HW-4 Probability Based Learning

Ques 2. The table below gives details of symptoms that patients presented and whether they were suffering from meningitis.

ID	HEADACHE	FEVER	VOMITING	MENINGITIS
1	true	true	false	false
2	false	true	false	false
3	true	false	true	false
4	true	false	true	false
5	false	true	false	true
6	true	false	true	false
7	true	false	true	false
8	true	false	true	true
9	false	true	false	false
10	true	false	true	true

Using this dataset calculate the following probabilities:

a. **P(VOMITING = true)**
Ans: P(Vomiting= True) = Number of instances where (Vomiting= True) / total number of instances
Vomiting= True in instance : d3,d4,d6,d7,d8,d10
Therefore, **P(Vomiting= True) = 6/10 = 0.6**

b. **P(HEADACHE = false)**
Ans: P(Headache= False) = Number of instances where (Headache= False) / total number of instances
Vomiting= True in instance : d2,d5,d9
Therefore, **P(Vomiting= True) = 3/10 = 0.3**

c. **P(HEADACHE = true, VOMITING = false)**
Ans: P(Headache= True, Vomiting= False) = Number of instances where (Headache= True, Vomiting= False) / total number of instances
Vomiting= True in instance : d1
Therefore, **P(Vomiting= True) = 1/10 = 0.1**
or by using product rule:
P(Headache= True, Vomiting= False): P(Headache= True| Vomiting= False) * P(Vomiting= False)

P(Headache= True| Vomiting= False)= ¼ as only one instance has headache= True when vomiting= False.
P(Vomiting= False)= 4/10 as four instances has vomiting = false out of ten
P(Headache= True, Vomiting= False): 1/4*4/10 = 0.1

d. **P(VOMITING = false | HEADACHE = true)**
Ans: P(Vomiting= False | Headache= True)= ¼ as only one instance has vomiting= False when headache= True= 0.1429

e. **P(MENINGITIS | FEVER = true, VOMITING = false)**
Ans: First we will calculate probability for Meningitis= True:
P(Meningitis= True | Fever= True, Vomiting= False) = 1/4= 0.25
P(Meningitis= False| Fever= True, Vomiting= False) = 3/4= 0.75
P(Meningitis | Fever= True, Vomiting= False) = (0.25,0.75)

Ques 3. Predictive data analytics models are often used as tools for process quality control and fault detection. The task in this question is to create a naive Bayes model to monitor a waste water treatment plant.[33] The table below lists a dataset containing details of activities at a waste water treatment plant for 14 days. Each day is described in terms of six descriptive features that are generated from different sensors at the plant. SS-IN measures the solids coming into the plant per day; SED-IN measures the sediment coming into the plant per day; COND-IN measures the electrical conductivity of the water coming into the plant.[34] The features SS-OUT, SED-OUT, and COND-OUT are the corresponding measurements for the water flowing out of the plant. The target feature, STATUS, reports the current situation at the plant: ok, everything is working correctly; settler, there is a problem with the plant settler equipment; or solids, there is a problem with the amount of solids going through the plant.

ID	SS-IN	SED-IN	COND-IN	SS-OUT	SED-OUT	COND-OUT	STATUS
1	168	3	1,814	15	0.001	1,879	ok
2	156	3	1,358	14	0.01	1,425	ok
3	176	3.5	2,200	16	0.005	2,140	ok
4	256	3	2,070	27	0.2	2,700	ok
5	230	5	1,410	131	3.5	1,575	settler
6	116	3	1,238	104	0.06	1,221	settler
7	242	7	1,315	104	0.01	1,434	settler
8	242	4.5	1,183	78	0.02	1,374	settler
9	174	2.5	1,110	73	1.5	1,256	settler
10	1,004	35	1,218	81	1,172	33.3	solids
11	1,228	46	1,889	82.4	1,932	43.1	solids
12	964	17	2,120	20	1,030	1,966	solids
13	2,008	32	1,257	13	1,038	1,289	solids

a. Create a naive Bayes model that uses probability density functions to model the descriptive features in this dataset (assume that all the descriptive features are normally distributed).

