Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CS7CS4/CSU44061 - Machine Learning
Assignment 2:

Student Name:
Hugh Jordan

Student Number:
16321743

Assignment 2

(a) Logistic Regression with Two Parameters

(i) The downloaded data (# id:14--28-14) was visualised by placing a marker on a 2D plot for each of the feature values. This is shown in Figure 1. It is immediately evident from Figure 1 that the data is not linearly separable.

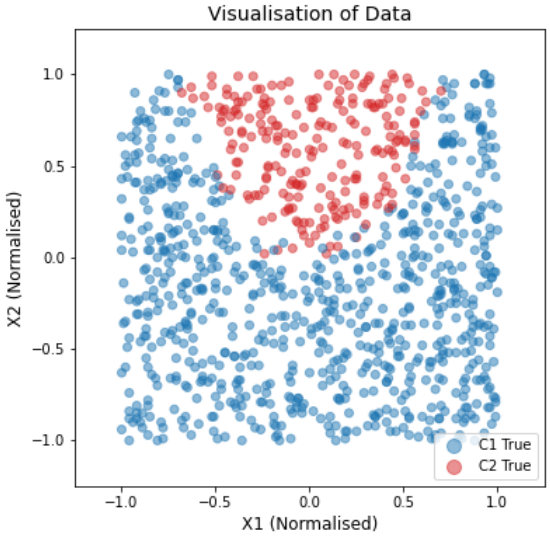

Figure 1. Visualisation of Data

(ii) Sklearn was then used to train a logistic regression classifier on the data. The parameter values of the trained model are shown below in Table 1. It is evident from this table that these are the parameter values of a linear classifier.

Model	θ_0	θ_1	θ_2
Logistic Regression	-2.26838413	-0.07142744	3.83592543

Table 1. Parameter Values of Linear Regression Model with two features

(iii) The trained logistic regression classifier was then used to predict the target values in the training data. This is shown in Figure 2. It is evident from Figure 2 that the classifier is linear. However, the data is not linearly separable. As a result, the classifier underfits the data and misclassified large sections of both classes. This is most evident beneath the decision boundary where the predicted values for C2 overlap with the true values of C1.

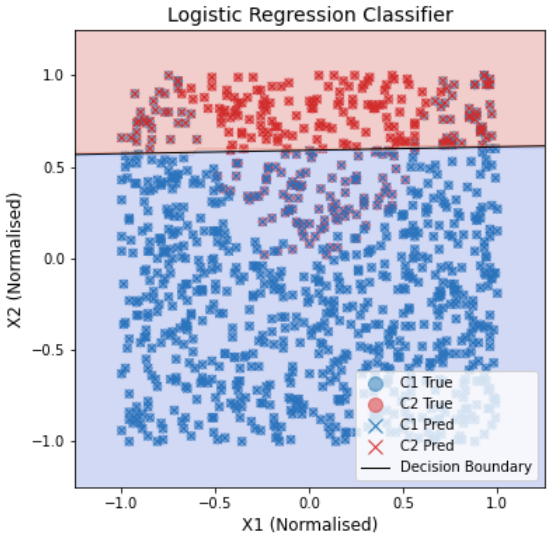

Figure 2. Logistic Regression Classifier with two features

(iv) The training data is not linearly separable. However, the classifier attempted to separate the two classes using a linear model. Consequently, the classifier only achieved a score of 83% on the training data. This poor performance is a result of underfitting. It is clear that the model was unable to capture the relationship between the training data and the target values. This is more than likely due to the fact that the model was only provided with two features.

(b) Linear SVM

(i) The penalty parameter values 0.001, 1 and 1000 were chosen to train an SVM classifier. The corresponding score for each of these classifiers is shown in Table 2. It is evident from Table 2 that if the penalty parameter is too small (C = 0.001) or if the penalty parameter is too large (C = 1000), then the model will fit the data poorly and the model's score will decrease as a result. However, if an appropriate penalty parameter (C = 1.0) is chosen, then the model can achieve a similar performance to that of a logistic regression classifier. This is to be expected considering that the only difference between Logistic Regression and Linear SVM is the choice of the cost function. In fact, both the logistic regression classifier and the SVM classifier that used a penalty parameter of 1.0 achieved a score of 83%.

