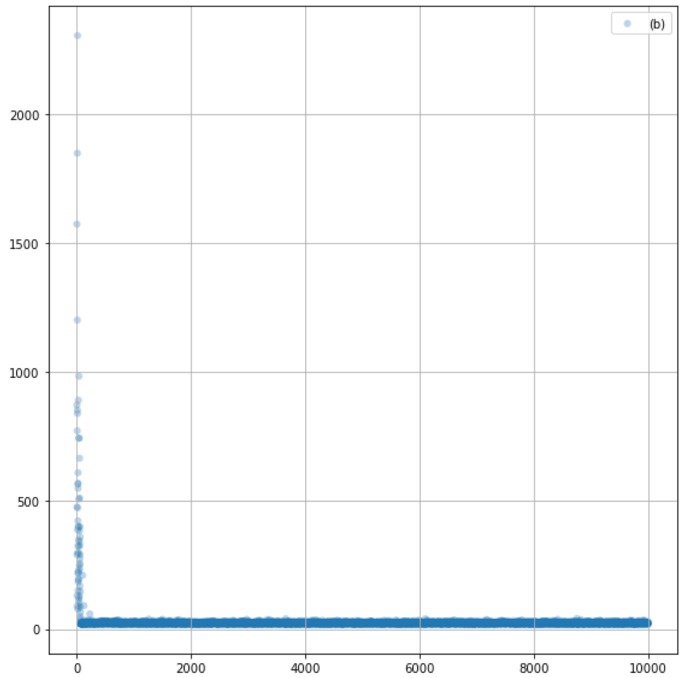
Applied Machine Learning
Post Block Assignment 2 19768206

1.

a) States = 176 white spaces
Actions = 4 actions per state
176*4 = 704



b) Scatter plot above shows training progression

c) Path:

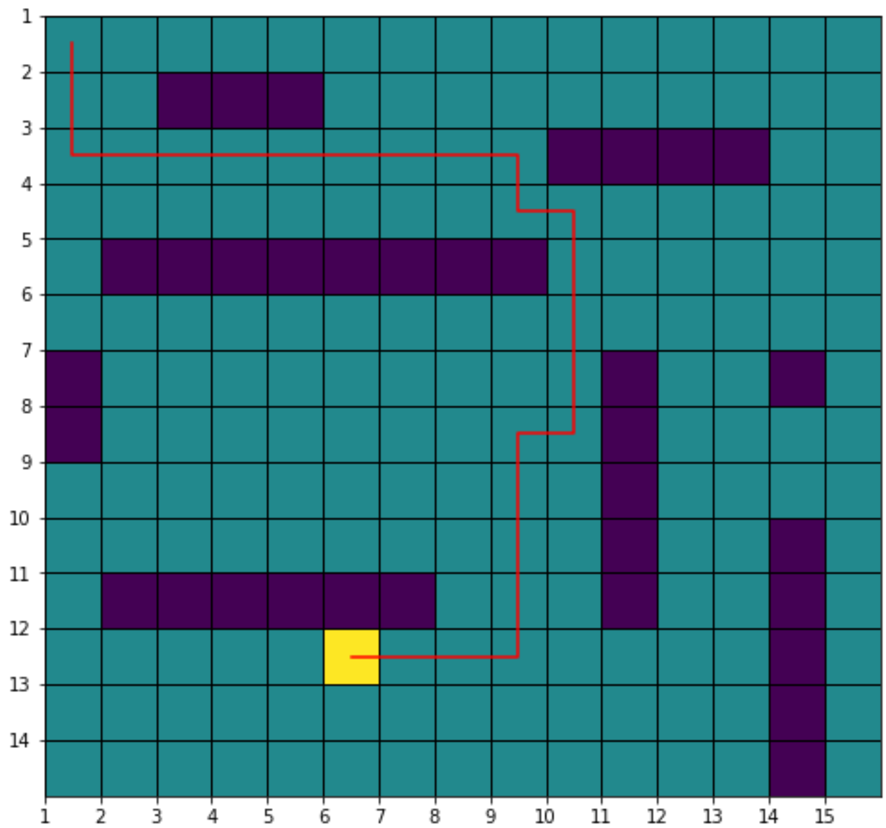


Table:

State	chosen action	up	right	down	left
State = 1 1	down	0.29198902	0.30735687	0.30735687	0.29198902
State = 2 1	down	0.29198902	0.32353354	0.32353354	0.30735687
State = 3 1	right	0.30735687	0.34056163	0.30735687	0.32353354
State = 3 2	right	0.32353354	0.35848592	0.32353354	0.32353354
State = 3 3	right	0.35848592	0.3773536	0.34056163	0.34056163
State = 3 4	right	0.3773536	0.39721432	0.35848592	0.35848592
State = 3 5	right	0.39721432	0.41812034	0.3773536	0.3773536
State = 3 6	right	0.39721432	0.44012667	0.39721432	0.39721432
State = 3 7	right	0.41812034	0.46329123	0.3773536	0.41812034
State = 3 8	right	0.44012667	0.48767498	0.44012667	0.44012667
State = 3 9	down	0.46329123	0.48767498	0.51334208	0.46329123
State = 4 9	right	0.48767498	0.54036009	0.51334208	0.44012667
State = 4 10	down	0.54036009	0.51334208	0.56880009	0.51334208
State = 5 10	down	0.54036009	0.54036009	0.59873694	0.56880009
State = 6 10	down	0.56880009	0.51334208	0.63024941	0.56880009
State = 7 10	down	0.59873694	0.63024941	0.66342043	0.59873694
State = 8 10	left	0.63024941	0.66342043	0.63024941	0.6983373
State = 8 9	down	0.59873694	0.66342043	0.73509189	0.66342043
State = 9 9	down	0.6983373	0.63024941	0.77378094	0.6983373
State = 10 9	down	0.73509189	0.73509189	0.81450625	0.81450625
State = 11 9	down	0.77378094	0.77378094	0.857375	0.857375
State = 12 9	left	0.81450625	0.81450625	0.81450625	0.9025
State = 12 8	left	0.857375	0.857375	0.857375	0.95
State = 12 7	left	0.95	0.9025	0.9025	1
State = 12 6	arrived	0	0	0	0

d) With a reward of -0.05 per square entered as orange:
The performance with the negative reward is slightly better per episode finishing the 10 000 episodes in 16 seconds versus the 22 seconds taken by the Q learning algorithm with only a reward at final state. We can see this in the scatter plot with the orange values consistently being slightly lower than the blue values.

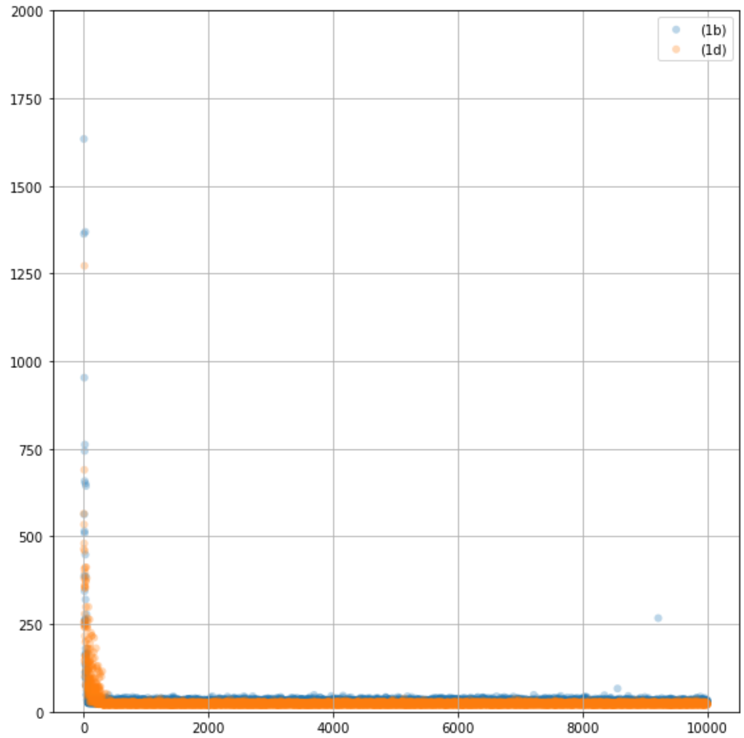
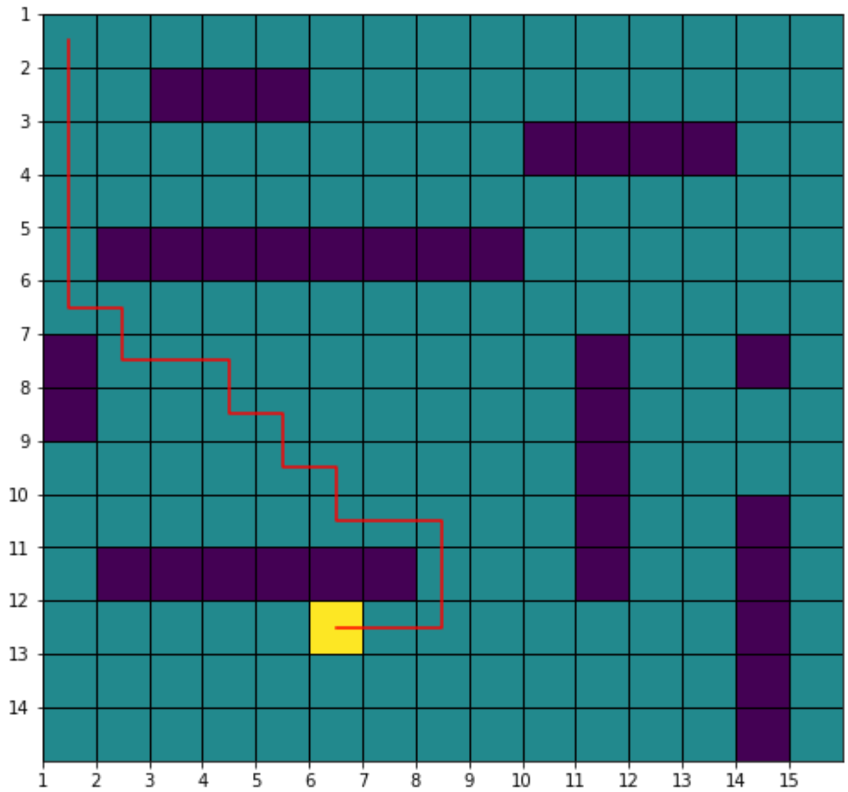