Ans: Probabilities of each target feature levels are:

P(status= ok) =4/13= 0.3077

P(status= settler) = 5/13= 0.3846

P(status= solids) = 4/13= 0.3077

In order to create a naïve bayes model that uses probability density function, we need to fit the normal distribution to each feature for level in the target. we calculated mean and standard deviation for each feature wrt to each level of the target feature. Below Table shows the probability distribution for each feature wrt target level.

Feature	Mean	Standard Deviation
P(SS-IN| OK)	189	45.4166
P(SED-IN| OK)	3.175	0.2363
P(COND-IN| OK)	1860.5	371.4023
P(SS-OUT| OK)	18	6.0553
P(SED-OUT| OK)	0.054	0.0974
P(COND-OUT| OK)	2036	532.1911
P(SS-IN| SETTLERS)	200.8000	55.1289
P(SED-IN| SETTLERS)	4.4600	1.8078
P(COND-IN| SETTLERS	1251.2000	116.2441
P(SS-OUT| SETTLERS)	98.0000	23.3773
P(SED-OUT| SETTLERS)	1.0180	1.5266
P(COND-OUT| SETTLERS)	1372.0000	142.5780
P(SS-IN| SOLIDS)	1301.0000	485.4400
P(SED-IN| SOLIDS)	32.5000	11.9583
P(COND-IN| SOLIDS)	1621.0000	453.0379
P(SS-OUT| SOLIDS)	49.1000	37.7559
P(SED-OUT| SOLIDS)	1293.0000	430.9509
P(COND-OUT| SOLIDS)	832.8500	958.3122

b. What prediction will the naive Bayes model return for the following query?
SS-IN = 222, SED-IN = 4.5, COND-IN = 1,518, SS-OUT = 74 SED-OUT = 0.25, COND-OUT = 1,642

Ans: we will use SciPy library from python to calculate the probability density function (ie by using scipy.stats.norm.pdf)

The calculation for status= ok:

P(SS-IN| OK): mean = 189, stdev= 45.4166 , n= 222 = 0.0067

```
1 round(norm.pdf(222,189, 45.4166 ),4)
```

```
0.0067
```

Likewise the probability density function for the above query are calculated in the table below:

Status	Feature	Mean	Standard Deviation	PDF
OK	P(SS-IN| OK)	189	45.4166	0.0067
OK	P(SED-IN| OK)	3.175	0.2363	$4.3079 \ X10^{-7}$
OK	P(COND-IN| OK)	1860.5	371.4023	0.0007
OK	P(SS-OUT| OK)	18	6.0553	$1.7650 \ X \ 10^{-20}$
OK	P(SED-OUT| OK)	0.054	0.0974	0.54078
OK	P(COND-OUT| OK)	2036	532.1911	0.0006
SETTLERS	P(SS-IN| SETTLERS)	200.8000	55.1289	0.0067
SETTLERS	P(SED-IN| SETTLERS)	4.4600	1.8078	0.2235
SETTLERS	P(COND-IN| SETTLERS	1251.2000	116.2441	0.0002
SETTLERS	P(SS-OUT| SETTLERS)	98.0000	23.3773	0.0101
SETTLERS	P(SED-OUT| SETTLERS)	1.0180	1.5266	0.2303
SETTLERS	P(COND-OUT| SETTLERS)	1372.0000	142.5780	0.0005
SOLIDS	P(SS-IN| SOLIDS)	1301.0000	485.4400	$6.9496x10^{-5}$
SOLIDS	P(SED-IN| SOLIDS)	32.5000	11.9583	0.0022
SOLIDS	P(COND-IN| SOLIDS)	1621.0000	453.0379	0.0009
SOLIDS	P(SS-OUT| SOLIDS)	49.1000	37.7559	0.0085
SOLIDS	P(SED-OUT| SOLIDS)	1293.0000	430.9509	$1.0291x10^{-5}$
SOLIDS	P(COND-OUT| SOLIDS)	832.8500	958.3122	0.0003

Formula to calculate PDF at each target level is :

$$\mathbb{M}(\mathbf{q}) = \underset{l \in levels(t)}{\arg\max} \left(\left(\prod_{i=1}^{m} P(\mathbf{q}[i] \mid t = l) \right) \times P(t = l) \right)$$

