Week 2 assignment

Name: Tianze Zhang
Student number: 19334401
Dataset id: id:8--16--8

(a)(i) Figure 1.1 shows an overview of the dataset, where the x axis represents the first column x1, and y axis represents the second column x2. The labels of each row are differentiated by different colours, label with -1 is presented as red and +1 as yellow, which are shown on the legend.

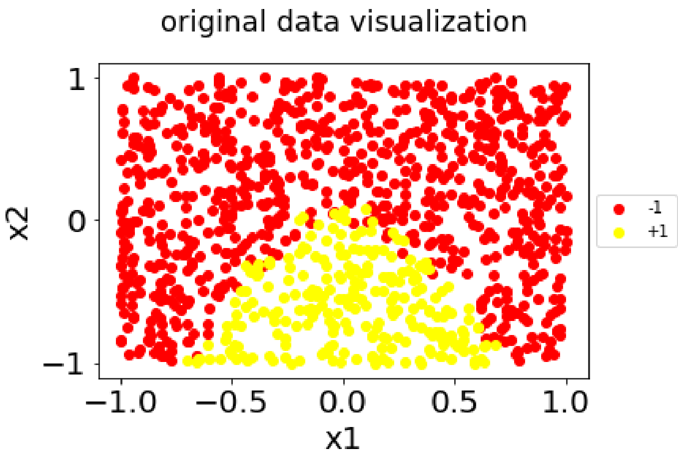

Figure 1.1

(ii)

```
#train test split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size = 0.2,stratify=y)

#training with logistic regression
lr = LogisticRegression(penalty='none', solver='lbfgs')
lr.fit(Xtrain,ytrain)
print(lr.intercept_)
print(lr.coef_)
```

Logistic Regression is used using two feature variables, with datasets of train and test split by 0.8 to 0.2. The model is trained with a logistic regression $y = \theta_0+\theta_1 x1+\theta_2 x2$, where θ_0, θ_1 and θ_2 are the parameters and x1, x2 are the input features. The parameter values of the trained model are $\theta_0 = -2.09$, $\theta_1 = 0.27$ and $\theta_2 = -3.5$. The coefficient that is larger and have bigger influence on the prediction results. Here assuming the value of feature x1, and x2 have the same value range. Therefore, x2 (feature 2) is the most influent feature in the prediction, because for each change in x2, the prediction will change by 3.51, however, only changes by 0.27 for each change of x1. Feature 2 is the decreasing feature of the prediction, because the coefficient of x2 is negative, which will resulting in a decreasing prediction as x2 increases. Feature 1 is the increasing feature because the coefficient of x1 is positive, which the prediction increases while x1 increases.

Logistic regression model: y = sign (0.27x1-3.5x2+2.09)

(iii) Figure 1.2 plots the predicted data on Xtest from the train logistic regression model (`y_pred = lr.predict(Xtest)`). The green plus symbols on the plot shows the predicted -1, and cyan represents +1.

The decision boundary is generated as follow:

The way the decision boundary is generated is by utilizing the coefficient of the logistic regression model and then generate points by the equation y=mx+c. Different points points_x between 1 and -1 are generated by the numpy's linspace function, and the corresponding y values points_y are calculated by substituting in the equation.

```
points_x = np.linspace(-1, 1, 1000)
line_bias = lr.intercept_
line_w = lr.coef_.T
points_y=[(line_w[0]*x+line_bias)/(-1*line_w[1]) for x in points_x]
plt.plot(points_x, points_y)
```

The equation is derived by the parameters of the model as follow:
For logistic regression $y = \theta_0+\theta_1 x_1+\theta_2 x_2$, we can think of the boundary as a line $x_2=mx_1+c$, For the gradient, m, consider two distinct points on the decision boundary, (x_1,x_2) and (x_3,x_4) , so that $m=(x_4-x_2)/(x_3-x_1)$.