Table for orange:

State	chosen action	up	right	down	left
State = 1 1	down	-0.2830282	-0.3188767	-0.2452928	-0.2830282
State = 2 1	down	-0.2830282	-0.2830282	-0.2055714	-0.2452928
State = 3 1	down	-0.2452928	-0.2452928	-0.1637593	-0.2055714
State = 4 1	down	-0.2055714	-0.2055714	-0.1197467	-0.1637593
State = 5 1	down	-0.1637593	-0.1197467	-0.0734175	-0.1197467
State = 6 1	right	-0.1197467	-0.02465	-0.0734175	-0.0734175
State = 6 2	down	-0.02465	0.02668417	0.02668417	-0.0734175
State = 7 2	down	-0.02465	0.08072018	0.08072018	0.02668417
State = 8 2	right	0.02668417	0.13760018	0.13760018	0.08072018
State = 8 3	down	0.08072018	0.19747388	0.19747388	0.08072018
State = 9 3	right	0.13760018	0.26049882	0.13760018	0.13760018
State = 9 4	right	0.19747388	0.32684086	0.32684086	0.19747388
State = 9 5	right	0.26049882	0.39667459	0.39667459	0.26049882
State = 9 6	down	0.32684086	0.47018378	0.47018378	0.32684086
State = 10 6	right	0.39667459	0.54756187	0.47018378	0.39667459
State = 10 7	right	0.47018378	0.6290125	0.54756187	0.47018378
State = 10 8	down	0.54756187	0.54756187	0.71475	0.54756187
State = 11 8	down	0.6290125	0.6290125	0.805	0.71475
State = 12 8	left	0.71475	0.71475	0.71475	0.9
State = 12 7	left	0.9	0.805	0.805	1
State = 12 6	arrived	0	0	0	0

e) Changing the formula to include a -1 reward for running into a wall results in even faster training times but not by much, 14 seconds versus the previous 16. In this version the agent is also terminated when hitting a wall, starting the next episode. The agent clearly takes shorter to complete an episode than the previous methods for early iterations, this is most likely due to it impacting the wall and being prematurely terminated before it reaches the end goal. This helps the agent converge sooner towards an optimal Q map. Through multiple runs of 10 000 episodes this agent is consistently faster than the previous version. This advantage would only grow with larger and more complicated mazes.

The path the agent followed is as below:



With the state action table as follows:

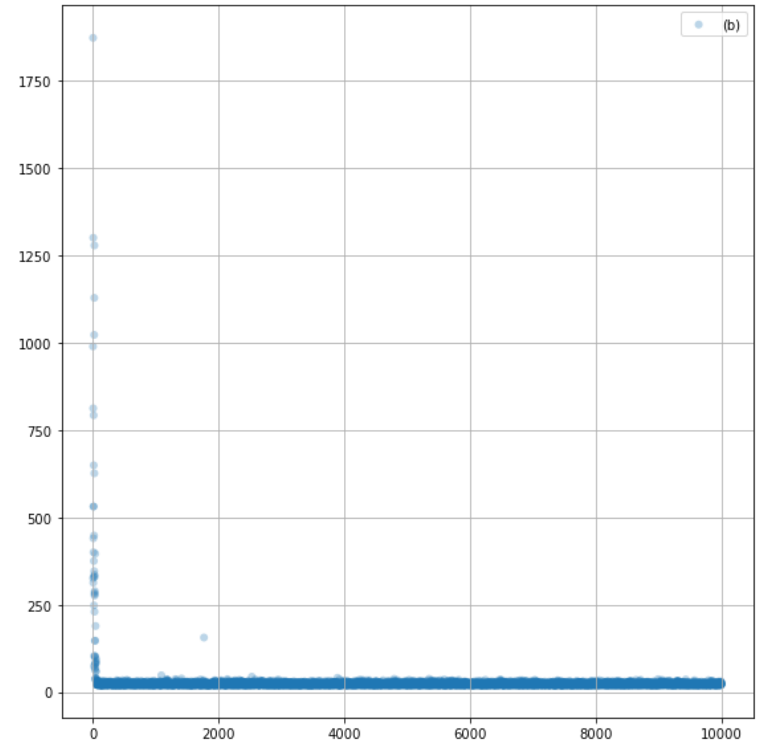
State	chosen action	up	right	down	left
State = 1 1	down	-0.2830282	-0.3188767	-0.2452928	-0.2830282
State = 2 1	down	-0.2830282	-0.2830282	-0.2055714	-0.2452928
State = 3 1	down	-0.2452928	-0.2452928	-0.1637593	-0.2055714
State = 4 1	down	-0.2055714	-0.2055714	-0.1197467	-0.1637593
State = 5 1	down	-0.1637593	-1	-0.0734175	-0.1197467
State = 6 1	right	-0.1197467	-0.02465	-1	-0.0734175
State = 6 2	down	-1	0.02668417	0.02668417	-0.0734175
State = 7 2	right	-0.02465	0.08072018	0.08072017	-1
State = 7 3	right	0.02668417	0.13760018	0.13760018	0.02668417
State = 7 4	down	0.08072017	0.19747388	0.19747388	0.08072018
State = 8 4	right	0.13760018	0.26049882	0.26049882	0.13760018