For **Status = OK:**

P(Ok|q) = P(ok) * P(SS-IN| OK)* P(SED-IN| OK)* P(COND-IN| OK)* P(SS-OUT| OK)* P(SED-OUT| OK)* P(COND-OUT| OK) = 0.3077 *0.0067* $4.3079 \ X10^{-7}$ * 0.0007 *1.7650 * 10^{-20} * 0.54078*0.0006 = **3.5602 X 10 $^{-36}$**

For **Status = Settlers:**

P(Settlers|q) = P(Settlers) * P(SS-IN| Settlers)* P(SED-IN| Settlers)* P(COND-IN| Settlers)* P(SS-OUT| Settlers)* P(SED-OUT| Settlers)* P(COND-OUT| Settlers) = 0.3846 *0.0067* 0.2235* 0.0002*0.0101* 0.2303*0.0005= **1.3396 X 10 $^{-13}$**

For **Status = Solids:**

P(Solids|q) = P(Solids) * P(SS-IN| Solids)* P(SED-IN| Solids)* P(COND-IN| Solids)* P(SS-OUT| Solids)* P(SED-OUT| Solids)* P(COND-OUT| Solids) = 0.3077 *6.9496x10^{-5}* 0.0022* 0.0009*0.0085* 1.0291x10^{-5}*0.0003= **1.1111 X 10 $^{-21}$**

Here, we have calculated the height of the PDFs instead of calculating actual probabilities for each feature taking a value. The score for each target level is a relative ranking, not a probability. The **target level = Settlers** has the highest value ie. highest ranking. Hence the model will predict that there is a problem with the settler equipment.

Ques 5. The table below lists a dataset containing details of policy holders at an insurance company. The descriptive features included in the table describe each policy holders' ID, occupation, gender, age, the type of insurance policy they hold, and their preferred contact channel. The preferred contact channel is the target feature in this domain.

ID	Occupation	Gender	Age	Policy Type	Pref Channel
1	lab tech	female	43	planC	email
2	farmhand	female	57	planA	phone
3	biophysicist	male	21	planA	email
4	sheriff	female	47	planB	phone
5	painter	male	55	planC	phone
6	manager	male	19	planA	email
7	geologist	male	49	planC	phone
8	messenger	male	51	planB	email
9	nurse	female	18	planC	phone

a. Using equal-frequency binning transform the Age feature into a categorical feature with three levels: young, middle-aged, mature.

Ans: There are 3 bins and 9 instances so 9/3= 3 instance/ bin. We need to create 3 bins with equal frequency.

First we will arrange the AGE feature in ascending order:

Age
18
19
21
43
47

49
51
55
57

Here, we put the first 3 instances into young, then next 3 instances into middle-aged and last three instances with mature. So the data set looks as shown below:

ID	OCCUPATION	GENDER	AGE	POLICY TYPE	PREF. CHANNEL
9	nurse	female	young	planC	phone
6	manager	male	young	planA	email
3	biophysicist	male	young	planA	email
1	lab tech	female	middle-aged	planC	email
4	sheriff	female	middle-aged	planB	phone
7	geologist	male	middle-aged	planC	phone
8	messenger	male	mature	planB	email
5	painter	male	mature	planC	phone
2	armhand	female	mature	planA	phone

Thresholds for different bins are calculated by finding the midpoint between the AGE values of two instances on either side of the boundary.
The threshold value between the young and middle-aged bins = average of instance d1 and d3/ 2 = 21+43/ 2= 32
The threshold value between the middle-aged and mature bins = average of instance d7 and d8 / 2 = 51+49/ 2= 50

b. Examine the descriptive features in the dataset and list the features that you would exclude before you would use the dataset to build a predictive model. For each feature you decide to exclude explain why you have made this decision.
Ans: with respect to descriptive features, the ID feature and the occupation feature will be removed before building the predictive model. The occupation feature is similar to ID feature as the it has different and unique level which are equal to the number of instances. Therefore, it doesn't make sense to include occupation feature in our data set and it should be removed before building predictive model.

c. Calculate the probabilities required by a naive Bayes model to represent this domain.

