Week 2 Assignment

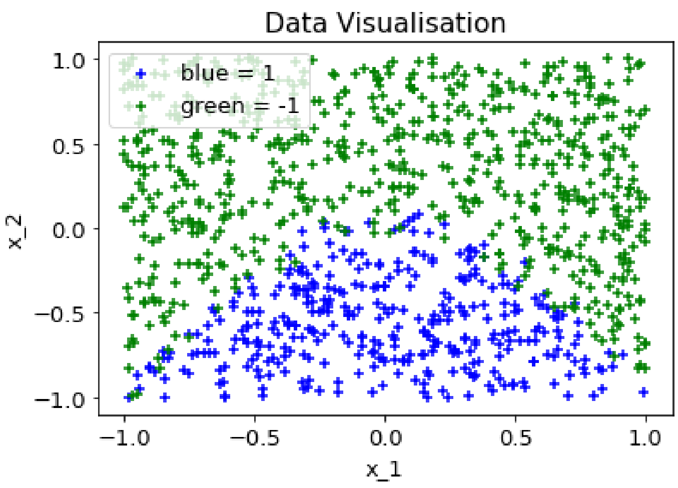
CS7CS4/CSU44061 Machine Learning

Cormac Madden – 18319983

Dataset ID: # id:18--18--18

a) i)

The data was read in using the code provided in the assignment and the graph was drawn as shown in Figure 1. (See appendix for code.)



a) ii)

The sklearn function was used to train a logistic regression model of the data.

It calculates the parameter values required to create a decision boundary line.

The logistic regression model for predictions is as follows:

$$\theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

And in this case with two features:

$$\theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Where x_1 and x_2 are the input features and θ_n are the parameters.

The resultant parameters were:

θ_0 = -1.7325 (intercept)

θ_1 = -0.2692 (slope parameters)

θ_2 = -5.5384

$\theta^T x$ = -1.7325 - 0.2692 x_1 - 5.5384 x_2

The prediction works by finding the sign of $\theta^T x$

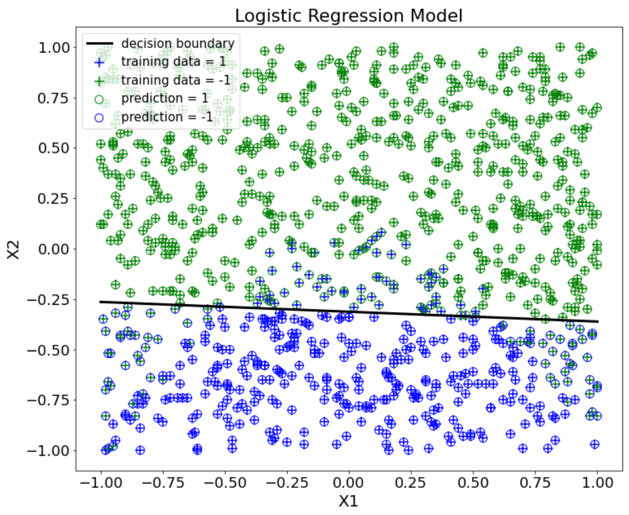
Sign($\theta^T x$) = 1 when $\theta^T x > 0$

Sign($\theta^T x$) = -1 when $\theta^T x < 0$

The x_2 feature has the **most influence** on the prediction because its associated parameter is the largest (θ_2 = -5.5384).

An increase in x_1 or in x_2 causes the prediction to decrease and a decrease in either causes the prediction model to increase. $\theta^T x$ = -1.7325 - 0.2692 x_1 - 5.5384 x_2

iii)



The decision boundary of the logistic regression model is the line at which the probability of the prediction being -1 or 1 is 0.5 .

Where $P(y = 1|x) = P(y = -1|x) = 0.5$

Given:

$$P(y = 1|x) = \frac{1}{1 + e^{-\theta^t x}}$$

Let equal 0.5

$$\frac{1}{1 + e^{-\theta^t x}} = \frac{1}{2}$$

$$1 + e^{-\theta^t x} = 2$$

$$e^{-\theta^t x} = 1$$

$$\theta^t x = 0$$

Substituting this into our model we can calculate the y values of the decision boundary

$$\theta^T \mathrm{x} = \theta_0 + \theta_1 \mathrm{x}_1 + \theta_2 \mathrm{x}_2 = 0$$

$$\mathrm{x}_2 = -\frac{\theta_1 \mathrm{x}_1}{\theta_2} - \frac{\theta_0}{\theta_2}$$

The code to implement the decision boundary is as follows.

```
line_y = -1*model.coef_[0,0]/model.coef_[0,1] * line_x + -model.intercept_ / model.coef_[0,1]
```

a) iv) The prediction data is plotted against the training data as shown in Figure 2.

