Tina Gholami 9531307
Data Mining_HW3 Report

Theoretical Questions (1-6):

***** The solution to the theoretical questions (questions 1 to 6 in the first part of the assignment) are sent in the "report" folder and is named "9531307_TinaGholami_Solutions of theory questions3.pdf".

Implementation 1: Decision Tree

(The name of the file: implementation1.ipynb, and tree1.jpg, tree2.jpg, tree3.jpg, tree4.jpg, tree5.jpg)

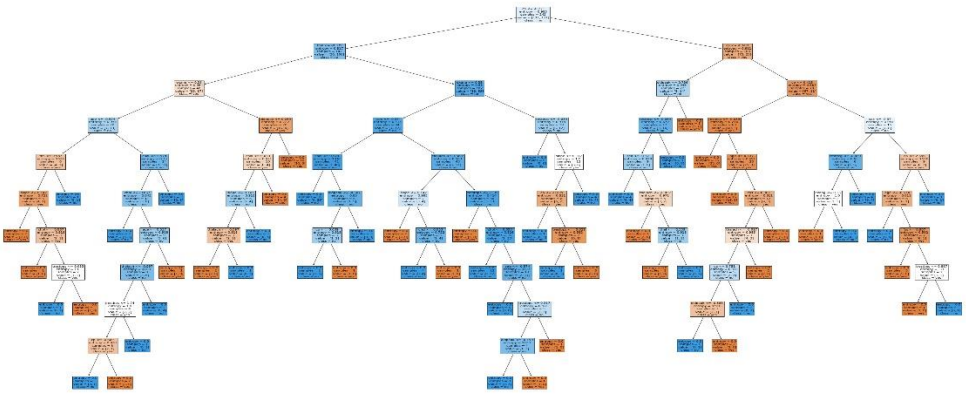
In part (1), I changed all the numerical data that had more than 4 classes to in-range data, meaning for example feature "age" from 1-10 became 0, 11-20 became 1, and so on. After that, I encoded these fewer classes to digits using "LabelEncoder" method from "sklearn". I encoded the non-in-range numerical data as well (so that their classes start from 0 to N_c). For the categorical and numerical features, I also applied label "LabelEncoder" method to convert them to digit classes. For example, class "sex" women became "0" and "man" became "1". And since "Unnamed: 0" column was actually useless, I dropped it so that it won't affect the decision tree classifier.

In part (2), I split the data to 80% train data and 20% test data, using "train_test_split" method from "sklearn", and with parameter "criterion = 'entropy'". (later in part (3), I tried other parameters as well.)

Then I tried to predict a single result from test_set[0] and the model predicted it correctly equal to y_test[0]. Afterwards, I applied the "predict" method for the whole test set and then concatenated the true results to the predicted results so that I could compare them better. Next, I used confusion matrix and accuracy for my model metrics to assess its performance. And the accuracy became "0.77" 0r just 77%.

Then I visualized the tree and saved it in "tree1.jpg":

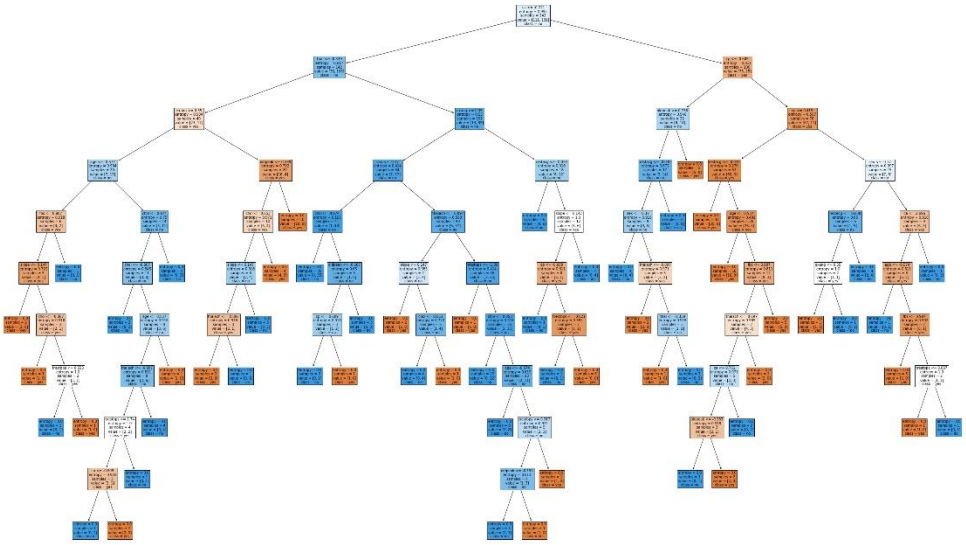
Tree 1: (criterion='entropy', random_state=0)