State = 8 5	down	0.19747388	0.32684086	0.32684086	0.19747388
State = 9 5	right	0.26049882	0.39667459	0.39667459	0.26049882
State = 9 6	down	0.32684086	0.47018378	0.47018378	0.32684086
State = 10 6	right	0.39667459	0.54756187	-1	0.39667459
State = 10 7	right	0.47018378	0.6290125	-1	0.47018378
State = 10 8	down	0.54756187	0.54756187	0.71475	0.54756187
State = 11 8	down	0.6290125	0.62901248	0.805	-1
State = 12 8	left	0.71475	0.71475	0.71475	0.9
State = 12 7	left	-1	0.805	0.805	1
State = 12 6	arrived	0	0	0	0

The agent took quicker to reach the goal in this attempt versus the first version where there was only the +1 reward at the final state, 21 steps versus 25.

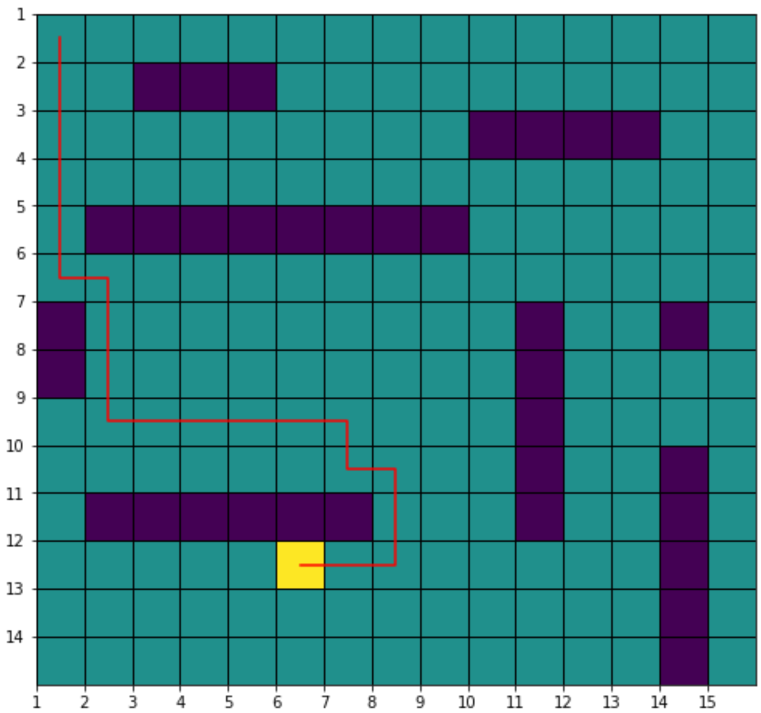
2.

a. The episodes for sarsa were terminated after 2200 steps were taken per episode, this was to prevent the training from taking too much time and even with the imposed limit the training still took 20 minutes to complete. There is no apparent convergence occuring in the figure showing number of steps taken per episode.



b. The SARSA agent achieves convergence early on in the episodes (from studying the image it appears convergence is achieved earlier than the base Q learning) but still takes longer to complete its 10 000 episodes at 29 seconds vs the Q learning at 22 seconds with similar reward structure.

The path followed by the SARSA agent is as follows:



Q Table:

State	chosen action	up	right	down	left
State = 1 1	down	0.30094312	0.27383413	0.31690709	0.3028594
State = 2 1	down	0.30196595	0.28500967	0.33212325	0.31836899
State = 3 1	down	0.32289028	0.32083875	0.35363885	0.33919841
State = 4 1	down	0.33806029	0.32332235	0.37715413	0.36092825
State = 5 1	down	0.3594483	0.38279529	0.40227572	0.37848978
State = 6 1	right	0.37908969	0.42882723	0.40505756	0.4077948
State = 6 2	down	0.42971416	0.40147192	0.45351219	0.40928447
State = 7 2	down	0.42985019	0.44968845	0.48060424	0.45726595
State = 8 2	down	0.45138622	0.48981013	0.50465674	0.47681997
State = 9 2	right	0.48329835	0.52713611	0.47300872	0.48029869
State = 9 3	right	0.47402907	0.55453121	0.47375236	0.50766356
State = 9 4	right	0.52998865	0.60572707	0.54310859	0.54219504
State = 9 5	right	0.56292378	0.64742536	0.47513683	0.56909591
State = 9 6	right	0.52038043	0.68228118	0.61352509	0.61032206
State = 9 7	down	0.65252339	0.64926151	0.72952737	0.61949146
State = 10 7	right	0.69571194	0.78199567	0.73247131	0.60803698
State = 10 8	down	0.66111721	0.68696462	0.83236244	0.72491869
State = 11 8	down	0.78223323	0.75856926	0.86944563	0.83099116
State = 12 8	left	0.83678461	0.73973734	0.79225994	0.94049158
State = 12 7	left	0.93655321	0.88998474	0.85719691	1
State = 12 6	arrived	0	0	0	0

A strange benefit noticed in this exercise is the lack of equal q values for actions per state. Some states in Q learning have Q values of equal value, this makes reproducible “best paths” difficult.

