Machine Learning - Week 4 Assignment
Mark Lysaght - 19334391

i) a)

Figure 1 shows a plot of the raw data from dataset #id:24-24-24-1. The data points seem random and there does not look to be a shape to the target values.

The points with target value y=+1 are marked by a green '+'. The points with target value y=-1 are marked by a blue 'o', as can be seen from the legend. On the x axis are the X1 features (first column of dataset) and on the y axis are the X2 features (second column of dataset).

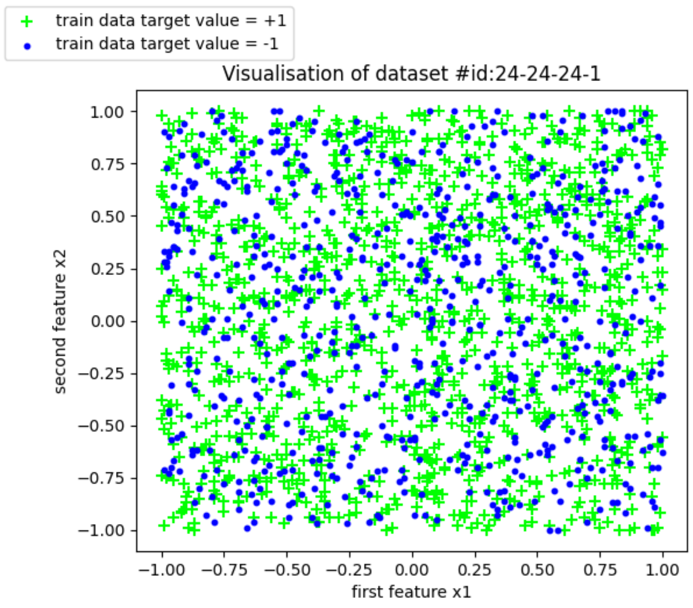


Figure 1. Visualisation of raw data from the first dataset.

i) Using sklearns PolynomialFeatures function, I augmented the two features of the dataset and trained a logistic regression classifier with L2 penalty to them. I used 5-fold cross validation to determine the maximum order polynomial to use. I did this by picking a range of polynomials q=[1,2,3,4,5,6] and fitting the original dataset to obtain a new model. The f1 error was plotted to allow for analysis of the optimal q. Note that the closer an f1 score is to 1 indicates that the model more accurately fits the data.

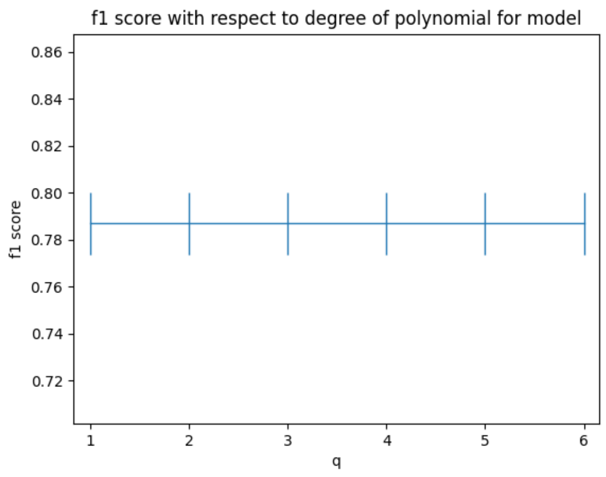


Figure 2. F1 score with respect to increasing complexity of feature model.

As can be seen in Figure 2, there appears to be no change in the f1 score for varying values of degree of polynomial and so for the maximum order polynomial for the logistic regression model, I chose q=1.
ii) 5-fold cross validation was applied again to determine the optimal weight for C, the penalty applied to the cost function. A similar approach as when determining the maximum order polynomial was used in that I picked a range of values for C, C=[0.01,0.05,0.1,0.5,1,5,10,50] and fit the logistic regression model with varying penalty. Once again we see apparently no change in the f1 score from a large range of values for C so I picked C=1.