(if we want to see a bigger picture of the tree, we should put larger values for “figsize” in “figure” method.)

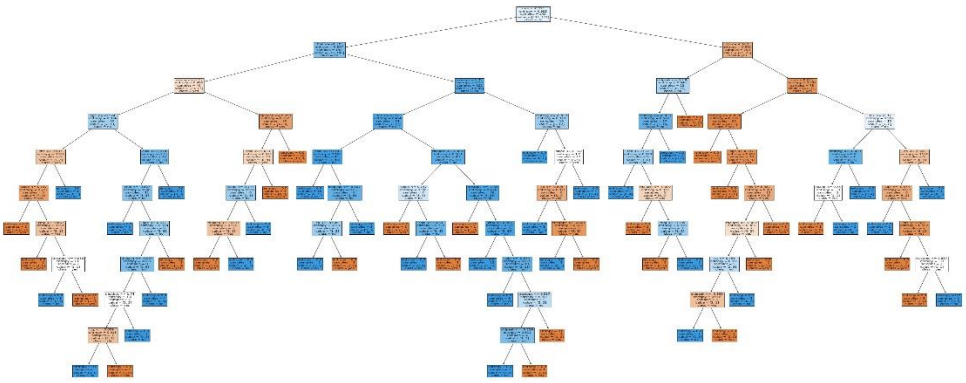
In part (3), I changed decision tree’s parameters and performed the same process again so that I can compare other possible trees by my model and also to see if I can reach higher accuracy. Here are the other 4 trees I made:

Tree 2: (criterion='gini', max_depth=3, min_samples_split=2, random_state=0)



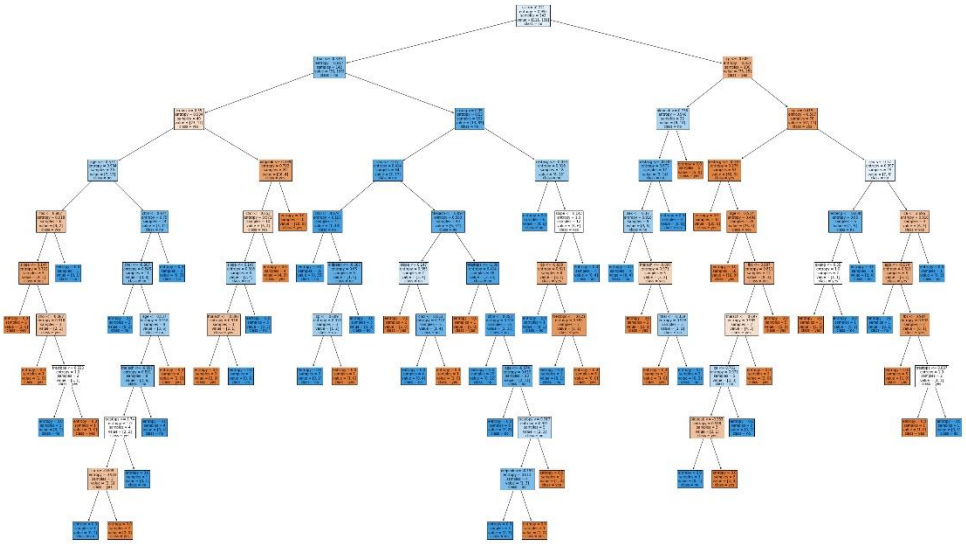
And the accuracy became “80”.