Penalty Parameter	Score
0.001	0.78
1.0	0.83
1000	0.75

Table 2. Score values of SVM classifiers

(ii) The three trained classifiers were then used to predict the target values in the training data. These predictions are shown in Figure 3 alongside their decision boundaries. It is evident from Figure 3 that when the penalty parameter is equal to 0.001, that every member of C2 is completely misclassified. Furthermore, it is evident that when the penalty parameter is equal to 1000, a large section of C1 is misclassified. Finally, it can be observed that the least number of members are misclassified when the penalty parameter is equal to 1.0. Therefore, careful consideration should be given to the choice of penalty parameters as it can have a significant impact on the classification results.

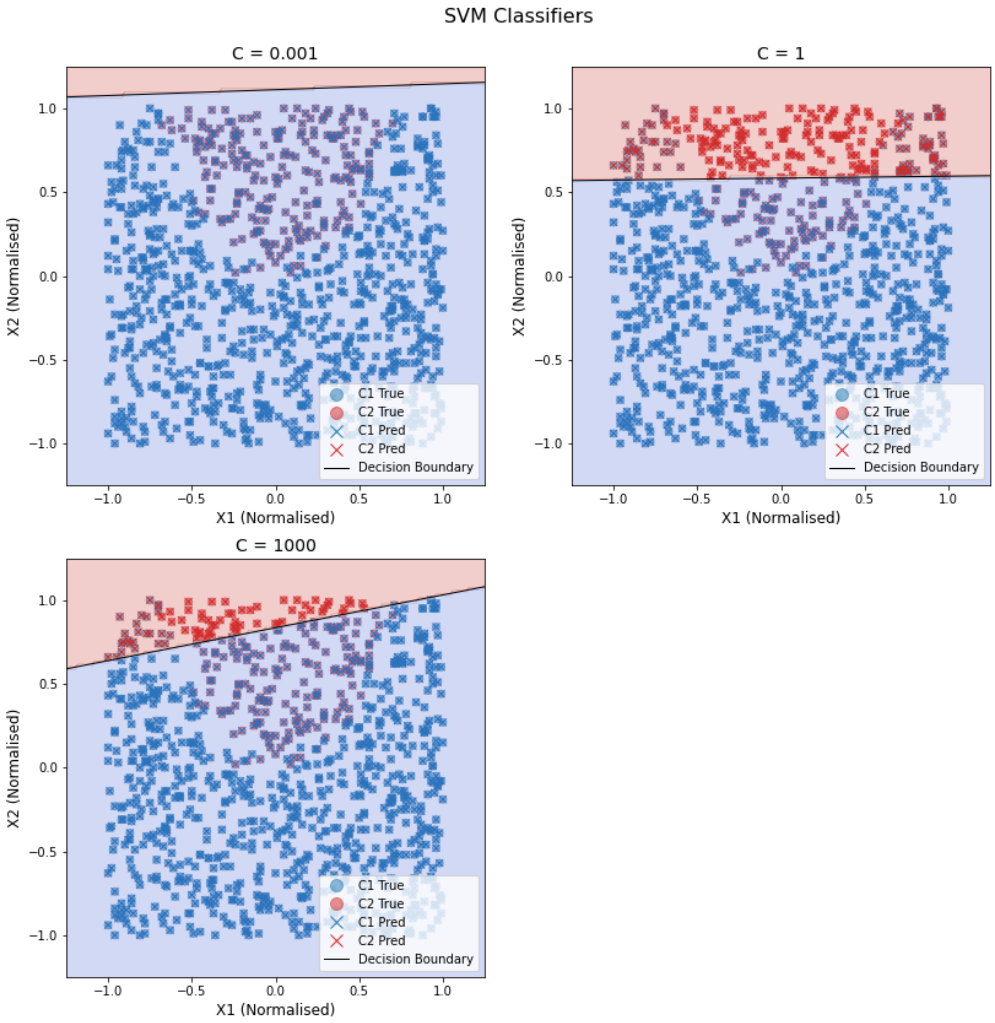

Figure 3. SVM Classifiers with a range of different penalty parameters

(iii) Changing the value of the penalty parameter can have a severe impact on the model's parameters and the classifier's predictions as a result. Consider the decision boundary. The boundary line is plotted using the model's parameters. Therefore, the significant difference between each of the classifier's decision boundaries strongly suggests that the penalty parameter has a severe impact on the model's parameters and, consequently, a significant impact on the classifier's predictions.