$0 = \theta_1 x_3+ \theta_2 x_4+ \theta_0 -(\theta_1 x_1+ \theta_2 x_2+ \theta_0)$

$-\theta_2 (x_4-x_2)= \theta_1(x_3-x_1)$

$(x_4-x_2)/ (x_3-x_1) = - \theta_1/ \theta_2$

$M=- \theta_1/ \theta_2$

Therefore, comes the equation $y = - \theta_1/ \theta_2*x+c$

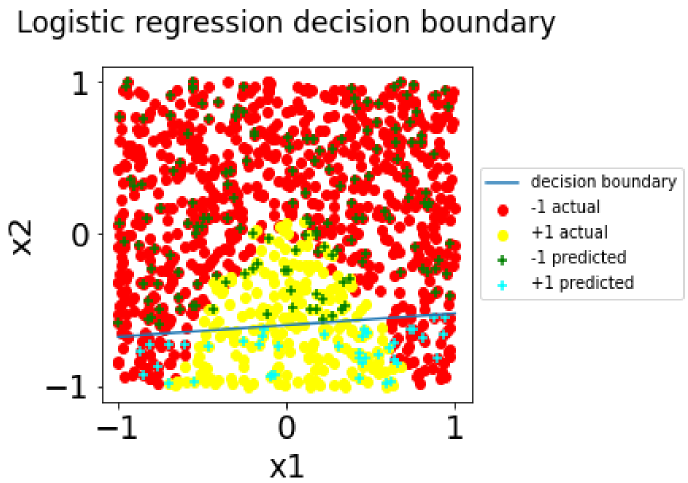

Figure 1.2

(iv) Based on the accuracy scores from training and testing(prediction), the accuracy score for testing is 0.795. This means approximately 79.5% proportion of the data is predicted accurately of the overall data. From the graph we can see, -1 data at the bottom left and bottom right under the decision boundary are predicted incorrectly, while the middle upper of the decision of the +1 data are predicted incorrectly.

(b)(i)

Here 3 linear SVM are trained with different a penalty parameter C of 0.0001, 1 and 100 respectively. Model1 gives an intercept of -0.32 and coefficient of 0.02 and -0.29. Model2 gives an intercept of -0.70 and coefficient of 0.11 and -1.19. Model 3 gives an intercept of -0.68 and coefficient of 0.09 and -1.24.

```
#training with linear SVM classifier
model1 = LinearSVC(C=0.001).fit(Xtrain, ytrain)
model2=LinearSVC(C=1).fit(Xtrain, ytrain)
model2=LinearSVC(C=100).fit(Xtrain, ytrain)
```

model1: y = sign (0.02x1-0.29x2-0.32)
model2: y = sign (0.11x1-1.19x2-0.7)
model3: y = sign (0.09x1-1.24x3-0.68)

(ii) Figure 2.1 plots the predicted test data from the linear SCM with a penalty parameter 0.001. Figure 2.2 plots the predicted test data from the linear SCM with a penalty parameter 1. Figure 3.3 plots the predicted test data from the linear SCM with a penalty parameter 100.

