Name: Anshika Sharma

UCLA: 305488635

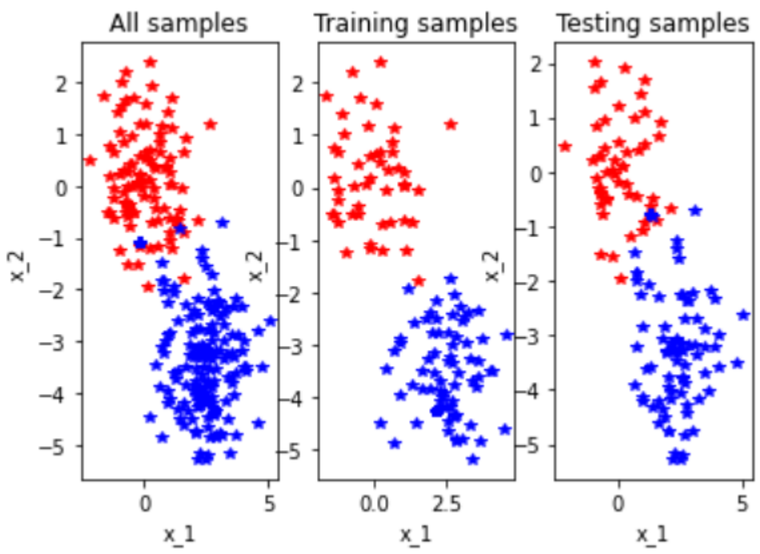
Homework Assignment 3

Part 1: Logistic Regression

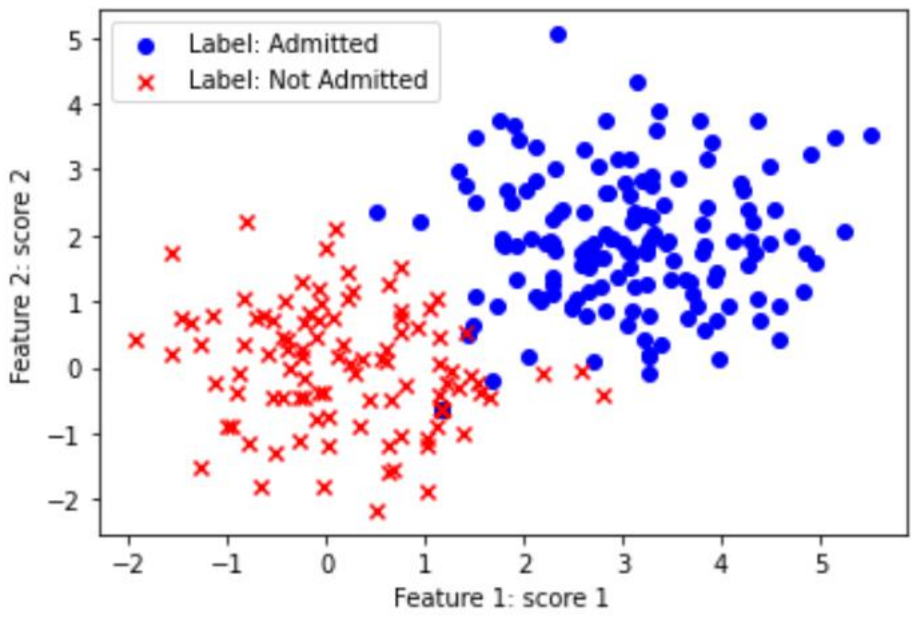
The first step was to generate the dataset, using the getDataSet() function. The dataset was composed of 250 samples (two features for each) that were associated with ground truth labels of either 0 or 1.

I split the dataset into a training dataset that consisted of 120 randomly selected samples, and the testing dataset which was composed of the 130 remaining samples.

I then plotted all the samples, the training samples and the testing samples in the following plots:



Next, I train a logistic model using my training data and the LogisticRegression() function. Then, I plot the data after assigning labels of "Admitted" and "Not Admitted":



We have a clear division between samples that are admitted and samples that are not admitted.