(c) Logistic Regression with Four Parameters

(i) Two additional features were created by adding the square of each feature. A logistic regression classifier was then trained using all four features. The parameter values of the trained model are shown in Table 3. It is evident from Table 3 that these are the parameters of a polynomial classifier.

Model	θ_0	θ_1	θ_2	θ_3	θ_4
Logistic Regression	-0.406132	0.771448	31.187171	-48.334913	-7.985815

Table 3. Parameter values of Linear Regression Model with four features

(ii) The trained classifier was then used to predict the target values in the training data. These predictions are shown in Figure 4. It is evident from Figure 4 that the logistic regression models no longer underfits the training data. The addition of the two extra features allowed the classifier to capture the relationship between the training data and the target values. As a result, the model no longer misclassified large portions of either class.

(iii) The performance of the classifier was compared against a baseline predictor that always predicts the most common class. This is shown in Table 4. It is evident from Table 4 that the classifier is significantly better than the alternative classifier. The logistic regression classifier achieves a score of 98% whereas the Dummy classifier only achieves a score of 64%. This baseline classifier was created using the sklearn's dummy classifier. This classifier is designed to be used as a comparison and not as a real classifier.

Model	Score
Logistic Regression	0.981982
Baseline (Dummy)	0.641642

Table 4. Classifier Comparison

(iv) The classifier decision boundary was then plotted in Figure 4. The equation for this boundary was calculated in a similar manner to the linear model. The equation for the linear logistic regression model is given by:

$$f(x, y) = c_0 + c_1 x + c_2 y$$

This formula can be extended, however, to utilise a higher degree polynomial. The equation for the polynomial logistic regression is given by:

$$f(x, y) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 x^2 + \theta_4 y^2$$

The decision line was then calculated allowing f(x,y) = 0 and then isolating the feature y in order to give the following equation:

$$y = -\frac{\theta_2}{2\theta_4} \pm \sqrt{\frac{-(\theta_0 + \theta_1 x + \theta_3 x^2)}{\theta_4} + \left(\frac{\theta_2}{2\theta_4}\right)^2}$$

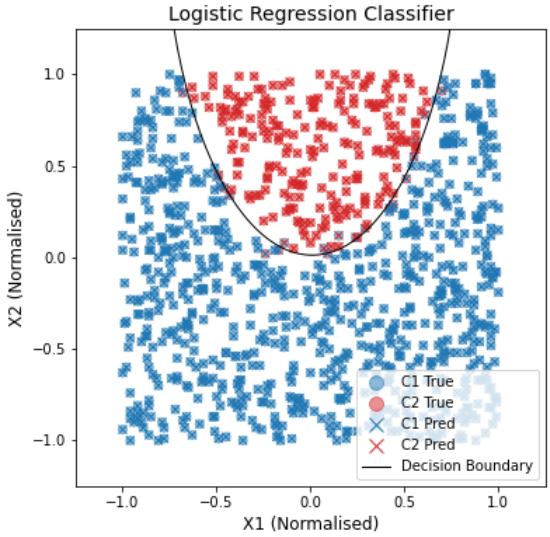

Figure 4. Logistic Regression with four parameters.