Ans: Before building the naïve bayes model we need to compute the probabilities of each level in the target feature and the conditional probabilities of each feature for each level in the target feature. Below is the probability calculation

P(pref channel = phone) = 0.56
P(pref channel = email) = 0.44

For pref channel = phone

P(gender= female | phone) = 0.6
P(gender= male | phone) = 0.4
P(Age= young | phone) = 0.2
P(Age= middle-aged| phone) = 0.4
P(Age= mature| phone)= 0.4
P(Policy= PlanA| phone)= 0.2
P(Policy= PlanB | phone)= 0.2
P(Policy= PlanC | phone)= 0.6

For pref channel = email:

P(gender= female | email) = 0.25
P(gender= male | email) = 0.75
P(Age= young | email) = 0.5
P(Age= middle-aged| email) = 0.25
P(Age= mature| email)= 0.25
P(Policy= PlanA| email)= 0.5
P(Policy= PlanB | email)= 0.25
P(Policy= PlanC | email)= 0.25

d. What target level will a naive Bayes model predict for the following query: GENDER = female, AGE = 30, POLICY = planA.

Ans: First we will convert Age feature into categorical value. For this we will compare with the threshold value calculated in part 1. The threshold value for young and middle-age is 32 and the age in query is 30. So age should be mapped to young bin. Hence the new query will be:

GENDER = female, AGE = Young, POLICY = planA.

The calculation for P(Channel = phone |q) is:
P(phone)= 0.56
P(Gender= female |Phone) = 0.6
P(Age= young|Phone)= 0.2
P(Policy = PlanA| Phone)= 0.2

P(phone | q) = P(phone) * P(Gender= female |Phone) * P(Age= young|Phone) * P(Policy = PlanA| Phone) = 0.56* 0.6*0.2* 0.2 = **0.01344**

The calculation for P(Channel = email |q) is:
P(email)= 0.44
P(Gender= female |email) = 0.25
P(Age= young|email)= 0.5
P(Policy = PlanA| email)= 0.5
P(email | q) = P(email) * P(Gender= female |email) * P(Age= young|email) * P(Policy = PlanA| email) = 0.44* 0.25*0.5* 0.5 = **0.0275**
With reference to conditional probabilities calculated above, target level = 'email' has the highest value. Hence model will return the **pref. channel = Email.**

6. Imagine that you have been given a dataset of 1,000 documents that have been classified as being about entertainment or education. There are 700 entertainment documents in the dataset and 300 education documents in the dataset. The tables below give the number of documents from each topic that a selection of words occurred in.

Word-document counts for the *entertainment* dataset

fun	is	machine	christmas	family	learning
415	695	35	0	400	70

Word-document counts for the *education* dataset

fun	is	machine	christmas	family	learning
200	295	120	0	10	105

a. What target level will a naive Bayes model predict for the following query document: "machine learning is fun"?
Ans: The naïve bayes model will label the query that has the highest probability under the assumption of conditional independence. For this, we will calculate the probability as each target level. Hence to calculate the conditional probability we will divide count by the total number of documents in each category.
Query is : **"machine learning is fun"**

P(entertainment) = 700/1000= 0.7
P(education) = 300/1000= 0.3

For target level = Entertainment

word	**count**	**P(WORD|ENTERTAINMENT)**
fun	415	415/700 = 0.5929
is	695	695/700= 0.9929
machine	35	35/700= 0.05
Christmas	0	0
family	400	400/700= 0.5714
learning	70	70/700= 0.1

For target level = Education

word	count	P(WORD|ENTERTAINMENT)
fun	200	200/300 = 0.6667
is	295	295/300= 0.9867
machine	120	120/300= 0.4
Christmas	0	0
family	10	10/300= 0.0333
learning	105	105/300= 0.35

Now, we will calculate the probability as each target level:
P(entertainment|q) = P(entertainment) * P(machine|entertainment) * P(learning|entertainment)* P(is|entertainment) * P(fun|entertainment) = 0.7 * 0.05* 0.1*0.9929*0.5929 = **0.00206**

P(education|q) = P(education) * P(machine| education) * P(learning| education)* P(is| education) * P(fun| education) = 0.3 * 0.4* 0.35*0.9867*0.6667 = **0.0276**