We were given two ways to learn a logistic regression, the first was through the gradient descent function and the second was through the sklearn method. After testing it using both methods on the testing data, we find that the sklearn method is more accurate and gives an overall better prediction than that of the gradient descent. Here is the output:

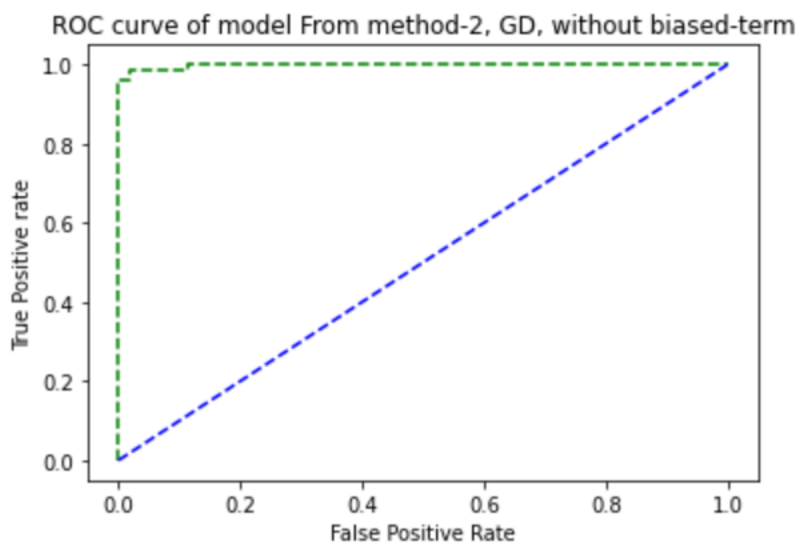
```
Scikit won.. :(
Your score:  0.8153846153846154
Scikits score:  0.9692307692307692
```

The sklearn method was consistently better across several different trials.

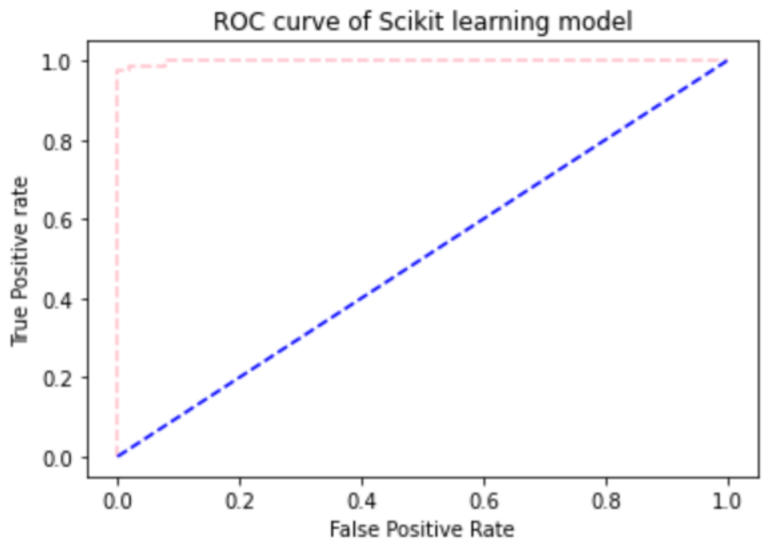
In order to evaluate the results, we compare the predicted labels and true labels by computing the average and standard errors and get the following results:

```
GD average error: 0.2 (0.4)
SK average error: 0.046153846153846156 (0.20981817995362856)
```

Now, I plot the ROC and calculate the AUC for the trained model that was formed using both Scikit learn and gradient descent both. Following is the result for both the implementations of the logistic regression model: By using Scikit-Learn module, the ROC curves and AUC of the trained model are calculated over testing samples as. The AUC score of two models is 0.99 confirming the good quality of separation; It means there is 99% chance that the model can distinguish between positive and negative class.



```
The AUC score is:  0.9980276134122288
```



```
The AUC score is:  0.998767258382643
```

Part 2: Confusion Matrix

Q1.