3. Q learning achieves convergence consistently in all implementations. The SARSA implementation was able to converge faster but took longer to complete all of its episodes, probably due to the added step of getting the correct action for the next state as well as the current state. Q learning achieved similar length routes to SARSA.
4. Start the agent much closer to the final reward state (1 block/state away) and for every episode move it to another block of equal Manhattan distance to final target state (avoiding placing the start state in walls), once all blocks within a certain Manhattan distance have been used increase the Manhattan distance by 1 and continue. This will achieve convergence sooner.
5. –
6. –

Section 3:

1.

		Positive					
danger	is	getaway	lawyer	critic	crazy	powerful	400
235	312	12	135	89	180	375	
		Negative					
danger	is	getaway	lawyer	critic	crazy	powerful	1100
412	637	122	48	102	99	357	
							1500

Danger is powerful:
(400/1500)*[(danger*is*powerful)/(400*400*400)] = 0.1145625 for positive
(1100/1500)*[(danger*is*powerful)/(1100*1100*1100)] = 0.00054127 for negative
Therefore target is positive.

Powerful lawyer is crazy:
(400/1500)*[(powerful*lawyer*is*crazy)/(400*400*400*400)] = 0.02961563 for positive
(1100/1500)*[(powerful*lawyer*is*crazy)/(1100^4)] = 0.00054127 for negative
Therefore target is positive

2.

Question3

June 5, 2021

[1]:
```
import numpy as np
import pandas as pd
import sklearn
from sklearn import model_selection, feature_extraction,metrics, mixture
from sklearn.naive_bayes import MultinomialNB
import matplotlib.pyplot as plt
import scipy.stats as stats
import math
from scipy.optimize import curve_fit
from pylab import *
import seaborn as sns
```

0.1 2. SMSSpam

Importing the data

[2]:
```
sms = pd.read_csv('/Users/jeandre/Desktop/Applied Machine Learning/Post Block␣
↪Assignemnt 1/SMSSpamCollection.txt',delimiter="\t",header=None)
```

some example output, making sure we have the right data

[3]:
```
sms
```

[3]:
```
         0                                                  1
0      ham  Go until jurong point, crazy.. Available only …
1      ham                      Ok lar… Joking wif u oni…
2     spam  Free entry in 2 a wkly comp to win FA Cup fina…
3      ham  U dun say so early hor… U c already then say…
4      ham  Nah I don't think he goes to usf, he lives aro…
…     …                                                  …
5567  spam  This is the 2nd time we have tried 2 contact u…
5568   ham               Will ü b going to esplanade fr home?
5569   ham  Pity, * was in mood for that. So…any other s…
5570   ham  The guy did some bitching but I acted like i'd…
5571   ham                         Rofl. Its true to its name

[5572 rows x 2 columns]
```

[4]:
```
target = sms[0]
words = sms[1]
```

Assign the words and labels to X and Y and perform a test train split for model evaluation.

[5]:
```
X_train, X_test, Y_train, Y_test = sklearn.model_selection.
↪train_test_split(words,target, test_size = 0.3, random_state = 0)
```

[6]:
```
X_train
```

[6]:
```
4380                 How are you. Just checking up on you
3887    Same, I'm at my great aunts anniversary party …
4755                     Ok lor… Or u wan me go look 4 u?
2707    S now only i took tablets . Reaction morning o…
4747             Orh i tot u say she now still dun believe.
                              …
4931    Hi, the SEXYCHAT girls are waiting for you to …
3264                                  So u gonna get deus ex?
1653    For ur chance to win a £250 cash every wk TXT:…
2607    R U &SAM P IN EACHOTHER. IF WE MEET WE CAN GO …
2732    Mm feeling sleepy. today itself i shall get th…
Name: 1, Length: 3900, dtype: object
```

Fitting a count vectorizer on the training data and then passing the test data through so that they share the same vocab.

[7]:
```
Y_train
count_vect = sklearn.feature_extraction.text.CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
X_test_counts = count_vect.transform(X_test)
values = Y_train.unique()
```