Tree 3: (criterion='gini', max_depth=3, min_samples_split=100, random_state=0)



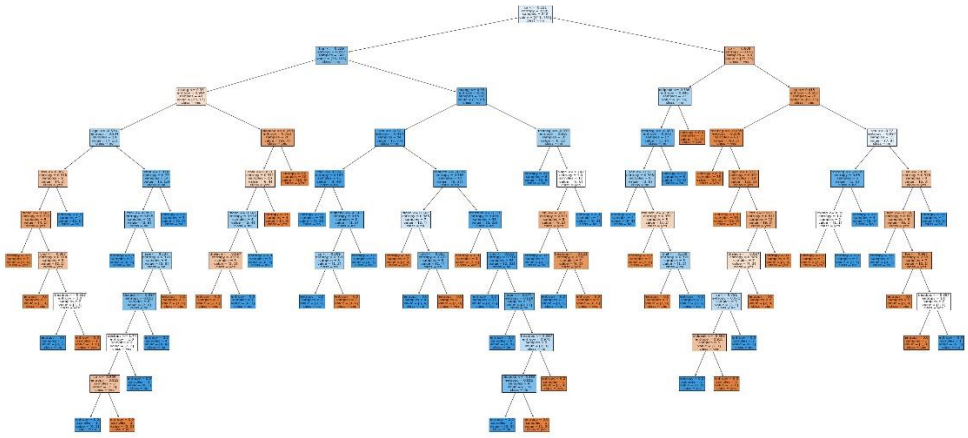
And the accuracy became 77%.

Tree 4: (criterion='gini', max_depth=20, min_samples_split=2, random_state=0)



And the accuracy became 86%.

Tree 5: (criterion='entropy', max_depth=20, min_samples_split=2)



And the accuracy became 77%.

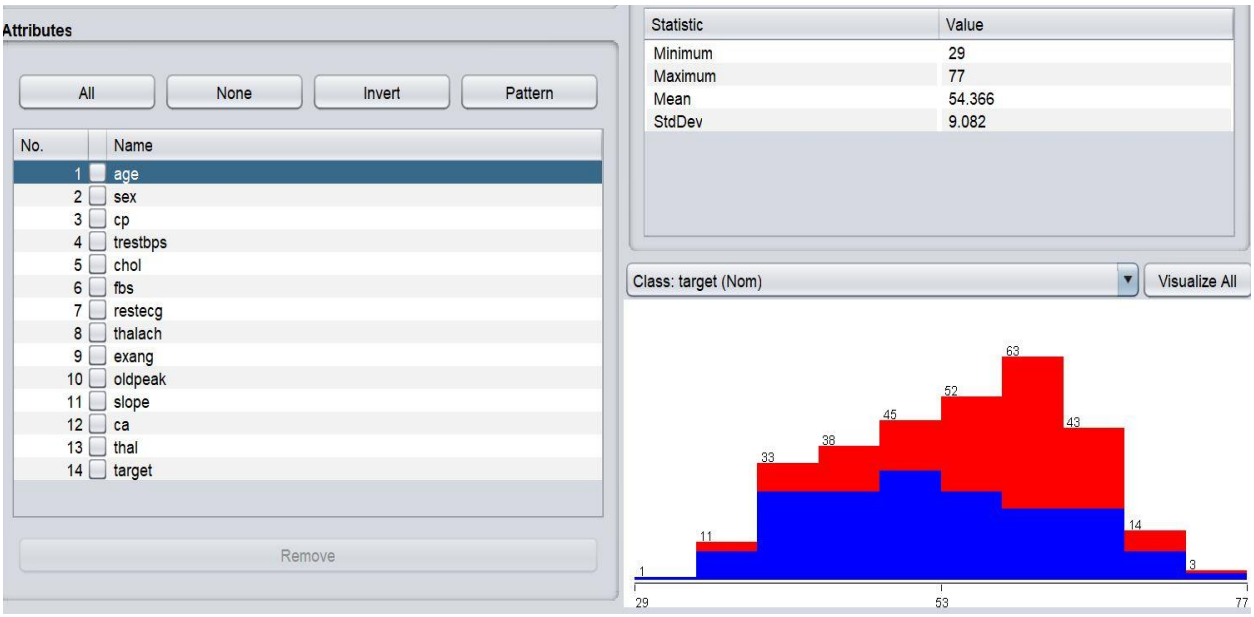
Therefore, we can conclude that by using “ criterion = ‘gini’ ” instead of “ ’entropy’ ”, and choosing a larger value for “”max_depth”, we will have higher accuracy for the decision tree classification model.