As the P(education|q) > P(entertainment|q) so the model will predict the **target= Education.**

b. What target level will a naive Bayes model predict for the following query document: "christmas family fun"?
Ans: As the "christmas" word is not available in any of the document of either topic. Due to this both conditional probabilities equal to 0. P(education|q) = 0 and P(entertainment|q) = 0 . Hence, the model will not be able to return the prediction.

c. What target level will a naive Bayes model predict for the query document in part (b) of this question, if Laplace smoothing with k = 10 and a vocabulary size of 6 is used?
Ans: Laplace smoothing for conditional probabilities are defined as

$$P(f = l \mid t) = \frac{count(f = l \mid t) + k}{count(f \mid t) + (k \times |Domain(f)|)}$$

Where count(f = l | t) is how frequent the event f = l occurs in the subset of rows in the dataset and t is the target level. Domain(f) is the number of the levels in the domain of the feature. Larger value of k means more smoothing occurs.
Initial Probabilities for query - **"christmas family fun"** are:

For target level = 'Entertainment':
Initial Probabilites are:
P(Christmas|Entertainment) = 0/700=0
P(family|Entertainment) = 400/700= 0.5714

P(fun|Entertainment)= 415/700 = 0.5929

Smoothing parameters:
K = 6
|Domain(vocabulary)| = 6
Count(Christmas|Entertainment) = 0
Count(family |Entertainment) = 400
Count(fun |Entertainment) = 415

Hence, smoothed conditional probabilities as per the above laplace formula are :
P(Christmas|Entertainment) = 0+ 10/(700 + (10*6)) = 0.0132
P(family |Entertainment) = 400+ 10/(700 + (10*6)) = 0.5395
P(fun |Entertainment) = 415+ 10/(700 + (10*6)) = 0.5592

For target level = 'Education':
Initial Probabilites are:
P(Christmas|Education) = 0/300=0
P(family| Education) = 10/300= 0.0333
P(fun| Education)= 200/300 = 0.6667

Smoothing parameters:
K = 6
|Domain(vocabulary)| = 6
Count(Christmas| Education) = 0
Count(family | Education) = 10
Count(fun | Education) = 200

Hence, smoothed conditional probabilities as per the above laplace formula are :
P(Christmas| Education) = 0+ 10/(300 + (10*6)) = 0.0278
P(family | Education) = 10+ 10/(300 + (10*6)) = 0.0556
P(fun | Education) = 200+ 10/(700 + (10*6)) = 0.5833

Now, we will calculate the probability as each target level:
P(entertainment|q) = P(entertainment) * P(Christmas|Entertainment) * P(family |Entertainment))* P(fun |Entertainment) = 0.7 * 0.0132* 0.5395*0.5592= **0.0028**