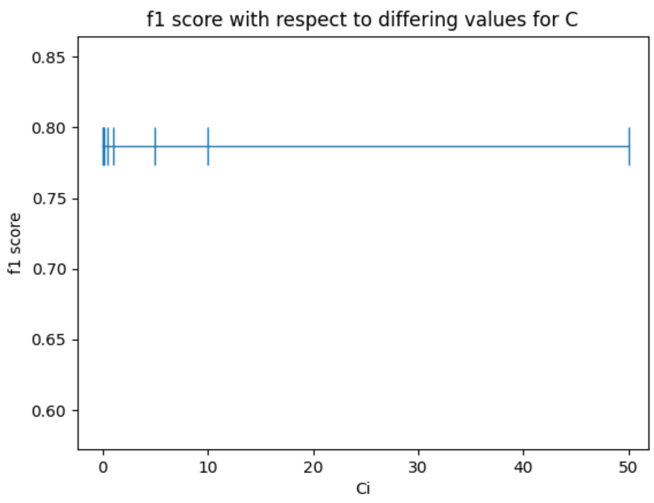


Figure 3. F1 score with respect to increasing value for penalty term C.

Now that we have picked a suitable degree polynomial and penalty term, we can plot the predictions from our model. On the plot below note that we see significantly fewer data points than the plot from just the raw data. This is because the test data predicted from the training data is only a fifth the size of the whole dataset for each fold of our cross validation.

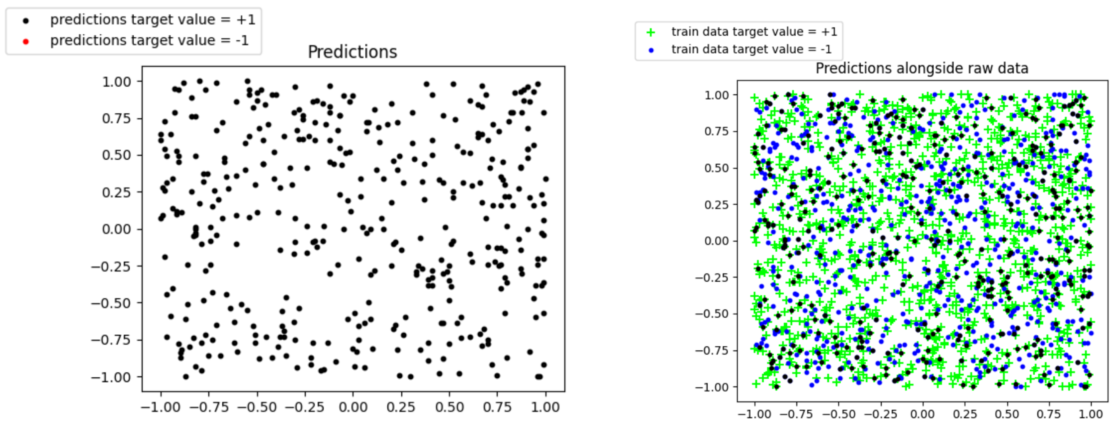


Figure 4. a) Predictions from logistic regression model. b) Predictions over raw data.

Figure 4 shows us that the logistic regression model predicted all of the data points to have target value=+1. This makes sense because we can see above in Figure 1 that the data appears random and so it is hard to predict the outcome from random data.

b) I trained a kNN classifier on the data. Once again using cross validation to select an accurate k to use. I plotted the root mean squared error (RMSE) against varying values of k to see the minimum RMSE. I started with a range of k = 5 to 200 with increments of 10 at first to see a general trend.

The smaller the value of the RMSE means that there is less error in our model.

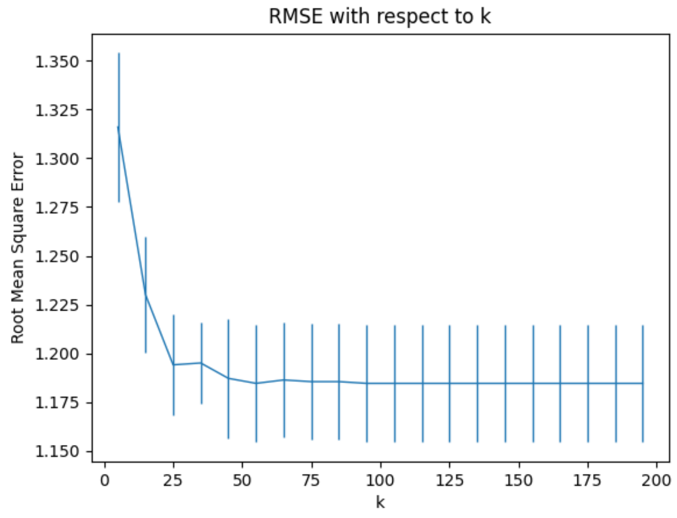


Figure 5. Root mean squared error with respect to increasing k.

We can see in figure 5 that the RMSE levels off at around k=50 so this was the value I chose for k. Then I plot the predictions of the kNN classifier with n_neighbours=50 found through 5-fold cross validation once again.