Implementation 2: Decision Tree with Weka

(The name of the file: heart.arff, 1.jpg)

In this part, I used software “Weka” to build and visualize the decision tree.

First, I uploaded my “.csv” file to Weka and changed its format to “.arff”. This new file is what we should work with in Weka. After that, we should delete the first column, since it does not hold any valuable information and is simply the number of the data that we have, so I removed it. And these are the features that Weka is going to use ti build the decision tree:



Then we should go to “classify” tab. In the top, we should choose “trees -> J48” (by default the setting is: “j48 –C 0.25 –M 2”). Then we hit “start” and our decision tree model will be made in the right side of the screen:

=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2

Relation: heart-weka.filters.unsupervised.attribute.Remove-R1

Instances: 303

Attributes: 14

age

sex

cp

trestbps

chol

fbs

restecg

thalach

exang

oldpeak

slope

ca

thal

target

Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree

---

thal = normal

| ca <= 0

| | exang = no: yes (6.0)

| | exang = yes: no (4.0/1.0)

| ca > 0: no (10.0)

thal = fixed_defect

| ca <= 0: yes (114.0/12.0)

| ca > 0

```
|   |   cp = severe
|   |   |   trestbps <= 138: no (4.0/1.0)
|   |   |   trestbps > 138: yes (3.0)
|   |   cp = medium: yes (16.0/1.0)
|   |   cp = weak
|   |   |   restecg <= 0
|   |   |   |   exang = no: no (3.0)
|   |   |   |   exang = yes: yes (2.0)
|   |   |   restecg > 0: yes (4.0)
|   |   cp = none
|   |   |   sex = male: no (14.0)
|   |   |   sex = female
|   |   |   |   slope <= 1: no (4.0/1.0)
|   |   |   |   slope > 1: yes (2.0)
thal = eversable_defect
|   cp = severe
|   |   chol <= 229: yes (3.0)
|   |   chol > 229
|   |   |   age <= 48: no (2.0)
|   |   |   age > 48: yes (3.0/1.0)
|   cp = medium
|   |   slope <= 1
|   |   |   ca <= 0
|   |   |   |   trestbps <= 122: yes (4.0)
|   |   |   |   trestbps > 122: no (3.0)
|   |   |   ca > 0: no (8.0/1.0)
|   |   slope > 1: yes (7.0/1.0)
|   cp = weak
|   |   oldpeak <= 0.7: yes (7.0/2.0)
|   |   oldpeak > 0.7: no (2.0)
```

```
|   cp = none
|   |   oldpeak <= 0.6
|   |   |   restecg <= 0: no (8.0/1.0)
|   |   |   restecg > 0
|   |   |   |   trestbps <= 136
|   |   |   |   |   ca <= 0: yes (4.0)
|   |   |   |   |   ca > 0
|   |   |   |   |   |   thalach <= 151: yes (2.0)
|   |   |   |   |   |   thalach > 151: no (3.0)
|   |   |   |   trestbps > 136: no (4.0)
|   |   oldpeak > 0.6: no (57.0)

Number of Leaves  :   28

Size of the tree :        50


Time taken to build model: 0.06 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         243               80.198  %
Incorrectly Classified Instances        60               19.802  %
Kappa statistic                          0.6013
Mean absolute error                      0.2423
Root mean squared error                  0.4225
Relative absolute error                 48.8423 %
Root relative squared error             84.8383 %
Total Number of Instances              303
```

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.812	0.210	0.822	0.812	0.817	0.601	0.801	0.763	yes
	0.790	0.188	0.779	0.790	0.784	0.601	0.801	0.752	no
Weighted Avg.	0.802	0.200	0.802	0.802	0.802	0.601	0.801	0.758	

=== Confusion Matrix ===

a	b	<-- classified as
134	31	a = yes
29	109	b = no

As we can see, the accuracy is nearly 80%.