It's clear that the two classes (y = 1, y = -1) are not linearly separable, and therefore a linear decision boundary is not suitable for predicting this data.

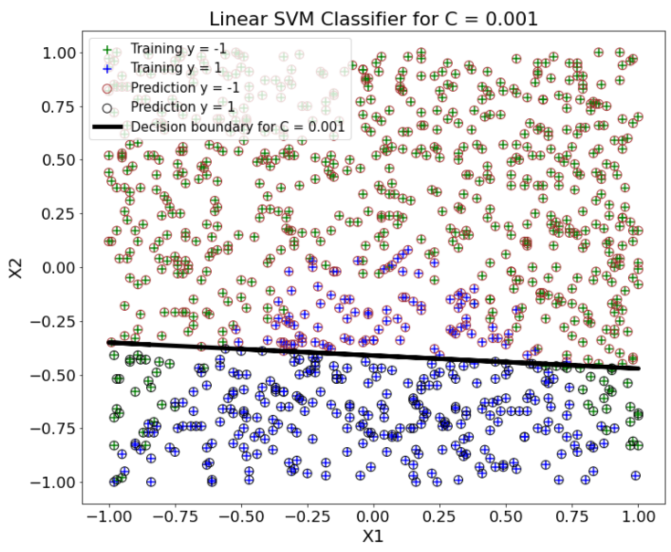
The graph shows a high number of false positives and false negatives, illustrated by the blue crosses inside green circles, and the green crosses inside blue circles, respectively.

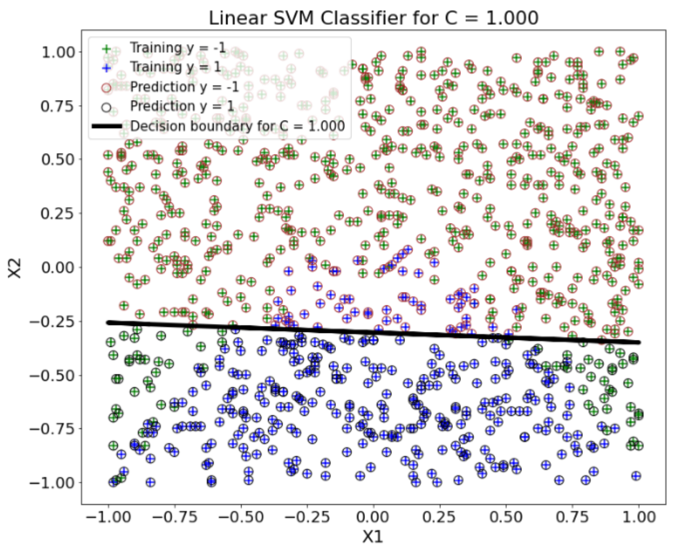
b) i)

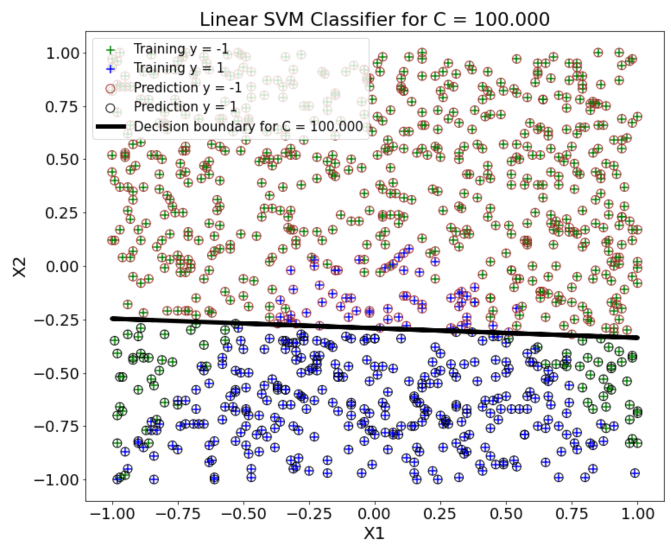
A linear SVM model was trained on the data using the following parameters: C = 0.001, C = 1, C = 100 and C = 1000