Given the following table:

Image ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
True class	C	C	C	C	C	D	D	D	D	D	D	D	D	M	M	M	M	M	M	M
Predicted class	D	C	D	D	M	D	D	C	C	M	M	D	C	C	C	M	M	D	D	M

Notes: C, cat; D, dog; M, monkey

I manually compute the following confusion matrix:

PREDICTION

GROUND TRUTH

	$\hat{y} = CAT$	$\hat{y} = DOG$	$\hat{y} = MONKEY$
$y = CAT$	1	3	1
$y = DOG$	3	3	2
$y = MONKEY$	2	2	3

Then I compute the following accuracy, precision, and recall rates:

ACCURACY
Accuracy = True positive + True negative/ Total
Accuracy = (1 + 3 + 3)/20
Accuracy = 7/20
Accuracy = 0.35

PRECISION

Cat Precision = 1/(1+3+2)
Cat Precision = 1/6

Dog Precision = 3/(3+3+2)
Dog Precision = 3/8
Dog Precision = 0.375

Monkey Precision = 3/(1+2+3)
Monkey Precision = 3/6
Monkey Precision = 0.5

RECALL

Cat Recall = 1/(1+3+1)
Cat Recall = 1/5
Cat Recall = 0.2

Dog Recall = 3/(3+3+2)
Dog Recall = 3/8
Dog Recall = 0.375

Monkey Recall = 3/(2+2+3)
Monkey Recall = 3/7

Q2: Comparative Studies

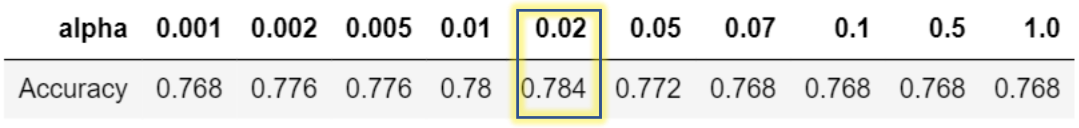
Finally, I create a function in Python to compute the confusion matrix of our dataset along with the accuracy, precision, and recall rates. I then use it to compute the confusion matrix for both the gradient descent method and the sklearn method. I get the following results:

```
SK Learn Confusion Matrix:
Confusion Matrix:
[[ 0. 51.  1.]
 [ 1.  5. 73.]]
Accuracy: 0.9538461538461539
Precision 0: 0.9107142857142857
Precision 1: 0.9864864864864865
Recall 0: 0.9807692307692307
Recall 1: 0.9358974358974359

GD Confusion Matrix:
Confusion Matrix:
[[ 0. 26. 26.]
 [ 1.  0. 78.]]
Accuracy: 0.8
Precision 0: 1.0
Precision 1: 0.75
Recall 0: 0.5
Recall 1: 1.0
```