Appendix:

```
import numpy as np
import pandas as pd
import math
import matplotlib
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
from matplotlib.lines import Line2D
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.dummy import DummyClassifier

df = pd.read_csv('week2.csv')
X1=df.iloc[:, 0]
X2=df.iloc[:, 1]
X3=X1**2
X4=X2**2
X=np.column_stack((X1,X2, X3, X4))
y=df.iloc[:, 2]

# Visualise the downloaded data
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(6, 6))
ax.set_title("Visualisation of Data", fontsize=14)
ax.set_xlabel("X1 (Normalised)", fontsize=12)
ax.set_ylabel("X2 (Normalised)", fontsize=12)
x_min, x_max = X[:, 0].min() - 0.25, X[:, 0].max() + 0.25
y_min, y_max = X[:, 1].min() - 0.25, X[:, 1].max() + 0.25
ax.set_xlim([x_min, x_max])
ax.set_ylim([y_min, y_max])
colors = ['C0' if marker == -1 else 'C3' for marker in y]
ax.scatter(X[:, 0], X[:, 1], marker='o', facecolors=colors, linewidth=1.0, alpha=0.5)
legend_elements = [Line2D([0], [0], marker='o', color='C0', label='C1 True',
                    alpha=0.5, linestyle = 'None', markersize=10),
                Line2D([0], [0], marker='o', color='C3', label='C2 True',
                    alpha=0.5, linestyle = 'None', markersize=10)]
ax.legend(handles=legend_elements, loc='lower right')
plt.savefig('data_visualisation')

# Train a logistic regression classifier on the data
logreg = LogisticRegression(penalty='none', solver='lbfgs')
logreg.fit(X[:,0:2], y)
print('Logistic Regression Parameter Values Table:')
print('%-20s %-21s %-20s' % ('Linear Model', 'Intercept', 'Slope'))
print('%-20s %-21a %-20a' % ('Logistic Regression', logreg.intercept_, logreg.coef_))

x = np.array([x_min, x_max])
decbond = -(logreg.coef_[0][0] * x + logreg.intercept_)/logreg.coef_[0][1]
```

```
x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
h = 0.02
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = logreg.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Classify values using trained logistic regression classifer and
# show the decision boundary of the classifier
ax.set_title("Logistic Regression Classifier", fontsize=14)
colors = ['C0' if marker == -1 else 'C3' for marker in logreg.predict(X[:,0:2])]
ax.scatter(X[:, 0], X[:, 1], marker='x', c=colors, linewidth=1.0, alpha=1.0)
ax.plot(x, decbond, c='black', linewidth=1.0)
ax.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.25)
legend_elements = [Line2D([0], [0], marker='o', color='C0', label='C1 True',
                    alpha=0.5, linestyle = 'None', markersize=10),
                Line2D([0], [0], marker='o', color='C3', label='C2 True',
                    alpha=0.5, linestyle = 'None', markersize=10),
                Line2D([0], [0], marker='x', color='w', label='C1 Pred',
                    markeredgecolor='C0', linestyle = 'None', markersize=10),
                Line2D([0], [0], marker='x', color='w', label='C2 Pred',
                    markeredgecolor='C3', linestyle = 'None', markersize=10),
                Line2D([0],[0], color='black', lw=1, label='Decision Boundary')]
ax.legend(handles=legend_elements, loc='lower right')
plt.savefig('lr_classifier_with_2_params')
plt.show()

# Train linear SVM classifer for a range of penalty parameter values
# and use each trained classifiers to predict the target values in
# the training data
print('SVM Parameter Values Table:')
print('%-8s %-21s %-20s' % ('Penalty', 'Intercept', 'Slope'))
penalties = [0.001, 1, 1000]
svms = []
for penalty in penalties:
   svm = LinearSVC(C=penalty)
   svm.fit(X[:,0:2], y)
   svms.append(svm)
   print('%-8s %-21a %-20a' % (penalty, svm.intercept_, svm.coef_))

# Create two addition features by adding the square of each feature and
# train a logistic regression classifier
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(12, 12));
ax = ax.flatten()
fig.suptitle('SVM Classifiers', fontsize=16)
fig.tight_layout(pad=4.0)
fig.subplots_adjust(top=0.925)
```

```
for itr, penalty in enumerate(penalties):
    ax[itr].set_title('C = ' + str(penalty), fontsize=14)
    ax[itr].set_xlabel('X1 (Normalised)', fontsize=12)
    ax[itr].set_ylabel('X2 (Normalised)', fontsize=12)
    x_min, x_max = X[:, 0].min() - 0.25, X[:, 0].max() + 0.25
    y_min, y_max = X[:, 1].min() - 0.25, X[:, 1].max() + 0.25
    ax[itr].set_xlim([x_min, x_max])
    ax[itr].set_ylim([y_min, y_max])