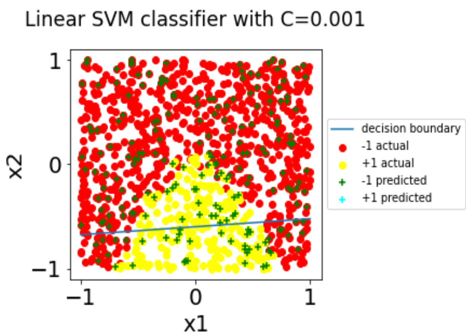

Figure 2.1

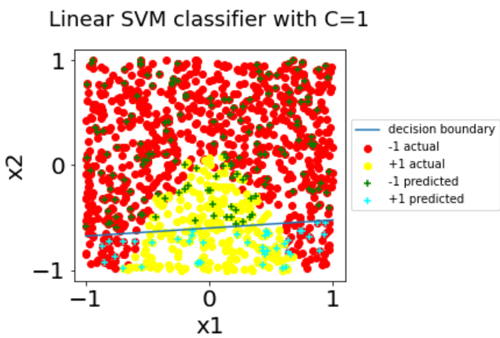

Figure 2.2

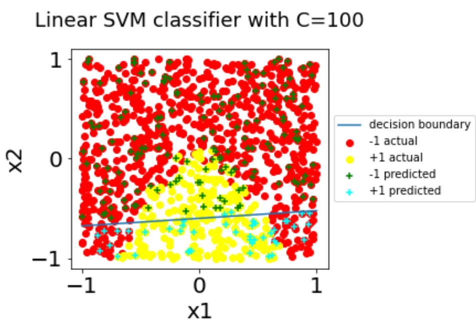

Figure 2.3

(iii) The cost function for linear SVM is $J(\theta) = 1/m\sum_{i=1}^{m} \max(0, 1 - y^{(i)}\theta^T x^{(i)}) + \theta^T\theta/C$, where C is the penalty parameter. For larger value of C, the penalty is less important, which means a smaller value of $\theta^T\theta/C$, thus perform a better job of getting all the training points classified. However, for small value of C will cause a stronger penalty and lager $\theta^T\theta/C$. Therefore, it often misclassifies more data. This is shown in figure 2.1 when C=0.01 it clearly misclassifies more points than when C is 1 or 100, since there are none +1 predicted anywhere. However, C=1 and C=100, they perform a similar job, because when C is too big, the model won't get more better increasing it.

(iv) The accuracy score of the test data on model1, model2 and model3 are 0.76,0.975 and 0.805 respectively. Model1's accuracy is less than the logistic regression due to the lowness of C. Model2's accuracy is same as logistic regression at (a), Model3's accuracy is larger than logistic at (a) because of the increasing of C. Additionally, the parameters of the logistic regression is larger than the parameters of all three models. For logistic having parameters 3.5, 0.27 and 3.09. While the largest of SVM 0.11, 1,24 and 0.7.

(c)(i) The logistic regression y = θ0+θ1x1+θ2x2+ θ3x1^2 + θ4x2^2, is trained with 4 features by a logistic regression from sklearn. The model gives an intercept θ0 of 2.78, θ1 of 0.56, θ2 of -26.41, θ3 of -52.48 and θ4 of -0.38.

Logistic regression: sign(0.56x1-26.41x2-52.48x3-0.38x4+2.78)

(c)(ii) Figure 3.1 plots the logistic regression with two additional features x1^2 and x2^2 but only keeping x1 and x2.

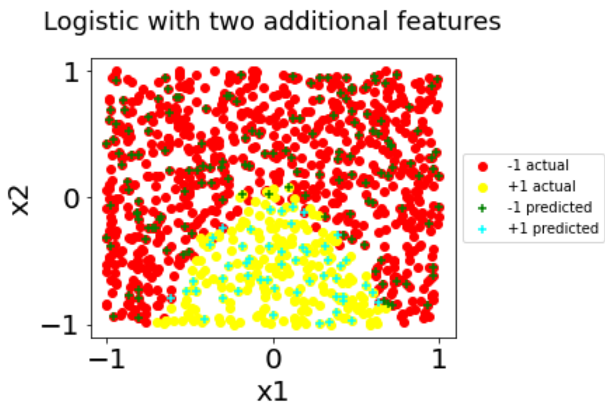

Figure 3.1

The model testing score is 0.955, significantly larger than the previous two models of 0.795 and 0.805. The parameters of this model are also significantly larger, with an intercept of 2.78, coefficient of 0.56 and -26.41. For logistic having parameters 3.5, 0.27 and 3.09. While the largest of SVM 0.11, 1,24 and 0.7. From looking at the graph, this model no longer misclassifies the bottom left and bottom right data, and the curve at the top.

(iii) Since the linear SVM of C=100 has a better accuracy than the logistic regression in part (a), I would choose it as my base predictor. Following from the previous question, the current model definitely outperforms the base predictor.