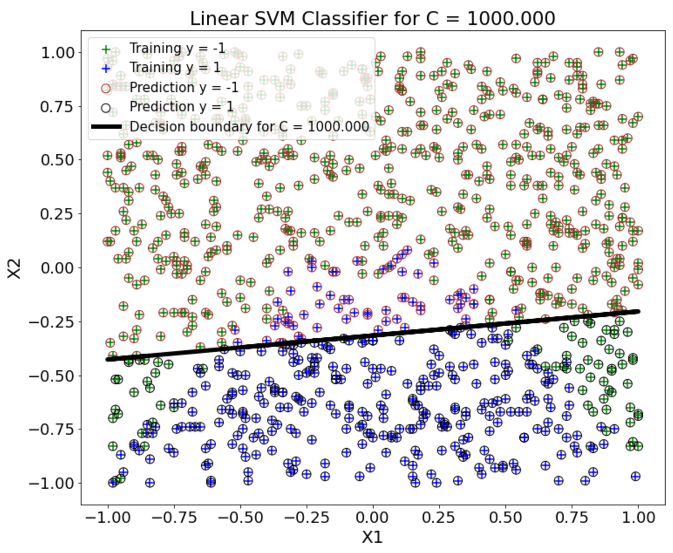
C	θ_0 (Intercept)	θ_1	θ_2
0.001	-0.20	-0.029	-0.48
1	-0.55	-0.08	-1.82
100	-0.6	-0.08	-1.82
1000	-0.17	-0.75	-2.31

b) ii)









b) iii)

Decreasing C gives the penalty more importance. This results in SVM choosing smaller model parameters θ, as seen in the table.

It can lead to a higher bias, but lower variance given the smaller model parameters.

Increasing C gives the penalty less importance, resulting in larger parameters.

$$J(\vartheta) = \frac{1}{m}\sum_{i=1}^{m} \max(0, 1 - y^i\theta^T x^i) + \frac{\vartheta^T\vartheta}{C}$$

b) iv)

The SVM model parameters and predictions are similar to that of the logistic regression model in part a. They are both limited by the linear boundary. There is variation in the accuracy but neither models reliably classify the data.

c) i)

```
X3 = X1**2
X4 = X2**2
X = np.column_stack((X1,X2,X3,X4))
model = LogisticRegression().fit(X, y)
```

The trained parameter values were:

θ_0 = -0.05559401

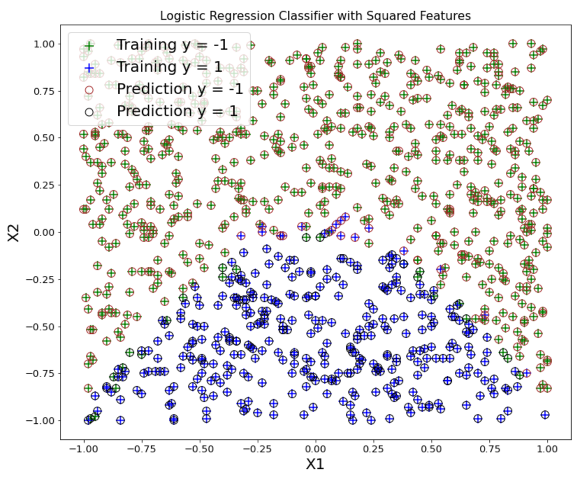
θ_1 = 0.01781205

θ_2 = -6.63393089

θ_3 = -6.74017505

θ_3 = 0.8710698

ii)