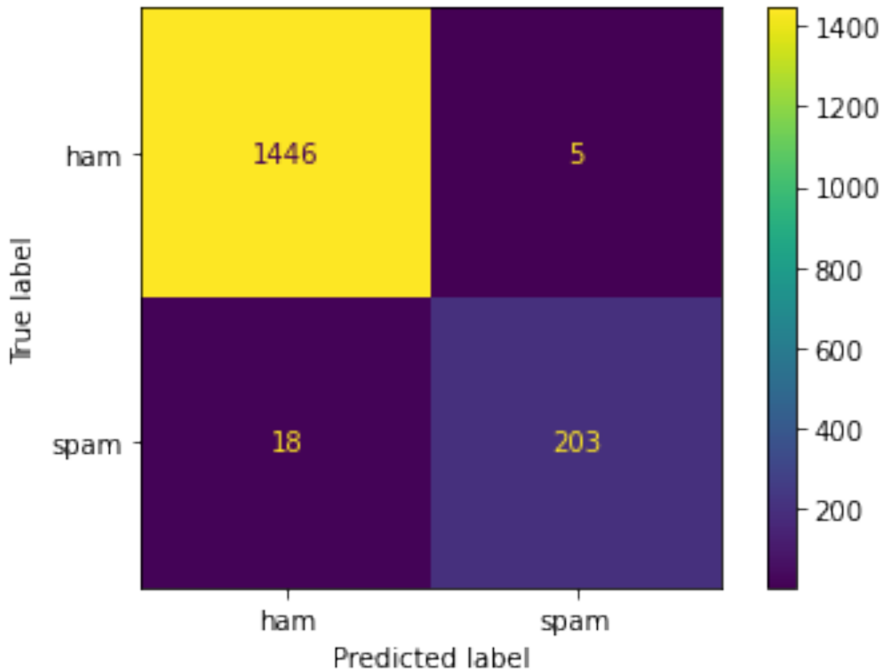
[8]:
```
nb = MultinomialNB()
nb.fit(X_train_counts,Y_train)
N_pred = nb.predict(X_test_counts)
metrics.accuracy_score(Y_test, N_pred)
```

[8]:
```
0.986244019138756
```

Good initial accuracy metric, looking at the confusion matrix we see the pereformance on ham is much better than performance on spam.

[9]:
```
metrics.plot_confusion_matrix(nb,X_test_counts,Y_test,display_labels=values)
```

[9]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f8542cd0e80>
```



0.2 3. GaussianMix

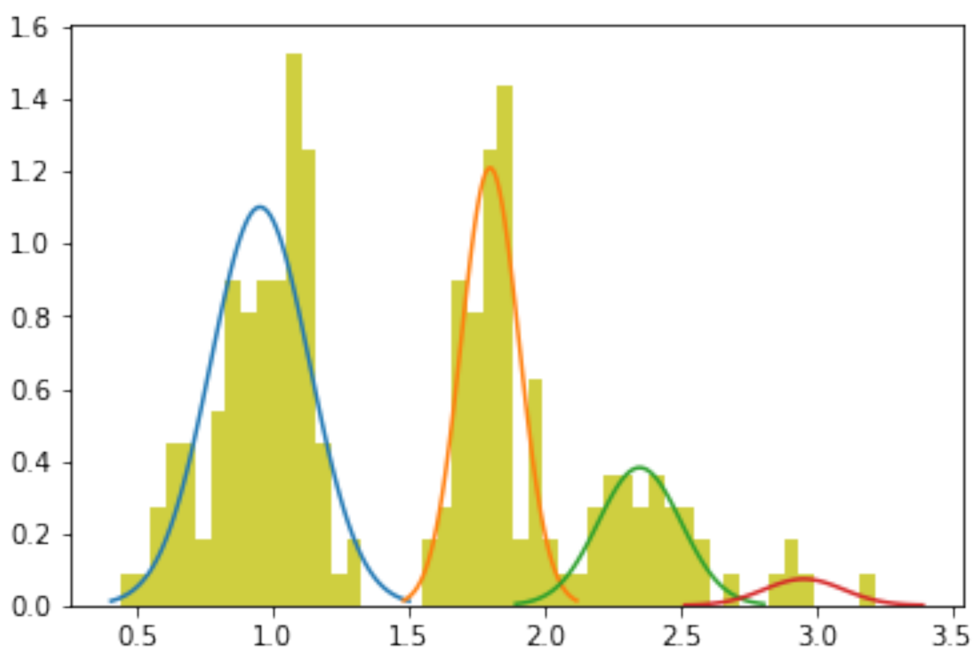
[10]:
```
gauss  = pd.read_csv('/Users/jeandre/Desktop/Applied Machine Learning/Post␣
 ↪Block Assignemnt 1/GaussianMix.csv')
```

[11]:
```
gauss
X = gauss["X"]
```

[12]:
```
X = np.asarray(X).reshape(-1,1) #reshaping since we have one feature
initial_weights = np.array([0.25,0.25,0.25,0.25]) #initial weights provided
initial_means = np.array([
                        [4],
                        [5],
                        [6],
                        [7]]) #initial means provided
initial_cov = np.array([1,1,1,1]) #initial variances, luckily all 1
#mixture.GaussianMixture uses EM in order to fit the gaussians to the data.
gm = mixture.GaussianMixture(covariance_type='spherical',n_components=4,tol=0.
 ↪0001,weights_init=initial_weights,
                                           ␣
 ↪precisions_init=initial_cov,means_init=initial_means)
gm.fit(X)
```

[12]: GaussianMixture(covariance_type='spherical',
                means_init=array([[4.],
       [5.],
       [6.],
       [7.]]),
                n_components=4, precisions_init=array([1., 1., 1., 1.]),
                tol=0.0001, weights_init=array([0.25, 0.25, 0.25, 0.25]))

[13]:
```
graphs = [1,2,3,4]
for k in graphs:
    mu = gm.means_[k-1]
    variance = gm.covariances_[k-1]
    sigma = math.sqrt(variance)
    x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100)
    plt.plot(x, gm.weights_[k-1]*stats.norm.pdf(x, mu, sigma))
n, bins, patches = plt.hist(X, 50, density=True, facecolor='y', alpha=0.75)
plt.show()
```



0.3 4. Cluster

[14]:
```
cluster = pd.read_csv('/Users/jeandre/Desktop/Applied Machine Learning/Post␣
↪Block Assignemnt 1/Cluster.csv')
```

[15]:
```
cluster
```

[15]:
```
              X         Y  Cluster(0)
0    7.345957  8.085866           2
1    2.480907  6.031914           2
2    9.661872  8.127609           1
3    8.046870  6.995351           1
4    7.038915  5.732423           4
..        …         …         …
395  2.056546 -5.661233           4
396  1.223968 -5.976484           2
397  2.020931 -5.966769           1
398  3.405836 -3.875425           2
399  1.828194 -4.709193           3

