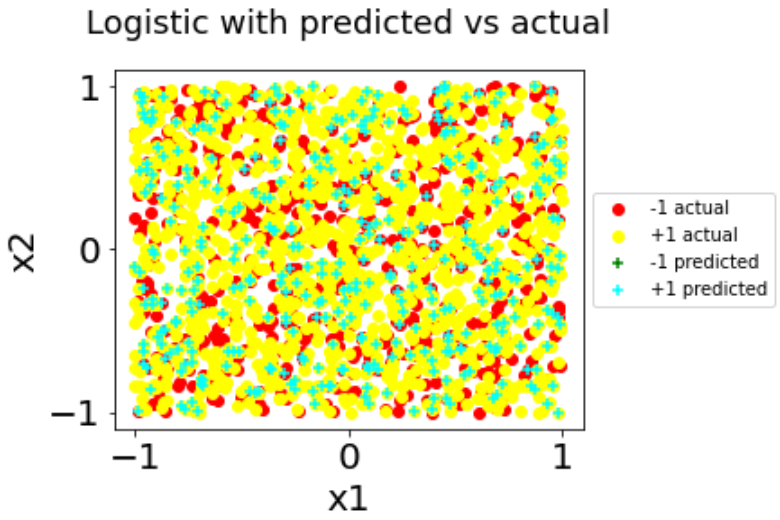
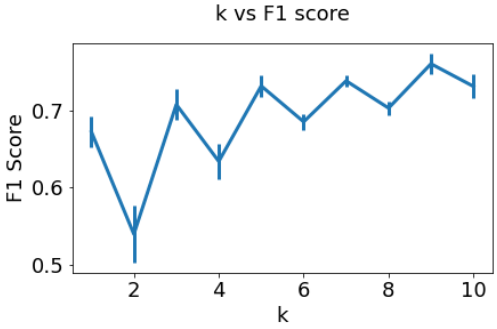


Figure 4.3

(b) As figure 5.0, increase amount of k leads to a higher f1 score, this is because since the data is widely distributed and uncorrelated the larger the k we includes as a group the better. However, it doesn't really help classifying the different labels.



`

Figure 5.0

As Figure 5.1, the KNN model with k=10 also performs not well, predicting mostly the same label with a confusion matrix of [[12,109] , [25, 224]], with a precision of 0.67 and recall of 0.1.

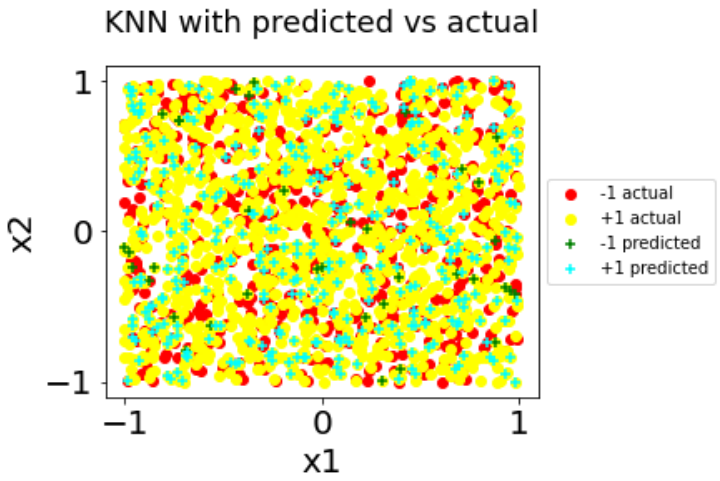


Figure 5.1

(c) If we train a random classifier same as the first part discussed before, we get a confusion matrix of [[0,121] , [0, 249]], with 0.67 precision and 0 recall. While the confusion matrix for logistic is [[0,121], [0, 249]] and KNN [[12,109], [25, 224]].

(d)