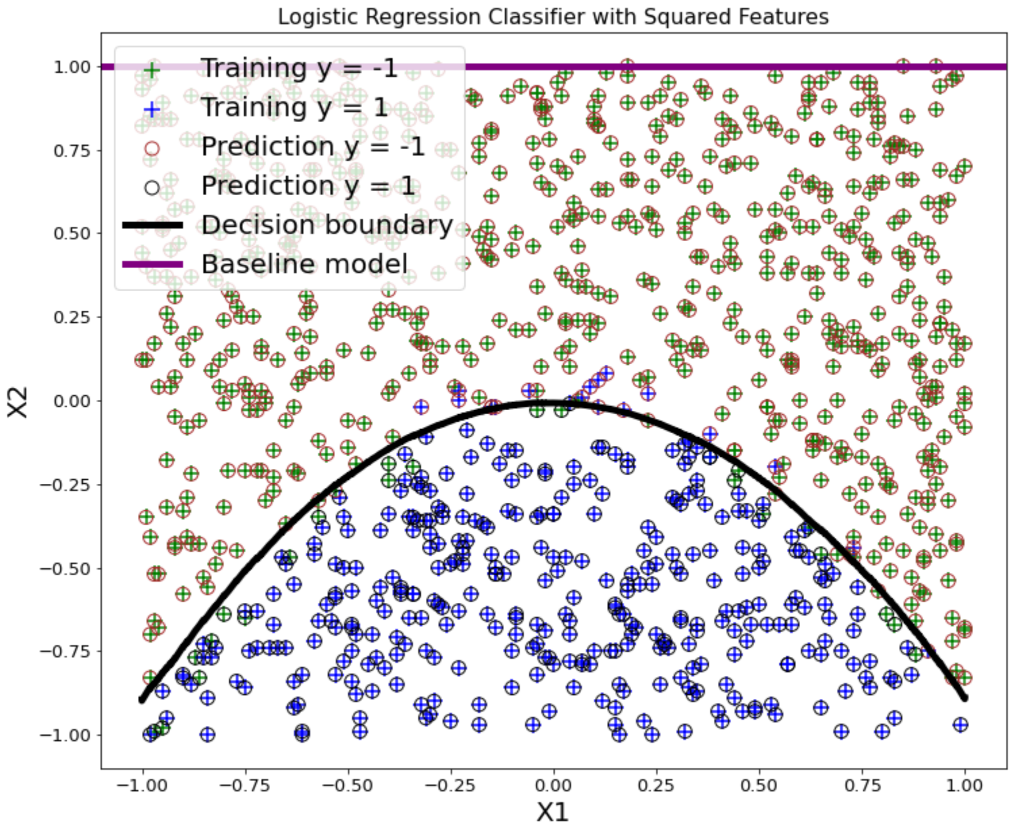
The predictions using this classifier are significantly more accurate than that used in part a) and b).

This is because the decision boundary is a quadratic curve as opposed to a straight line.

iii)

A classifier that always predicts the most common class predicted the training data correct 64% of the time, whereas the new classifier predicted the training data correct 99.4% of the time. The new classifier is potentially 1.55 times better than the basic classifier.

iv)