[400 rows x 3 columns]
```

[16]:
```
X = cluster[list(cluster.columns[0:2])]
```

[17]:
```
counts = cluster["Cluster(0)"].value_counts()
print(counts)
counts = np.asarray(counts)
groups = [2,3,1,4]
weights = []
for index,count in enumerate(counts):
    print("Probability of being in class ",groups[index],"=",count/400)
    weights.append(count/400) #calculate initial weights for use in prior␣
↪assumptions
```

```
2    111
3    100
1     98
4     91
Name: Cluster(0), dtype: int64
Probability of being in class  2 = 0.2775
Probability of being in class  3 = 0.25
Probability of being in class  1 = 0.245
Probability of being in class  4 = 0.2275
```

[18]:
```
# Initial weights is the only assumption made here
#Using mixture.GaussianMixture here again therefore EM again.
gm2 = mixture.GaussianMixture(covariance_type='spherical',n_components=4,tol=0.
↪001,verbose=1,weights_init=weights)
```

[19]:
```
labels = gm2.fit_predict(X)
len(labels)
labels = pd.DataFrame(labels,columns=["labels"])
X = X.merge(labels,left_index=True, right_index=True)
X
```

```
Initialization 0
Initialization converged: True
```

[19]:

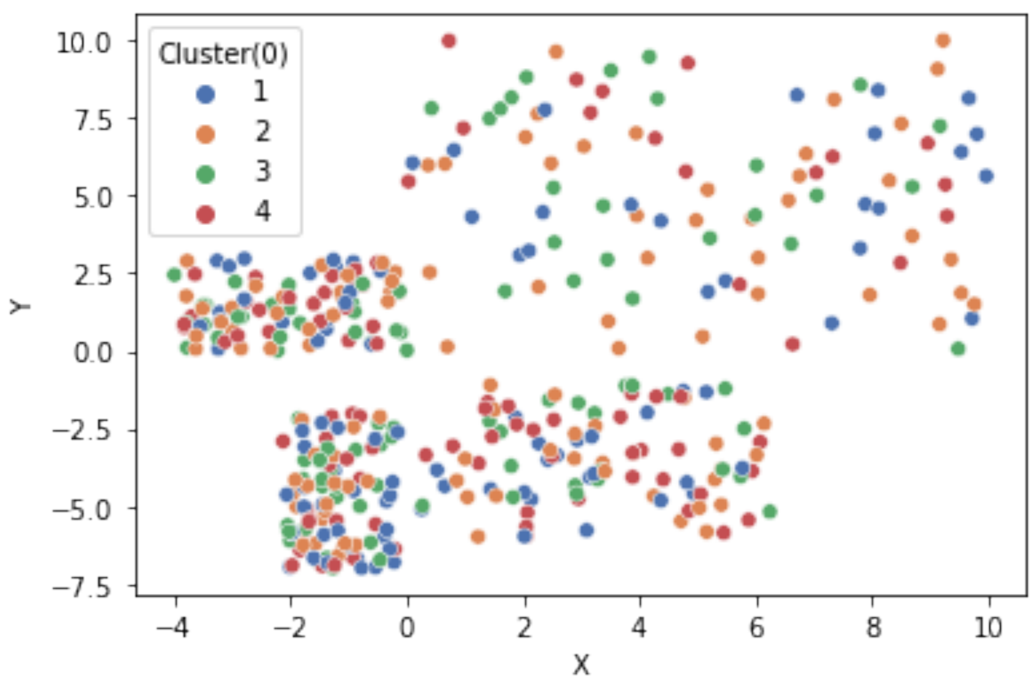
```
              X         Y  labels
0    7.345957  8.085866       0
1    2.480907  6.031914       0
2    9.661872  8.127609       0
3    8.046870  6.995351       0
4    7.038915  5.732423       0
..        ...       ...     ...
395  2.056546 -5.661233       2
396  1.223968 -5.976484       1
397  2.020931 -5.966769       2
398  3.405836 -3.875425       2
399  1.828194 -4.709193       2

[400 rows x 3 columns]
```

[20]:

```python
sns.scatterplot(data=cluster, x="X", y="Y", hue="Cluster(0)",palette="deep") #
↪Initial clusters
```

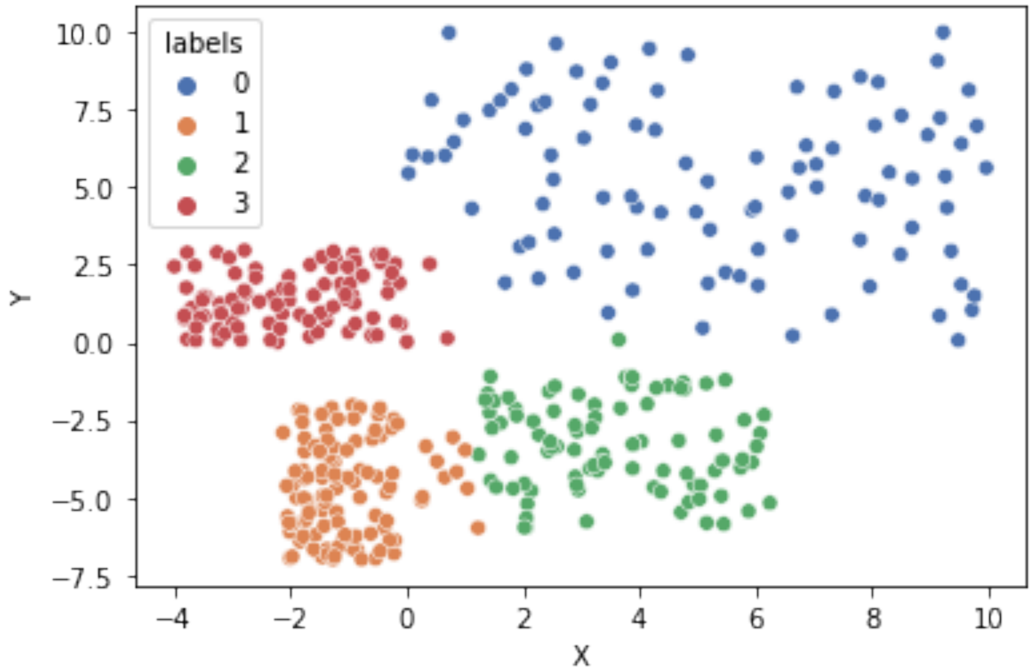
[20]: <AxesSubplot:xlabel='X', ylabel='Y'>



[21]:

```python
sns.scatterplot(data=X, x="X", y="Y", hue="labels",palette="deep") # Fitted
↪cluster
```

[21]: <AxesSubplot:xlabel='X', ylabel='Y'>



[ ]:

CC = Common cold

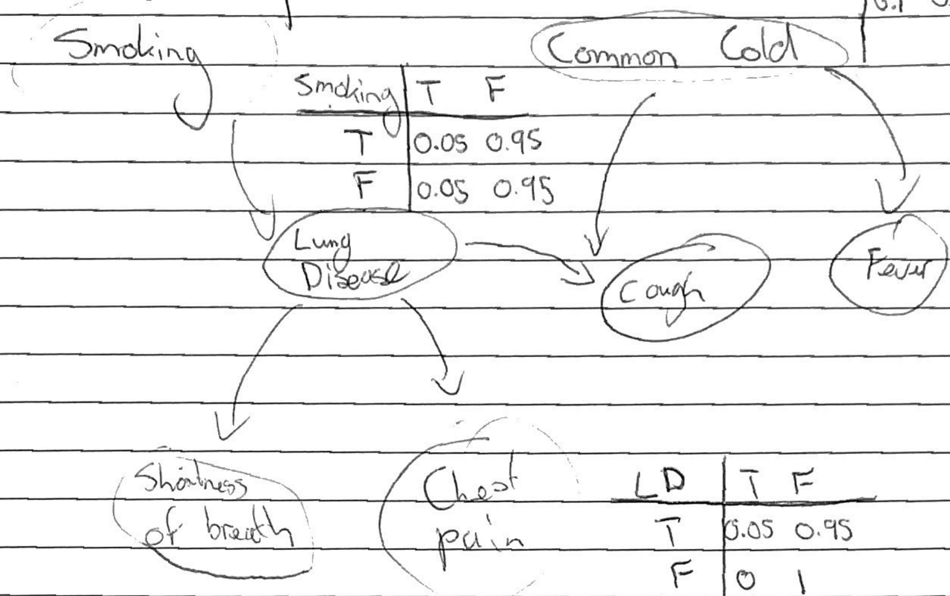
LD = Lung Disease.

DATE

a)

T	F
0.2	0.8

T	F
0.1	0.9



Smoking	T	F
T	0.05	0.95
F	0.05	0.95

CC	T	F
T	0.05	0.95
F	0.2	0.8

b)

LD	T	F
T	0.05	0.95
F	0	1

LD	T	F
T	0.05	0.95
F	0.05	0.95

Cough:

CC	LD	T	F
T	F	0.05	0.95
F	T	0.1	0.9
F	F	0.35	0.65

c) non-smoking lung disease = ~~0.05~~ 0.8×0.05

no cold = 0.9

value for cold FT

$0.8 \times 0.05 \times 0.9 \times (0.1)$

$= 9/2500 = 0.0036 = 0.36\%$

Therefore will predict false

Typo