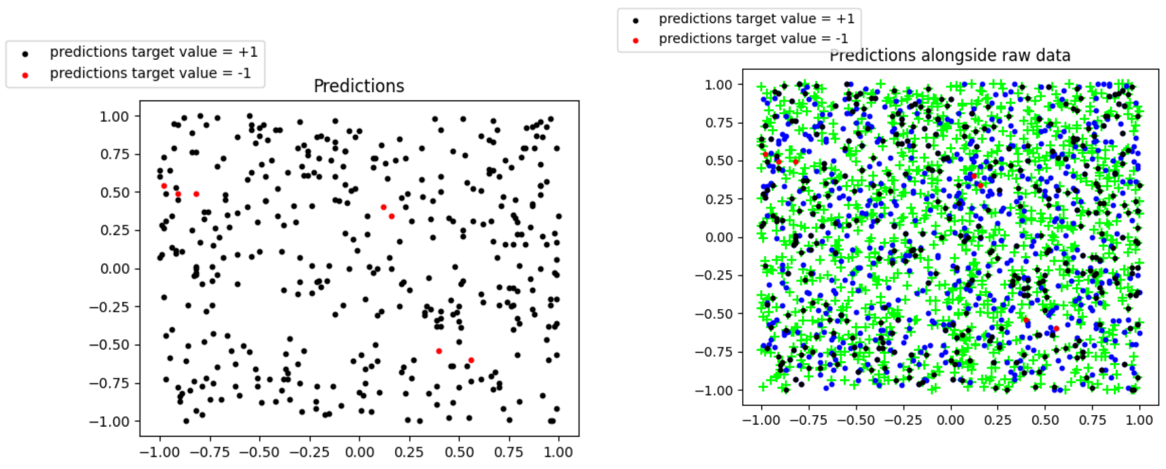
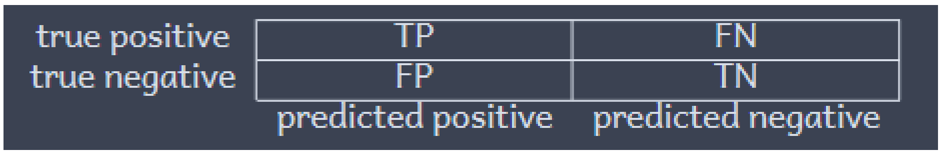


Figure 6. a) Predictions from kNN model. b) Predictions over raw data.

c) The confusion matrix was calculated for both the logistic regression classifier from part a) and the kNN classifier from part b). The confusion matrix is used to check the accuracy of predictions. On the diagonals we can see the validity of the predictions.



true positive	TP	FN
true negative	FP	TN
	predicted positive	predicted negative

Figure 7. Confusion matrix template for reference

After running confusion_matrix(ytest,preds) on the Logistic regression classifier I got the confusion matrix:

0	141

0	253

We can attribute this to our data seen in Figure 4 as there are no positive values seen on our graph. We have a large amount of false predictions according to our matrix so this shows us that logistic regression is not a good classifier for our dataset.

I ran the same function but with the kNN classifier to get the confusion matrix:

2	139
5	248

This time, our matrix shows us that we have some predicted positive values, but more than double of them are false positive values. Once again this shows that kNN classifier is not a good classifier to apply to our dataset.

Confusion matrices were then calculated for two baseline models. The first being a dummy classifier using the “most frequent” target value. This will predict all values to be that of the target value of majority in our dataset. For this model I got the confusion matrix:

0	141
0	253

The results of this dummy model are not surprising.

Confusion matrix for the second baseline model using the “stratified” strategy. This strategy picks points randomly from our dataset so the results should be better than that of any of the seen before models due to the dataset in itself appearing random. We get the confusion matrix:

57	84
82	171

The results are much better than the models before as the values seem more evenly spread but there are still a lot more false predictions than enough to deem this a suitable classifier of our dataset. However, it is the best of a bad bunch.

d) I plotted ROC curves for each of the classifiers used in part c). An ROC curve is a graphical way of illustrating how accurate predictions are for a certain classifier. The closer the graph is to the top left, the more accurate the classifier is on your particular dataset.Vice-versa for the bottom right.