(iv)Didn't code it out, but I would attempt as follow:

1. Use the best model with the quadratic features above,
Based on the model parameters, the decision boundary equation is
2.78 + 0.56x1-26.41x2-52.48 x1 square -0.38 x2 square = 0
2. To resolve the decision boundary equation (from x1 to x2)

```
def boundary(x1):
 ans = [(-b_ + tmp**0.5) / (2*a_), (-b_ - tmp**0.5) / (2*a_)]
 ans.sort()
 return ans
```

3. Draw the decision boundary

Appendix

```
#!/usr/bin/env python
# coding: utf-8

# # Week2 assignment    Tianze Zhang  dataset: id:8--16--8

# # (a) logistic regression
#
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC

df = pd.read_csv('week2.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)

#training with logistic regression
lr = LogisticRegression(penalty='none',solver='lbfgs')
```

```
lr.fit(Xtrain,ytrain)
print(lr.intercept_)
print(lr.coef_)
print(lr.score(Xtest,ytest))
#predict
y_pred = lr.predict(Xtest)
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 regression decision boundary",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='+')
points_x = np.linspace(-1, 1, 1000)
line_bias = lr.intercept_
line_w = lr.coef_.T
points_y=[(line_w[0]*x+line_bias)/(-1*line_w[1]) for x in points_x]
plt.plot(points_x, points_y)
plt.xlabel('x1'); plt.ylabel('x2')
plt.legend(['decision boundary','-1 actual','+1 actual','-1 predicted','+1 predicted'],loc='center left',
bbox_to_anchor=(1, 0.5),prop={'size': 10})
plt.savefig('logistic.png', facecolor='w', transparent=False)
plt.show()


# # (b) linear SVM classifier
#training with linear SVM classifier
model1 = LinearSVC(C=0.001).fit(Xtrain, ytrain)
print(model1.intercept_)
print(model1.coef_)
print(model1.score(Xtest,ytest))
model2=LinearSVC(C=1).fit(Xtrain, ytrain)
print(model2.intercept_)
print(model2.coef_)
print(model2.score(Xtest,ytest))
model3=LinearSVC(C=100).fit(Xtrain, ytrain)
print(model3.intercept_)
print(model3.coef_)
print(model3.score(Xtest,ytest))

#predict
y_pred1 = model1.predict(Xtest)
```

```
predict1=np.column_stack([Xtest, y_pred1])
df = pd.DataFrame(predict1, columns = ['x1','x2','y'])
neg_pred1 = df[df.y==-1]
pos_pred1 = df[df.y==1]
x1_pred_neg1 = neg_pred1.iloc[:,0]
x2_pred_neg1 = neg_pred1.iloc[:,1]
x1_pred_pos1 = pos_pred1.iloc[:,0]
x2_pred_pos1 = pos_pred1.iloc[:,1]

y_pred2 = model2.predict(Xtest)
predict2=np.column_stack([Xtest, y_pred2])
df = pd.DataFrame(predict2, columns = ['x1','x2','y'])
neg_pred2 = df[df.y==-1]
pos_pred2 = df[df.y==1]
x1_pred_neg2 = neg_pred2.iloc[:,0]
x2_pred_neg2 = neg_pred2.iloc[:,1]
x1_pred_pos2 = pos_pred2.iloc[:,0]
x2_pred_pos2 = pos_pred2.iloc[:,1]

y_pred3 = model3.predict(Xtest)
predict3=np.column_stack([Xtest, y_pred3])
df = pd.DataFrame(predict3, columns = ['x1','x2','y'])
neg_pred3 = df[df.y==-1]
pos_pred3 = df[df.y==1]
x1_pred_neg3 = neg_pred3.iloc[:,0]
x2_pred_neg3 = neg_pred3.iloc[:,1]
x1_pred_pos3 = pos_pred3.iloc[:,0]
x2_pred_pos3 = pos_pred3.iloc[:,1]

plt.rc('font', size=20)
plt.rcParams['figure.constrained_layout.use'] = True
plt.title("Linear SVM classifier with C=0.001",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='+')
points_x = np.linspace(-1, 1, 1000)
line_bias = lr.intercept_