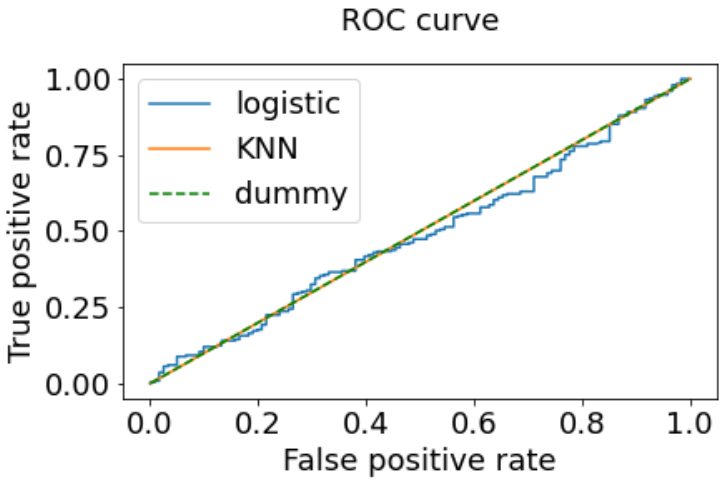


Figure 6.0

(e) None of those three models are good enough, since the data is uncorrelated and consist of only 'noise'. We can see that both three models gives similar confusion matrix, with a low TP rate but high TN rate. And the AUC of the RUC are similar, all 45 degrees from origin. In this case, since the trained model performs no better than the random classifier then it's no need to even train a model.

Appendix

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.dummy import DummyClassifier
from sklearn.metrics import roc_curve

df = pd.read_csv('week4(1).txt')
print(df.head())
# data for visulization
neg = df[df.y==-1]
pos = df[df.y==1]
x1_neg = neg.iloc[:,0]
x2_neg = neg.iloc[:,1]
x1_pos = pos.iloc[:,0]
x2_pos = pos.iloc[:,1]
# data for training and testing
x1 = df.iloc[:,0]
x2 = df.iloc[:,1]
X=np.column_stack((x1,x2))
y = df.iloc[:,2]

plt.rc('font', size=20)
plt.rcParams['figure.constrained_layout.use'] = True
plt.title("original data visualization",fontsize=18,pad=20)
plt.scatter(x1_neg, x2_neg, color='red')
plt.scatter(x1_pos, x2_pos, color='yellow')
plt.xlabel('x1'); plt.ylabel('x2')
plt.legend(['-1','+1'],loc='center left', bbox_to_anchor=(1, 0.5),prop={'size': 10})
plt.savefig('original-data.png', facecolor='w', transparent=False)
plt.show()


#train test split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size = 0.2,stratify=y)
#cross-validation to select the maximum order of polynomial to use
mean_error=[]
std_error=[]
q_range = [1,2,3,4,5,6]
Xtrain = np.array(Xtrain)

for q in q_range:
    poly_features = PolynomialFeatures(q)
    poly_X_train = poly_features.fit_transform(Xtrain)
    model = LogisticRegression(penalty='l2');
```

```
    scores = cross_val_score(model,poly_X_train, ytrain, cv=5, scoring='f1')
    mean_error.append(np.array(scores).mean())
    std_error.append(np.array(scores).std())

plt.rc('font', size=18); plt.rcParams['figure.constrained_layout.use'] = True
plt.errorbar(q_range,mean_error,yerr=std_error,linewidth=3)
plt.title("number of features vs F1 score",fontsize=18,pad=20)
plt.xlabel('q'); plt.ylabel('F1 Score')
plt.savefig('tuningQ.png', facecolor='w', transparent=False)
plt.show()

#cross-validation to select the weight C given to the penalty in the cost function.
mean_error=[]
std_error=[]
Ci_range = [0.01, 0.1, 1, 5, 10, 25, 50, 100]
for Ci in Ci_range:
    model = LogisticRegression(C=Ci,penalty='l2');
    scores = cross_val_score(model,Xtrain, ytrain, cv=5, scoring='f1')
    mean_error.append(np.array(scores).mean())
    std_error.append(np.array(scores).std())
plt.rc('font', size=18); plt.rcParams['figure.constrained_layout.use'] = True
plt.errorbar(Ci_range,mean_error,yerr=std_error,linewidth=3)
plt.title("penalty parameter vs F1 score",fontsize=18,pad=20)
plt.xlabel('Ci'); plt.ylabel('F1 Score')
plt.savefig('tuningC.png', facecolor='w', transparent=False)
plt.show()

#training with logistic classifier
Xtrain = np.array(Xtrain)
poly_features = PolynomialFeatures(2)
poly_X_train = poly_features.fit_transform(Xtrain)
lr = LogisticRegression(penalty='l2',C=1,solver='lbfgs')
lr.fit(poly_X_train,ytrain)

#predict

Xtest = np.array(Xtest)
poly_features = PolynomialFeatures(2)
poly_X_test = poly_features.fit_transform(Xtest)
y_pred = lr.predict(poly_X_test)
print(confusion_matrix(ytest, y_pred))
print(classification_report(ytest, y_pred))
predict=np.column_stack([Xtest, y_pred])
df = pd.DataFrame(predict, columns = ['x1','x2','y'])
neg_pred = df[df.y==-1]
pos_pred = df[df.y==1]
x1_pred_neg = neg_pred.iloc[:,0]
```