Appendix

```python
# %%
import numpy as np
import math
from matplotlib import cm
import pandas as pd
import scipy as sp
import seaborn as sn
import matplotlib.pyplot as plt
import matplotlib as matlib
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC

df = pd.read_csv('week2.csv')
df.columns = ["X1","X2","y"]
X1=df.iloc[:,0]
X2=df.iloc[:,1]
X=np.column_stack((X1,X2))
y=df.iloc[:,2]
df.isnull();

plt.rc('font', size =13)
plt.xlabel("x_1")
plt.ylabel("x_2")
plt.title("Data Visualisation")
plt.scatter(X[y > 0, 0], X[y > 0, 1], marker = "+",c='b')
plt.scatter(X[y < 0, 0], X[y < 0, 1], marker = "+", c='g')
plt.legend(["blue = 1","green = -1"], fancybox = True)

# %%
#ii)
model = LogisticRegression(penalty="none", solver='lbfgs').fit(X,y)

# %%
#iii)
line_x = np.array([-1,1])
line_y = -1*model.coef_[0,0]/model.coef_[0,1] * line_x + -model.intercept_ /
model.coef_[0,1]

plt.figure(figsize=(12,10), dpi=72)
plt.rcParams.update({'font.size': 18})
plt.plot(line_x,line_y, linewidth =3, color = "black")
plt.scatter(X[y > 0, 0], X[y > 0, 1], marker = "+", s=150,c='b')
plt.scatter(X[y < 0, 0], X[y < 0, 1], marker = "+",s=150,c='g')
X_pred = X[np.where(model.predict(X) == -1)]
X_pred = X[np.where(model.predict(X) == 1)]
plt.scatter(X_pred[:, 0], X_pred[:, 1], marker='o', facecolor='none',
edgecolor='g', s=100, label="Prediction y = -1")
plt.scatter(X_pred[:, 0], X_pred[:, 1], marker='o', facecolor='none',
edgecolor='b',s=100, label="Prediction y = 1")
plt.legend(["decision boundary","training data = 1","training data = -
1","prediction = 1","prediction = -1"], loc = "upper left", fontsize = 15)
plt.title("Logistic Regression Model")
plt.xlabel("X1", fontsize=20)
plt.ylabel("X2", fontsize=20)
```

```
# %%
# (b)(i)
X_m1 = X[np.where(y == -1)]
X_p1 = X[np.where(y == 1)]

C = [0.001, 1, 100, 1000]

for Ci in C:
    modelSVC = LinearSVC(C=Ci).fit(X, y)
    theta = [modelSVC.intercept_[0], modelSVC.coef_[0][0], modelSVC.coef_[0][1]]
    print (theta)

    plt.figure(figsize=(12,10))
    plt.rcParams.update({'font.size': 18})

    # Training data
    plt.scatter(X_m1[:, 0], X_m1[:, 1], c='g', marker='+', s=100, label="Training
y = -1")
    plt.scatter(X_p1[:, 0], X_p1[:, 1], c='b', marker='+', s=100, label="Training
y = 1")

    # Predictions
    X_pred = X[np.where(modelSVC.predict(X) == -1)]
    X_pred = X[np.where(modelSVC.predict(X) == 1)]
    plt.scatter(X_pred[:, 0], X_pred[:, 1], marker='o', facecolor='none',
edgecolor='brown', s=100, label="Prediction y = -1")
    plt.scatter(X_pred[:, 0], X_pred[:, 1], marker='o', facecolor='none',
edgecolor='black', s=100, label="Prediction y = 1")

    # Decision boundary
    plt.plot(X1, -1/theta[2] * (theta[0] + theta[1]*X1), linewidth=5, c='black',
label="Decision boundary for C = %.3f"%Ci)

    plt.legend(scatterpoints=1, fontsize=15,loc = "upper left")
    plt.title("Linear SVM Classifier for C = %.3f"%Ci)
    plt.xlabel("X1", fontsize=20)
    plt.ylabel("X2", fontsize=20)
    plt.show()

# %%
#(c)(i)
# Additional features
X3 = X1**2
X4 = X2**2
X = np.column_stack((X1,X2,X3,X4))
model3 = LogisticRegression().fit(X, y)

#(c)(ii)
plt.figure(figsize=(12,10), dpi=72)
plt.rcParams.update({'font.size': 13})

# Training
plt.scatter(X_m1[:, 0], X_m1[:, 1], c='g', marker='+', s=150, label="Training y =
-1")
plt.scatter(X_p1[:, 0], X_p1[:, 1], c='b', marker='+', s=150, label="Training y =
1")

# Predictions
X_pred = X[np.where(model3.predict(X) == -1)]
X_pred = X[np.where(model3.predict(X) == 1)]
plt.scatter(X_pred[:, 0], X_pred[:, 1], marker='o', facecolor='none',
edgecolor='brown', s=100, label="Prediction y = -1")
plt.scatter(X_pred[:, 0], X_pred[:, 1], marker='o', facecolor='none',
edgecolor='black', s=100, label="Prediction y = 1")

# Baseline
(values,counts) = np.unique(y, return_counts=True)
mostCommonValue = values[np.argmax(counts)]

# (c)(iv)
# Decision boundary
X1_sorted = np.sort(X1)
X2_sorted = np.sort(X2)
a = -model3.coef_[0][2]/model3.coef_[0][1]
b = -model3.coef_[0][0]/model3.coef_[0][1]
c = -model3.intercept_[0]/model3.coef_[0][1] - model3.coef_[0][3] /
model3.coef_[0][1] * np.power(X2_sorted, 2)
boundary = a * np.power(X1_sorted, 2) + b * X1_sorted + c

plt.plot(X1_sorted, boundary, linewidth=5, c='black', label="Decision boundary")
plt.axhline(-mostCommonValue, c='purple', linewidth=5, label='Baseline model3')
plt.legend(scatterpoints=1, fontsize=20, loc = "upper left")
plt.title("Logistic Regression Classifier with Squared Features")
plt.xlabel("X1", fontsize=20)
plt.ylabel("X2", fontsize=20)
plt.show()
```