line_w = lr.coef_.T
points_y=-(line_w[0] / line_w[1]) * points_x - line_bias / line_w[1]
plt.plot(points_x, points_y)
plt.xlabel('x1'); plt.ylabel('x2')
plt.legend(['decision boundary','-1 actual','+1 actual','-1 predicted','+1 predicted'],loc='center left',
bbox_to_anchor=(1, 0.5),prop={'size': 10})
plt.savefig('SVM C0001.png', facecolor='w', transparent=False)
plt.show()

plt.rc('font', size=20)
plt.rcParams['figure.constrained_layout.use'] = True
plt.title("Linear SVM classifier with C=1",fontsize=18,pad=20)
plt.scatter(x1_neg, x2_neg, color='red')
```

```
plt.scatter(x1_pos, x2_pos, color='yellow')
plt.scatter(x1_pred_neg2, x2_pred_neg2, color='green',marker='+')
plt.scatter(x1_pred_pos2, x2_pred_pos2, color='cyan',marker='+')
points_x = np.linspace(-1, 1, 1000)
line_bias = lr.intercept_
line_w = lr.coef_.T
points_y=-(line_w[0] / line_w[1]) * points_x - line_bias / line_w[1]
plt.plot(points_x, points_y)
plt.xlabel('x1'); plt.ylabel('x2')
plt.legend(['decision boundary','-1 actual','+1 actual','-1 predicted','+1 predicted'],loc='center left',
bbox_to_anchor=(1, 0.5),prop={'size': 10})
plt.savefig('SVM C1.png', facecolor='w', transparent=False)
plt.show()

plt.rc('font', size=20)
plt.rcParams['figure.constrained_layout.use'] = True
plt.title("Linear SVM classifier with C=100",fontsize=18,pad=20)
plt.scatter(x1_neg, x2_neg, color='red')
plt.scatter(x1_pos, x2_pos, color='yellow')
plt.scatter(x1_pred_neg3, x2_pred_neg3, color='green',marker='+')
plt.scatter(x1_pred_pos3, x2_pred_pos3, color='cyan',marker='+')
points_x = np.linspace(-1, 1, 1000)
line_bias = lr.intercept_
line_w = lr.coef_.T
points_y=-(line_w[0] / line_w[1]) * points_x - line_bias / line_w[1]
plt.plot(points_x, points_y)
plt.xlabel('x1'); plt.ylabel('x2')
plt.legend(['decision boundary','-1 actual','+1 actual','-1 predicted','+1 predicted'],loc='center left',
bbox_to_anchor=(1, 0.5),prop={'size': 10 })
plt.savefig('SVM C100.png', facecolor='w', transparent=False)
plt.show()


# # (c) Logistic regression with two additional features

x3 = x1 **2
x4 = x2 **2
X=np.column_stack((x1,x2,x3,x4))

#train test split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size = 0.2,stratify=y)

#training with logistic regression
lr4 = LogisticRegression(penalty='none',solver='lbfgs')
lr4.fit(Xtrain,ytrain)
print(lr4.intercept_)
print(lr4.coef_)
print(lr4.score(Xtest,ytest))

#predict
y_pred = lr4.predict(Xtest)
```

```
predict=np.column_stack([Xtest, y_pred])
df = pd.DataFrame(predict, columns = ['x1','x2','x3','x4','y'])
neg_pred = df[df.y==-1]
pos_pred = df[df.y==1]
x1_pred_neg4 = neg_pred.iloc[:,0]
x2_pred_neg4 = neg_pred.iloc[:,1]
x1_pred_pos4 = pos_pred.iloc[:,0]
x2_pred_pos4 = pos_pred.iloc[:,1]

plt.rc('font', size=20)
plt.rcParams['figure.constrained_layout.use'] = True
plt.title("Logistic regression with two additional features",fontsize=18,pad=20)
plt.scatter(x1_neg, x2_neg, color='red')
plt.scatter(x1_pos, x2_pos, color='yellow')
plt.scatter(x1_pred_neg4, x2_pred_neg4, color='green',marker='+')
plt.scatter(x1_pred_pos4, x2_pred_pos4, 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('SVMAdditional.png', facecolor='w', transparent=False)
plt.show()
```