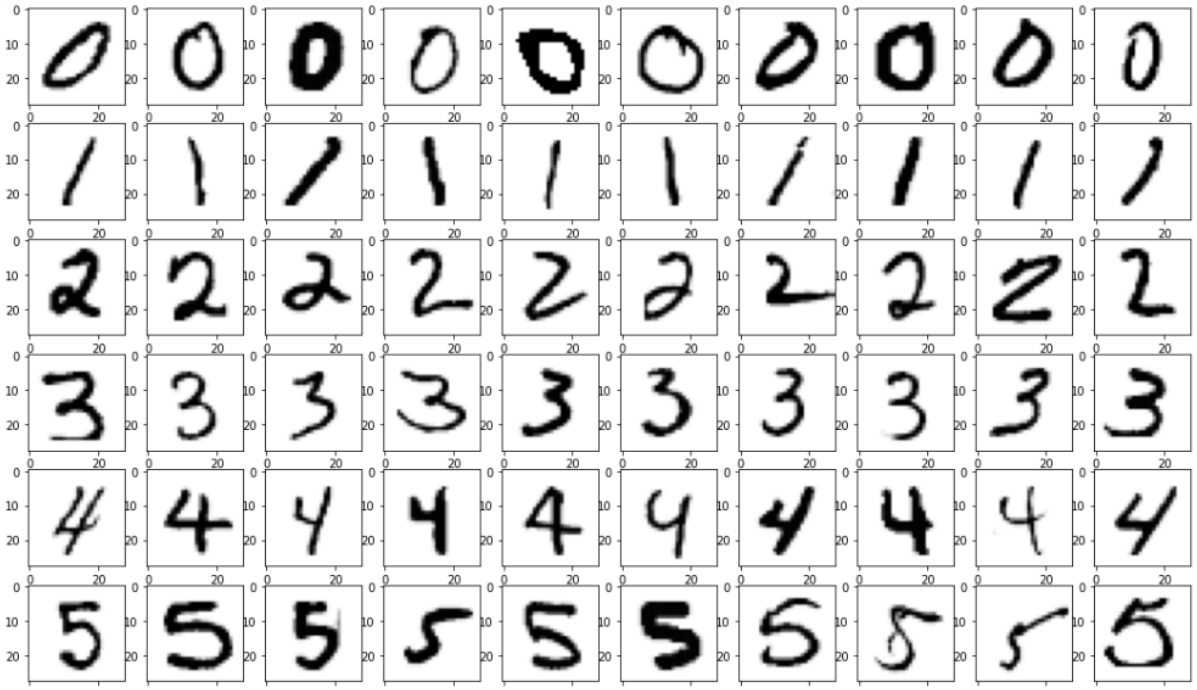
P(education|q) = P(education) * P(Christmas| Education) * P(family | Education))* P(fun | Education) = 0.3*0.0278 * 0.0556* 0.5833= **0.0003**

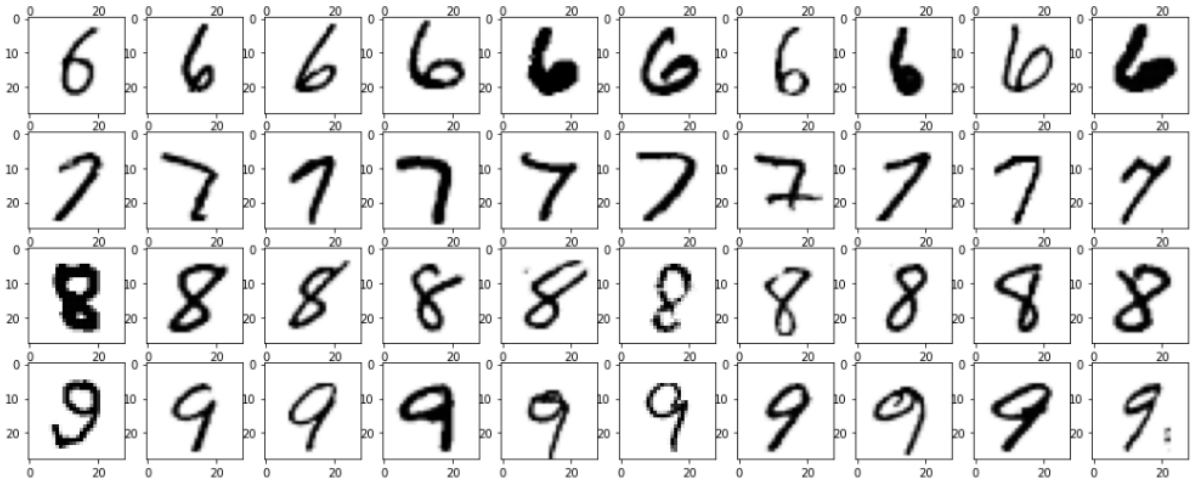
As the P(entertainment |q) > P(education |q) so the model will predict the **target= Entertainment.**

Naive Bayes Classifier for Digit Recognition

A. Create a 10x10 grid to visualize 10 examples of each digit.

```
import pandas as pd
data = pd.DataFrame(mnist.data)
data['target'] = mnist.target

target_10s = {}
target_class = mnist.target.unique().to_list()
target_class.sort()

for target in target_class:
  target_data = pd.DataFrame(data[data.target == target])
  target_data.drop(columns=['target'], inplace=True)
  target_10s[target] = target_data.sample(n = 10).to_numpy()
i = 0
plt.figure(figsize=(20,20))
for target in target_class:
    i = i + 1;
    for j in range(len(target_10s[target])):
        ax = plt.subplot2grid((len(target_class)+1,len(target_10s[target])+1), (i,j))
        ax.imshow(np.reshape(target_10s[target][j], (28, 28)))
plt.show()
```