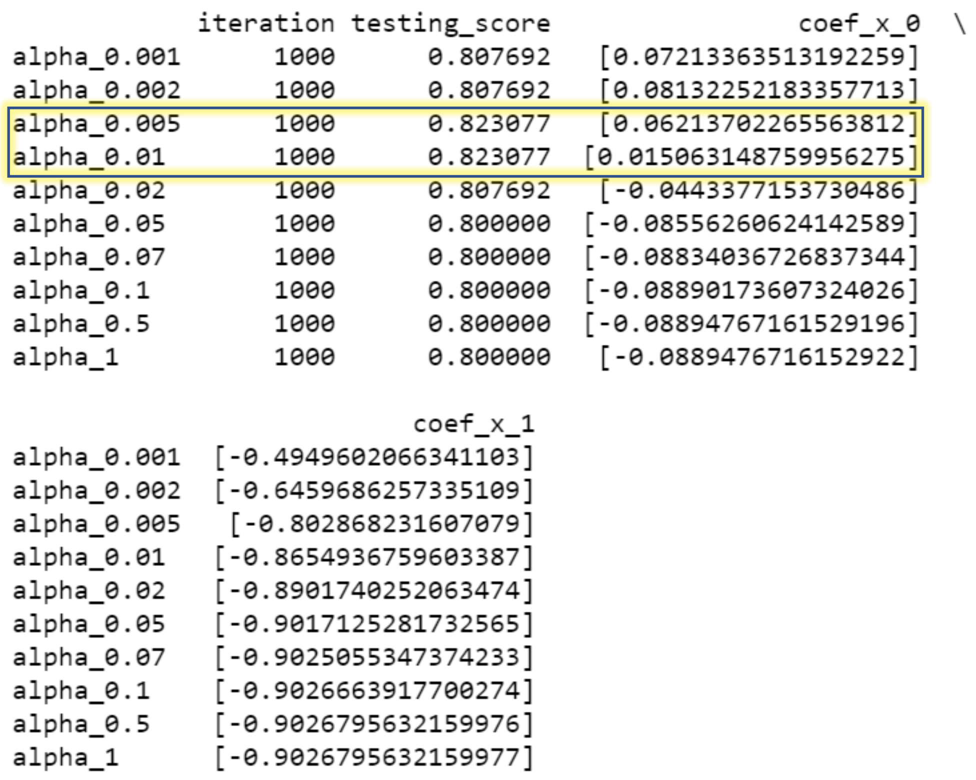
Part 3: Cross-Validation

For this part, I used the KFold package from the sklearn.model_selection. As instructed, I ran domly split the training data into 5 evenly sized groups and pulled out one groupat a time and trained the logistic model using other folds of the raining sample and test it or in other words, I calculated the classification accuracy over the current fold. I save the accuracy and do the same process for all the five groups that I created. Then, I averaged the accuracy over the five results to get the mean validation accuracy. I carried out this whole process for all the ten lear ning rates that were given to us. The "optimal learning rate" is the rate that gives us the highe st mean validation score.



alpha	0.001	0.002	0.005	0.01	0.02	0.05	0.07	0.1	0.5	1.0
Accuracy	0.768	0.776	0.776	0.78	0.784	0.772	0.768	0.768	0.768	0.768

I get the following result. The optimal learning rate is 0.02, with accuracy 0.784. But it is still less than the sklearn model. In the last step of this part, I used the entire training sample to train the model and made predictions on the testing set. For each learning rate, I calculated the model's accuracy over the testing samples. I obtained the following result:



```
             iteration testing_score                  coef_x_0  \
alpha_0.001       1000      0.807692   [0.07213363513192259]
alpha_0.002       1000      0.807692   [0.08132252183357713]
alpha_0.005       1000      0.823077   [0.06213702265563812]
alpha_0.01        1000      0.823077  [0.015063148759956275]
alpha_0.02        1000      0.807692   [-0.0443377153730486]
alpha_0.05        1000      0.800000  [-0.08556260624142589]
alpha_0.07        1000      0.800000  [-0.08834036726837344]
alpha_0.1         1000      0.800000  [-0.08890173607324026]
alpha_0.5         1000      0.800000  [-0.08894767161529196]
alpha_1           1000      0.800000   [-0.0889476716152922]

                          coef_x_1
alpha_0.001  [-0.4949602066341103]
alpha_0.002  [-0.6459686257335109]
alpha_0.005   [-0.802868231607079]
alpha_0.01   [-0.8654936759603387]
alpha_0.02   [-0.8901740252063474]
alpha_0.05   [-0.9017125281732565]
alpha_0.07   [-0.9025055347374233]
alpha_0.1    [-0.9026663917700274]
alpha_0.5    [-0.9026795632159976]
alpha_1      [-0.9026795632159977]
```

The best testing accuracy score is achieved when learning rate equals 0.1 which is not the same as the optimal learning rate that we found above which was 0.02. So, indeed the model with the optimal learning rate does not outperform all other candidates. We can say that applying cross validation does not guarantee the optimal value of the hyper parameter. The difference in performance score could be due to the training set in cross-validation being very small and hence cannot be a very good representation of the entire sample. That is why, It is always advised to choose a large sample set.