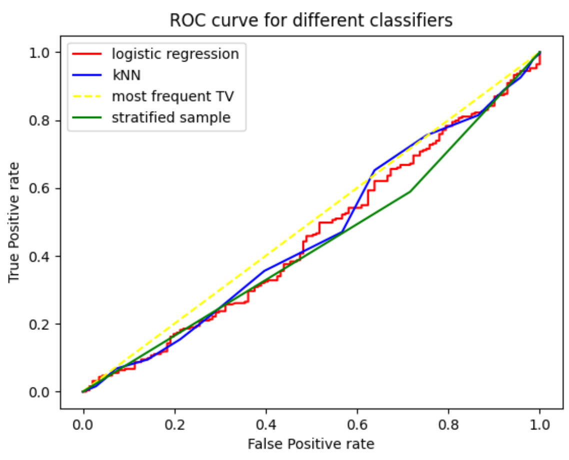


Figure 8. ROC curves

We can see that all of our classifiers perform quite poorly to our dataset which is not surprising given the random nature of the dataset.

e) As mentioned in part d) all of the classifiers seemed to perform quite poorly on this dataset and I would not think any of the classifiers would be suitable to deploy to provide an accurate machine learning model. The only classifier which seemed to be able to predict any meaningful data was the “stratified sample” classifier which purely relied on random training data. This is not a good thing because for future generations of using this classifier, the data that is predicted is unpredictable by nature. From figure 7, the only classifier that veers into the true positive side of the graph is the kNN classifier, so if I was forced to give a recommendation I would reluctantly say the kNN model.

ii) a) Figure 9 shows a plot of the raw data from dataset #id:24-48-24-1. The data points seem to shape to a quadratic curve.

The points with target value y=+1 are marked by a green ‘+’. The points with target value y=-1 are marked by a blue ‘o’, as can be seen from the legend. On the x axis are the X1 features (first column of dataset) and on the y axis are the X2 features (second column of dataset).

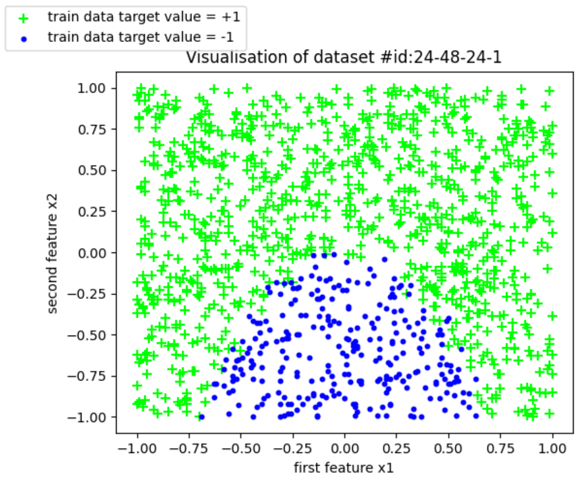


Figure 9. Visualisation of raw data from the second dataset.

i) Using sklearns PolynomialFeatures function, I augmented the two features of the dataset and trained a logistic regression classifier with L2 penalty to them. I used 5-fold cross validation to determine the maximum order polynomial to use. I did this by picking a range of polynomials q=[1,2,3,4,5,6] and fitting the original dataset to obtain a new model. The f1 error was plotted to allow for analysis of the optimal q. Note that the closer an f1 score is to 1 indicates that the model more accurately fits the data.

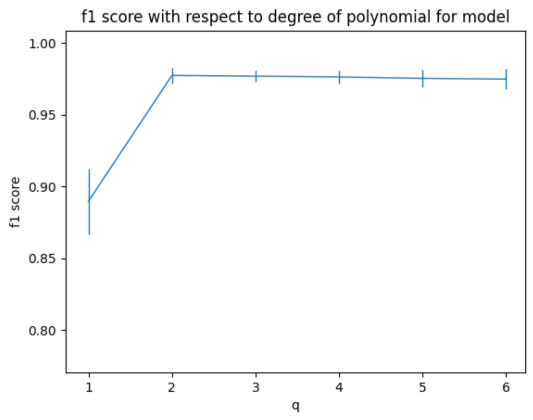


Figure 10. F1 score with respect to increasing complexity of feature model.

As can be seen in Figure 10, there appears to be a significant change in the f1 score from q=1 to q=2. This makes sense, given the plot of the data in figure 8 resembling a quadratic curve. For this reason, I chose q=2.
ii) 5-fold cross validation was applied again to determine the optimal weight for C, the penalty applied to the cost function. A similar approach as when determining the maximum order polynomial was used in that I picked a range of values for C, C=[0.01,0.05,0.1,0.5,1,5,10,50] and fit the logistic regression model with varying penalty. The f1 score tends very close to 1 for values greater than 5 so for this reason, I chose C=5 for the penalty.