So now, we should change confidenceFactor parameter to 0.05 for the first run and see the results again:

=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.05 -M 2

Relation: heart-weka.filters.unsupervised.attribute.Remove-R1

Instances: 303

Attributes: 14

age

sex

cp

trestbps

chol

fbs

restecg

thalach

exang

oldpeak

slope

ca

thal

target

Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree

```
thal = normal
|   ca <= 0
|   |   exang = no: yes (6.0)
|   |   exang = yes: no (4.0/1.0)
|   ca > 0: no (10.0)
thal = fixed_defect
|   ca <= 0: yes (114.0/12.0)
|   ca > 0
|   |   cp = severe
|   |   |   trestbps <= 138: no (4.0/1.0)
|   |   |   trestbps > 138: yes (3.0)
|   |   cp = medium: yes (16.0/1.0)
|   |   cp = weak
|   |   |   restecg <= 0
|   |   |   |   exang = no: no (3.0)
|   |   |   |   exang = yes: yes (2.0)
|   |   |   restecg > 0: yes (4.0)
|   |   cp = none: no (20.0/3.0)
thal = eversable_defect
|   cp = severe: yes (8.0/3.0)
|   cp = medium
|   |   slope <= 1
|   |   |   ca <= 0
|   |   |   |   trestbps <= 122: yes (4.0)
|   |   |   |   trestbps > 122: no (3.0)
|   |   |   ca > 0: no (8.0/1.0)
|   |   slope > 1: yes (7.0/1.0)
|   cp = weak
|   |   oldpeak <= 0.7: yes (7.0/2.0)
|   |   oldpeak > 0.7: no (2.0)
```

| cp = none: no (78.0/7.0)

Number of Leaves : 19

Size of the tree : 32

Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	234	77.2277 %
Incorrectly Classified Instances	69	22.7723 %
Kappa statistic	0.5384	
Mean absolute error	0.2925	
Root mean squared error	0.4329	
Relative absolute error	58.9649 %	
Root relative squared error	86.9285 %	
Total Number of Instances	303	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.818	0.283	0.776	0.818	0.796	0.539	0.787	0.759	yes
	0.717	0.182	0.767	0.717	0.742	0.539	0.787	0.710	no
Weighted Avg.	0.772	0.237	0.772	0.772	0.771	0.539	0.787	0.736	

=== Confusion Matrix ===

```
  a   b   <-- classified as
135  30 |   a = yes
 39  99 |   b = no
```

As we can see, the accuracy reduced to 77%.

Then, we should change confidenceFactor parameter to 0.35 for the second run and classify the data again:

=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.35 -M 2

Relation: heart-weka.filters.unsupervised.attribute.Remove-R1

Instances: 303

Attributes: 14

age

sex

cp

trestbps

chol

fbs

restecg

thalach

exang

oldpeak

slope

ca

thal

target

Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree

```
thal = normal
|   ca <= 0
|   |   exang = no: yes (6.0)
|   |   exang = yes: no (4.0/1.0)
|   ca > 0: no (10.0)
thal = fixed_defect
|   ca <= 0: yes (114.0/12.0)
|   ca > 0
|   |   cp = severe
|   |   |   trestbps <= 138: no (4.0/1.0)
|   |   |   trestbps > 138: yes (3.0)
|   |   cp = medium: yes (16.0/1.0)
|   |   cp = weak
|   |   |   restecg <= 0
|   |   |   |   exang = no: no (3.0)
|   |   |   |   exang = yes: yes (2.0)
|   |   |   restecg > 0: yes (4.0)
|   |   cp = none
|   |   |   sex = male: no (14.0)
|   |   |   sex = female
|   |   |   |   slope <= 1: no (4.0/1.0)
|   |   |   |   slope > 1: yes (2.0)
thal = eversable_defect
|   cp = severe
|   |   chol <= 229: yes (3.0)
|   |   chol > 229
|   |   |   age <= 48: no (2.0)
|   |   |   age > 48: yes (3.0/1.0)
|   cp = medium
|   |   slope <= 1
```