```
x2_pred_neg = neg_pred.iloc[:,1]
x1_pred_pos = pos_pred.iloc[:,0]
x2_pred_pos = pos_pred.iloc[:,1]

plt.rc('font', size=20)
plt.rcParams['figure.constrained_layout.use'] = True
plt.title("Logistic with predicted vs actual",fontsize=18,pad=20)
plt.scatter(x1_neg, x2_neg, color='red')
plt.scatter(x1_pos, x2_pos, color='yellow')
plt.scatter(x1_pred_neg, x2_pred_neg, color='green',marker='+')
plt.scatter(x1_pred_pos, x2_pred_pos, color='cyan',marker='+')

plt.xlabel('x1'); plt.ylabel('x2')
plt.legend(['-1 actual','+1 actual','-1 predicted','+1 predicted'],loc='center left', bbox_to_anchor=(1,
0.5),prop={'size': 10  })
plt.savefig('logistic1.png', facecolor='w', transparent=False)
plt.show()
#cross-validation to select k
mean_error=[]
std_error=[]
k_range = [1,2,3,4,5,6,7,8,9,10]

for k in k_range:
   model = KNeighborsClassifier(n_neighbors=k,weights='uniform');
   scores = cross_val_score(model,Xtrain, ytrain, cv=5, scoring='f1')
   mean_error.append(np.array(scores).mean())
   std_error.append(np.array(scores).std())
plt.rc('font', size=18); plt.rcParams['figure.constrained_layout.use'] = True
plt.errorbar(k_range,mean_error,yerr=std_error,linewidth=3)
plt.title("k vs F1 score",fontsize=18,pad=20)
plt.xlabel('k'); plt.ylabel('F1 Score')
plt.savefig('tuningKNN.png', facecolor='w', transparent=False)
plt.show()
#training with KNN classifier
model = KNeighborsClassifier(n_neighbors=1,weights='uniform')
model.fit(Xtrain,ytrain)
#predict

y_pred1 = model.predict(Xtest)
print(confusion_matrix(ytest, y_pred1))
print(classification_report(ytest, y_pred1))
predict=np.column_stack([Xtest, y_pred1])
df = pd.DataFrame(predict, columns = ['x1','x2','y'])
neg_pred = df[df.y==-1]
pos_pred = df[df.y==1]
x1_pred_neg1 = neg_pred.iloc[:,0]
x2_pred_neg1 = neg_pred.iloc[:,1]
x1_pred_pos1 = pos_pred.iloc[:,0]
```

```
x2_pred_pos1 = pos_pred.iloc[:,1]

plt.rc('font', size=20)
plt.rcParams['figure.constrained_layout.use'] = True
plt.title("KNN with predicted vs actual",fontsize=18,pad=20)
plt.scatter(x1_neg, x2_neg, color='red')
plt.scatter(x1_pos, x2_pos, color='yellow')
plt.scatter(x1_pred_neg1, x2_pred_neg1, color='green',marker='+')
plt.scatter(x1_pred_pos1, x2_pred_pos1, color='cyan',marker='+')

plt.xlabel('x1'); plt.ylabel('x2')
plt.legend(['-1 actual','+1 actual','-1 predicted','+1 predicted'],loc='center left', bbox_to_anchor=(1,
0.5),prop={'size': 10  })
plt.savefig('KNN.png', facecolor='w', transparent=False)
plt.show()

dummy = DummyClassifier(strategy='most_frequent').fit(Xtrain, ytrain)
ydummy = dummy.predict(Xtest)
print(confusion_matrix(ytest, ydummy))
print(classification_report(ytest, ydummy))

plt.rc('font', size=18); plt.rcParams['figure.constrained_layout.use'] = True

fpr, tpr, _ = roc_curve(ytest,lr.decision_function(poly_X_test))
plt.plot(fpr,tpr)
fpr1, tpr1, _ = roc_curve(ytest,y_pred1)
plt.plot(fpr1,tpr1)
fpr2, tpr2, _ = roc_curve(ytest,ydummy)
plt.title("ROC curve",fontsize=18,pad=20)
plt.plot(fpr2,tpr2,color='green',linestyle='--')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.legend(["logistic", "KNN", "dummy"])
plt.savefig('ROC.png', facecolor='w', transparent=False)
plt.show()
```