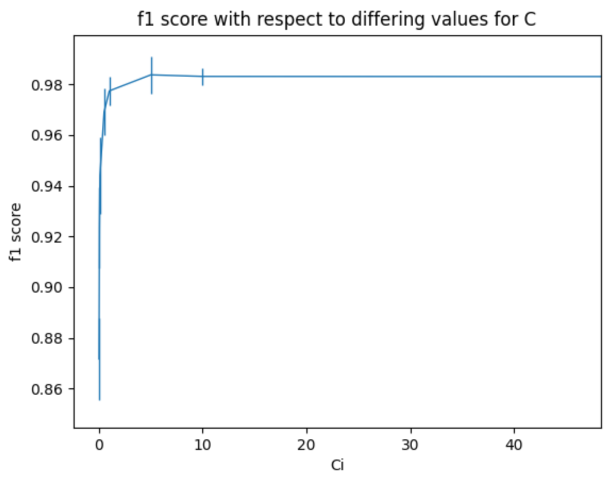


Figure 11. F1 score with respect to increasing weight of penalty C.

Now that we have picked a suitable degree polynomial and penalty term, we can plot the predictions from our model. On the plot below note that we see significantly fewer data points than the plot from just the raw data. This is because the test data predicted from the training data is only a fifth the size of the whole dataset for each fold of our cross validation.

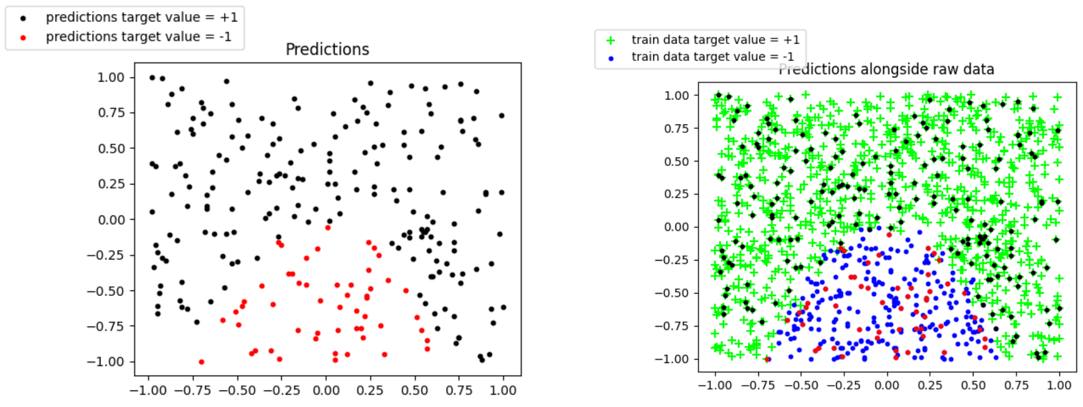


Figure 12. a) Predictions from logistic regression model. b) Predictions over raw data.

Figure 12 a) indicates to us that the logistic regression model predicted the data points quite accurately as the shape of a quadratic curve is still to be seen in our plot. When we plot it against the original data in b) we see clearer how the predictions fit to the original data.

b) I trained a kNN classifier on the data. Once again using cross validation to select an accurate k to use. I plotted the root mean squared error (RMSE) against varying values of k to see the minimum RMSE. I started with a range of k = 5 to 50 with increments of 2.

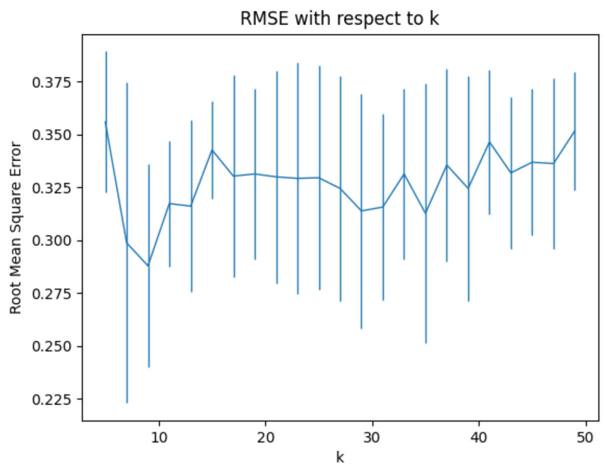


Figure 12. RMSE with respect to k

We see from figure 12 that the RMSE reaches a minimum when k=9 so for this reason, I chose k=9. Then I plot the predictions of the kNN classifier with n_neighbours=9 found through 5-fold cross validation once again.