| | | ca <= 0
| | | | trestbps <= 122: yes (4.0)
| | | | trestbps > 122: no (3.0)
| | | ca > 0: no (8.0/1.0)
| | slope > 1: yes (7.0/1.0)
| cp = weak
| | oldpeak <= 0.7: yes (7.0/2.0)
| | oldpeak > 0.7: no (2.0)
| cp = none
| | oldpeak <= 0.6
| | | restecg <= 0: no (8.0/1.0)
| | | restecg > 0
| | | | trestbps <= 136
| | | | | ca <= 0: yes (4.0)
| | | | | ca > 0
| | | | | | thalach <= 151: yes (2.0)
| | | | | | thalach > 151: no (3.0)
| | | | trestbps > 136: no (4.0)
| | oldpeak > 0.6: no (57.0)

Number of Leaves : 28

Size of the tree : 50

Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	242	79.868 %
Incorrectly Classified Instances	61	20.132 %
Kappa statistic	0.5958	
Mean absolute error	0.2345	
Root mean squared error	0.4263	
Relative absolute error	47.2595 %	
Root relative squared error	85.5858 %	
Total Number of Instances	303	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.794	0.196	0.829	0.794	0.811	0.596	0.812	0.783	yes
	0.804	0.206	0.766	0.804	0.784	0.596	0.812	0.751	no
Weighted Avg.	0.799	0.200	0.800	0.799	0.799	0.596	0.812	0.769	

=== Confusion Matrix ===

a	b	<-- classified as
131	34	a = yes
27	111	b = no

And the accuracy increased to 79%.

Implementation 3: Naïve Bayes Text Classification

(The name of the file: Implementation3.ipynb)

In part (1), after loading the dataset, I changed all the words in ‘sw.tst’ and 'reviews_train.csv' to lowercase. Then, I separated words based on their “pos” or “neg” classes into “positive_words” and “negative_words”. I also made a “stop_words” array for the stop words the in file ‘sw.txt. Then, I deleted punctuations from positive_words” and “negative_words”. Then I converted these two arrays of positive and negative sentences to arrays of positive and negative words by using nltk’s method “.tokenize.word_tokenize()”. I also commented a cell that deletes words with lengths less than 1, such as ‘6’ or ‘a’. Then I eliminated the stop words that are in positive and negative arrays. After that, I made a dictionary of both positive and negative words by using “collections.Counter” method. Next, I calculated the total number of positive and negative words in addition to the probability of the positive and the negative classes separately.

After preparing the data, in part (2) we should go through all the negative and positive samples in the test set “reviews_test.csv” and classify each one according to the parameters done earlier in part (1). The classification is done by comparing the log posterior P(X|Y)P(Y) for both classes. Therefor for this section, I did the same data preparation on the test set and calculated the osterior probability by:

```
probability_positive = (positive_dic[str(word)] + a) / (num_dic_positive + a*d)
total_probability_positive *= probability_positive



pr_positive = np.append(pr_positive, (total_probability_positive * p_class_positi
ve))
pr_negative = np.append(pr_negative, (total_probability_negative * p_class_negati
ve))
```

But part (2) will yield 0 answer since if one of the probabilities is very low, then the answer and thus the accuracy become very little, if not 0. This is an issue with Naïve Bayes classifier that when a test sample contains a word which is not present in the dictionary, P(word|class) equals to zero. To mitigate this issue, one solution is to employ Laplace Smoothing (it has a parameter α) to enhance our classifier.

Then after finding and matching the predicted positive or negative class for each review in the test set, based on the calculated probability for each one (as in the above code lines), I concatenated the predicted and true class labels for all the data in the test set.

Then, I calculated the confusion matrix and accuracy to assess the model’s performance. The accuracy became 0.6 or 60%.

```
Out[137]: 0.6
```

And as a comparison, the classifier results with and without Laplace Smoothing and using different values for α, varies. If α=0 then the answer becomes 0. So we have to set α as 1 or higher. (I used α=1, which is what we should set for a typical Laplace Smoothing classifier method.) I also tried α=3, 10 but the accuracy didn't change. For α=50, accuracy reduced to 55%. Therefor in Naïve Bayes classifier, we should generally set α a non-zero number.