    colors = ['C0' if marker == -1 else 'C3' for marker in y]
    ax[itr].scatter(X[:, 0], X[:, 1], marker='o', facecolors=colors, linewidth=1.0, alpha=0.5)
    colors = ['C0' if marker == -1 else 'C3' for marker in svms[itr].predict(X[:,0:2])]
    ax[itr].scatter(X[:, 0], X[:, 1], marker='x', c=colors, linewidth=1.0, alpha=1.0)

    x = np.array([x_min, x_max])
    decbond = -(svms[itr].coef_[0][0] * x + svms[itr].intercept_)/svms[itr].coef_[0][1]
    ax[itr].plot(x, decbond, c='black', linewidth=1.0)
    x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
    y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
    h = 0.02
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
    Z = svms[itr].predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    ax[itr].contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.25)

    legend_elements = [Line2D([0], [0], marker='o', color='C0', label='C1 True',
                       alpha=0.5, linestyle = 'None', markersize=10),
                   Line2D([0], [0], marker='o', color='C3', label='C2 True',
                       alpha=0.5, linestyle = 'None', markersize=10),
                   Line2D([0], [0], marker='x', color='w', label='C1 Pred',
                       markeredgecolor='C0', linestyle = 'None', markersize=10),
                   Line2D([0], [0], marker='x', color='w', label='C2 Pred',
                       markeredgecolor='C3', linestyle = 'None', markersize=10),
                   Line2D([0],[0], color='black', lw=1, label='Decision Boundary')]
    ax[itr].legend(handles=legend_elements, loc='lower right')
ax[-1].axis('off')
plt.savefig('svm_classifiers')
plt.show()

# Compare the performance of the classsifier against a reasonable baseline predictor
logreg = LogisticRegression(penalty='none', solver='lbfgs')
logreg.fit(X, y)
print('Logistic Regression Parameter Values Table:')
print('%-20s %-21s %-20s' % ('Linear Model', 'Intercept', 'Slope'))
print('%-20s %-21a %-20a' % ('Logistic Regression', logreg.intercept_, logreg.coef_))

# Plot the classifier decision boundary
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(6, 6))
```

```
ax.set_title("Logistic Regression Classifier", fontsize=14)
ax.set_xlabel("X1 (Normalised)", fontsize=12)
ax.set_ylabel("X2 (Normalised)", fontsize=12)
x_min, x_max = X[:, 0].min() - 0.25, X[:, 0].max() + 0.25
y_min, y_max = X[:, 1].min() - 0.25, X[:, 1].max() + 0.25
ax.set_xlim([x_min, x_max])
ax.set_ylim([y_min, y_max])
colors = ['C0' if marker == -1 else 'C3' for marker in y]
ax.scatter(X[:, 0], X[:, 1], marker='o', facecolors=colors, linewidth=1.0, alpha=0.5)
colors = ['C0' if marker == -1 else 'C3' for marker in logreg.predict(X)]
ax.scatter(X[:, 0], X[:, 1], marker='x', c=colors, linewidth=1.0, alpha=1.0)

x = np.arange(x_min, x_max, h)
a = (logreg.coef_[0][2] * x**2 + logreg.coef_[0][0] * x +
logreg.intercept_[0])/logreg.coef_[0][3]
b = logreg.coef_[0][1]/(2 * logreg.coef_[0][3])
decbond = -b - np.sqrt(-a + b**2)
ax.plot(x, decbond, c='black', linewidth=1.0)
# x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
# y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
# h = 0.02
# xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
# Z = logreg.predict(np.c_[xx.ravel(), yy.ravel()])
# Z = Z.reshape(xx.shape)
# ax.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.25)

legend_elements = [Line2D([0], [0], marker='o', color='C0', label='C1 True',
                    alpha=0.5, linestyle = 'None', markersize=10),
                Line2D([0], [0], marker='o', color='C3', label='C2 True',
                    alpha=0.5, linestyle = 'None', markersize=10),
                Line2D([0], [0], marker='x', color='w', label='C1 Pred',
                    markeredgecolor='C0', linestyle = 'None', markersize=10),
                Line2D([0], [0], marker='x', color='w', label='C2 Pred',
                    markeredgecolor='C3', linestyle = 'None', markersize=10),
                Line2D([0],[0], color='black', lw=1, label='Decision Boundary')]
ax.legend(handles=legend_elements, loc='lower right')
plt.savefig('lr_classifier_with_4_params')
plt.show()

dummy = DummyClassifier(strategy='stratified')
dummy.fit(X[:, 2:4], y)
print('Classifier Comparison:')
print('%-20s %-20s' % ('Linear Model', 'Score'))
print('%-20s %-20f' % ('Logistic Regression', logreg.score(X, y)))
print('%-20s %-20f' % ('Dummy', dummy.score(X, y)))
```