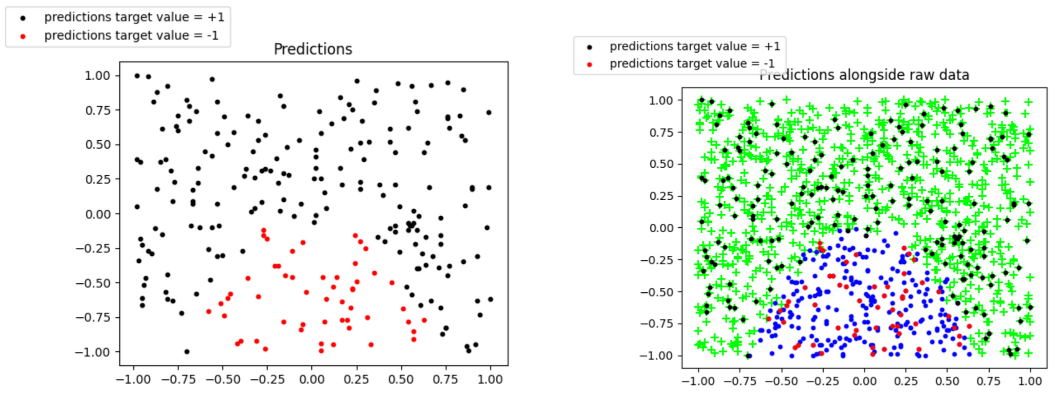


Figure 13. a) Predictions from kNN model. b) Predictions over raw data.
Figure 13 a) indicates to us that the kNN predicted the data points quite accurately as the shape of a quadratic curve is still to be seen in our plot. When we plot it against the original data in b) we see clearer how the predictions fit to the original data.

c) The confusion matrix was calculated for both the logistic regression classifier from part a) and the kNN classifier from part b). The confusion matrix is used to check the accuracy of predictions. See figure 7.

After running confusion_matrix(ytest,preds) on the Logistic regression classifier I got the confusion matrix:

49	3
4	188

This confusion matrix is very promising as there are very few false predictions. We can determine from this that our logistic regression classifier accurately fits our dataset.

I ran the same function but with the kNN classifier to get the confusion matrix:

50	2
4	188

The kNN confusion matrix is also a very accurate classifier for our dataset producing very similar true and false predictions.

Confusion matrices were then calculated for two baseline models. The first being a dummy classifier using the “most frequent” target value. This will predict all values to be that of the target value of majority in our dataset. For this model I got the confusion matrix:

0	52
0	192

The results of this dummy model do not indicate anything about our data. It is just used for a reference.

Confusion matrix for the second baseline model using the “stratified” strategy. This strategy picks points randomly from our dataset. We get the confusion matrix:

13	39
39	153

Once again these results are just a baseline to plot against our more accurate models.

d) I plotted ROC curves for each of the classifiers used in part c). An ROC curve is a graphical way of illustrating how accurate predictions are for a certain classifier. The closer the graph is to the top left, the more accurate the classifier is on your particular dataset.Vice-versa for the bottom right.

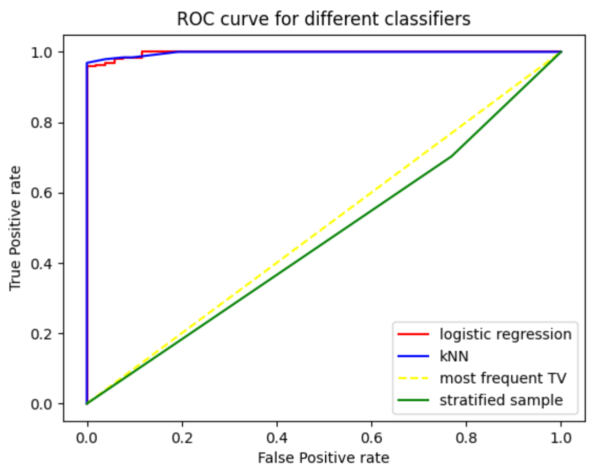


Figure 14. ROC curves

We can see from figure 14 that both the logistic regression and the kNN classifiers perform very well for predicting true positive values for our dataset.

e) Both the logistic regression and kNN classifiers are very accurate classifiers for this dataset. They performed much better than the baseline dummy models. The kNN and logistic regression classifiers both closely resemble perfect classifiers and this can be seen in our confusion matrices through their very few false predictions. From looking at figure 14, the kNN classifier performs slightly better than the logistic regression model so I would recommend the kNN for this dataset.

APPENDIX FOR CODE

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error
from sklearn.metrics import f1_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.dummy import DummyClassifier
from sklearn.metrics import roc_curve

def PartA(df, id):
    X1 = df.iloc[:, 0]
    X2 = df.iloc[:, 1]
    X = np.column_stack((X1, X2))
    y = df.iloc[:, 2]
    plotRawData(X, y, id)
    kf = KFold(n_splits=5)
    q_range = [1, 2, 3, 4, 5, 6]
    aLogRegChoosePolynomial(kf, X, y, q_range)
    Ci_range = [0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50]
    aLogRegChooseC(kf, X, y, Ci_range)
    if id == 'id:24-24-24-1':
        q = 1
        c = 1
        k = 50
        aLogRegPlotPred(kf, X, y, q, c)
        k_range = range(5, 200, 10)
        bKnnChoosek(kf, X, y, k_range)
        bKnnPlotPred(kf, X, y, k)
        cBaseline(kf, X, y)
        dPlotROC(kf, X, y, q, c, k)
    elif id == 'id:24-48-24-1':
        q = 2
        c = 5
        k = 9
        aLogRegPlotPred(kf, X, y, q, c)
        k_range = range(5, 50, 2)
        bKnnChoosek(kf, X, y, k_range)
        bKnnPlotPred(kf, X, y, c)
        cBaseline(kf, X, y)
        dPlotROC(kf, X, y, q, c, k)

def plotRawData(X, y, id):
    plt.title("Visualisation of dataset #" + id)
    plt.scatter(*X[y == 1].T, s=50, marker='+',
                label="train data target value = +1", c='lime')
    plt.scatter(*X[y == -1].T, s=10, marker='o',
                label="train data target value = -1", c='b')
    plt.legend(bbox_to_anchor=(0.3, 1.2))
    plt.xlabel("first feature x1")
    plt.ylabel("second feature x2")
    plt.show()

def aLogRegChoosePolynomial(kf, X, y, q_range):
    f1 = []
    std_error = []
    model = LogisticRegression(penalty='l2', C=1)
    for q in q_range:
        Xpoly = PolynomialFeatures(q).fit_transform(X)
        temp = []
        for train, test in kf.split(Xpoly):
            model.fit(Xpoly[train], y[train])
            ypred = model.predict(Xpoly[test])
            temp.append(f1_score(y[test], ypred))
        f1.append(np.array(temp).mean())
        std_error.append(np.array(temp).std())
    plt.errorbar(q_range, f1, yerr=std_error, linewidth=1)
    plt.ylim(0.7, 1)
    plt.xlabel('q')
    plt.ylabel('f1 score')
    plt.title('f1 score with respect to degree of polynomial for model')
    plt.show()

def aLogRegChooseC(kf, X, y, Ci_range):
    f1 = []
    std_error = []
    Xpoly = PolynomialFeatures(2).fit_transform(X)
    for Ci in Ci_range:
        model = LogisticRegression(penalty='l2', C=Ci)
        temp = []
        for train, test in kf.split(Xpoly):
            model.fit(Xpoly[train], y[train])
            ypred = model.predict(Xpoly[test])
            temp.append(f1_score(y[test], ypred))
        f1.append(np.array(temp).mean())
        std_error.append(np.array(temp).std())
    plt.errorbar(Ci_range, f1, yerr=std_error, linewidth=1)
    plt.xlabel('Ci')
    plt.ylabel('f1 score')
    plt.ylim(0.5, 1)
    plt.title('f1 score with respect to differing values for C')
    plt.show()


def aLogRegPlotPred(kf, X, y, q, c):
    model = LogisticRegression(penalty='l2', C=c)
    Xpoly = PolynomialFeatures(q).fit_transform(X)
    for train, test in kf.split(Xpoly):
        model.fit(Xpoly[train], y[train])
        ypred = model.predict(Xpoly[test])
    print(model.intercept_)
    print(model.coef_)
    plt.scatter(*X[test][ypred == 1].T, color='black', s=10,
                label='predictions target value = +1')
    plt.scatter(*X[test][ypred == -1].T, color='red', s=10,
                label='predictions target value = -1')
    plt.legend(bbox_to_anchor=(0.3, 1.2))
    plt.title('Predictions')
    plt.show()
    plt.scatter(*X[y == 1].T, s=50, marker='+',
                label="train data target value = +1", c='lime')
    plt.scatter(*X[y == -1].T, s=10, marker='o',
                label="train data target value = -1", c='b')
    plt.scatter(*X[test][ypred == 1].T, color='black', s=10)
    plt.scatter(*X[test][ypred == -1].T, color='red', s=10)
    plt.legend(bbox_to_anchor=(0.3, 1.2))
    plt.title('Predictions alongside raw data')
    plt.show()
    cConfusionMatrix(y[test], ypred)

def bKnnChoosek(kf, X, y, k_range):
    mean_error = []
    std_error = []
    for k in k_range:
        temp = []
        model = KNeighborsClassifier(n_neighbors=k, weights='uniform')
        for train, test in kf.split(X):
            model.fit(X[train], y[train])
            ypred = model.predict(X[test])
            temp.append(pow(mean_squared_error(y[test], ypred), 0.5))
        mean_error.append(np.array(temp).mean())
        std_error.append(np.array(temp).std())
    plt.errorbar(k_range, mean_error, yerr=std_error, linewidth=1)
    plt.xlabel('k')
    plt.ylabel('Root Mean Square Error')
    plt.title('RMSE with respect to k')
    plt.show()

def bKnnPlotPred(kf, X, y, k):
    model = KNeighborsClassifier(n_neighbors=k, weights='uniform')
    for train, test in kf.split(X):
        model.fit(X[train], y[train])
        ypred = model.predict(X[test])
    plt.scatter(*X[test][ypred == 1].T, color='black',s=10, label='predictions target value
= +1')
    plt.scatter(*X[test][ypred == -1].T, color='red', s=10,
                label='predictions target value = -1')
    plt.legend(bbox_to_anchor=(0.3, 1.2))
    plt.title('Predictions')
    plt.show()
    plt.scatter(*X[y == 1].T, s=50, marker='+',
                c='lime')
    plt.scatter(*X[y == -1].T, s=10, marker='o',
                c='b')
    plt.scatter(*X[test][ypred == 1].T, color='black',
                s=10, label='predictions target value = +1')
    plt.scatter(*X[test][ypred == -1].T, color='red', s=10,
```

```python
                label='predictions target value = -1')
    plt.legend(bbox_to_anchor=(0.3, 1.2))
    plt.title('Predictions alongside raw data')
    plt.show()
    cConfusionMatrix(y[test], ypred)

def cConfusionMatrix(ytest, preds):
    print(confusion_matrix(ytest, preds))
    pass

def cBaseline(kf, X, y):
    mf = DummyClassifier(strategy="most_frequent")
    st = DummyClassifier(strategy="stratified")
    for train, test in kf.split(X):
        mf.fit(X[train], y[train])
        ypredmf = mf.predict(X[test])
        st.fit(X[train], y[train])
        ypredst = st.predict(X[test])
    cConfusionMatrix(y[test], ypredmf)
    cConfusionMatrix(y[test], ypredst)

def dPlotROC(kf, X, y, q, c, k):
    lr = LogisticRegression(penalty='l2', C=c)
    Xpoly = PolynomialFeatures(q).fit_transform(X)
    knn = KNeighborsClassifier(n_neighbors=k, weights='uniform')
    mf = DummyClassifier(strategy="most_frequent")
    st = DummyClassifier(strategy="stratified")
    for train, test in kf.split(X):
        lr.fit(Xpoly[train], y[train])
        knn.fit(X[train], y[train])
        mf.fit(X[train], y[train])
        st.fit(X[train], y[train])
    fprlr, tprlr, _ = roc_curve(y[test], lr.decision_function(Xpoly[test]))
    fprknn, tprknn, _ = roc_curve(y[test], knn.predict_proba(X[test])[:, 1])
    fprmf, tprmf, _ = roc_curve(y[test], mf.predict_proba(X[test])[:, 1])
    fprst, tprst, _ = roc_curve(y[test], st.predict_proba(X[test])[:, 1])
    plt.plot(fprlr, tprlr, color='red', label='logistic regression')
    plt.plot(fprknn, tprknn, color='blue', label='kNN')
    plt.plot(fprmf,tprmf,color='yellow',linestyle='dashed', label='most frequent TV')
    plt.plot(fprst, tprst, c='g', label='stratified sample')
    plt.legend()
    plt.ylabel('True Positive rate')
    plt.xlabel('False Positive rate')
    plt.title('ROC curve for different classifiers')
    plt.show()

# id:24-24-24-1
df = pd.read_csv("week4_1.csv", header=None)
PartA(df, 'id:24-24-24-1')
# id:24-48-24-1
df = pd.read_csv("week4_2.csv", header=None)
PartA(df, 'id:24-48-24-1')
```