B. Fit a Naive Bayes classifier and report accuracy on the dev data. Remember that Naive Bayes estimates P(feature|label). While sklearn can handle real-valued features, let's start by mapping the pixel values to either 0 or 1. You can do this as a preprocessing step, or with the binarize argument. With binary-valued features, you can use BernoulliNB. Next try mapping the pixel values to 0, 1, or 2, representing white, grey, or black. This mapping requires MultinomialNB. Does the multi-class version improve the results? Why or why not?

Note, the answers may vary depending on what thresholds you set.

```
from sklearn.metrics import accuracy_score
```

```
from sklearn.naive_bayes import BernoulliNB
```

```
Bernouli_model = BernoulliNB(binarize=0.333)
Bernouli_model.fit(mini_train_data, mini_train_labels)
y_bi_pred = Bernouli_model.predict(dev_data)
print("Accuracy score: ", round(accuracy_score(dev_labels, y_bi_pred) * 100.0,2))
```

```
Accuracy score:  82.0
```

Checking at different Thresholds

```
accuracy=[]
for i in [0.2,0.33,0.4,0.45,0.5]:
    Bernouli_model = BernoulliNB(binarize=i)
    Bernouli_model.fit(mini_train_data, mini_train_labels)
    y_pred = Bernouli_model.predict(dev_data)
    accuracy.append(round(accuracy_score(dev_labels, y_pred) * 100.0,2))

```

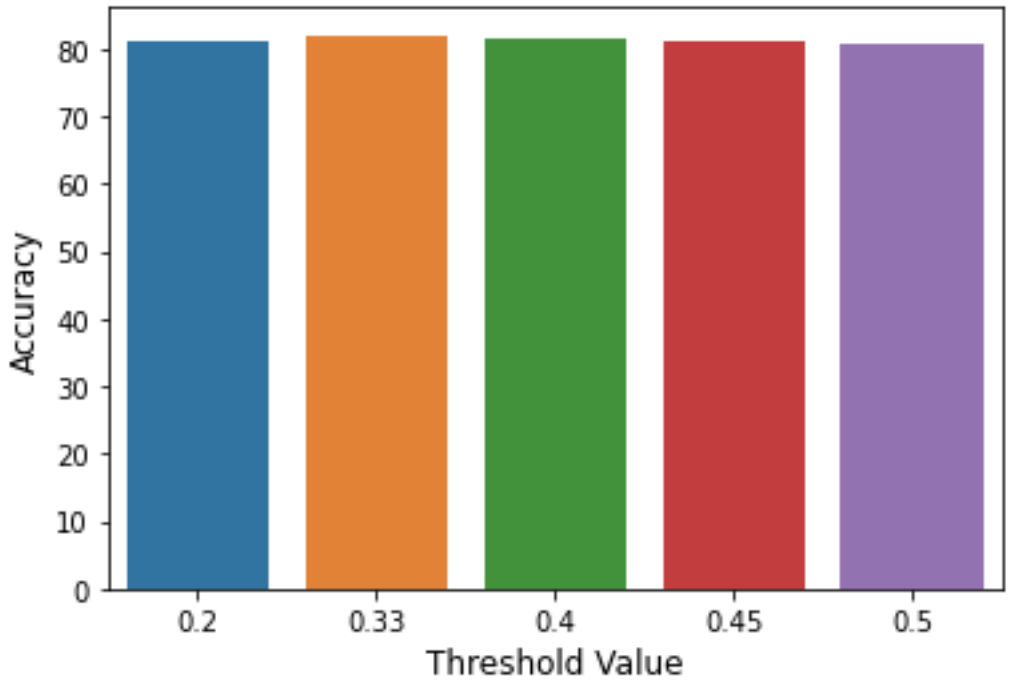
```
i = [0.2,0.33,0.4,0.45,0.5]
```

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
sns.barplot(x=i, y= accuracy)
plt.xlabel('Threshold Value', size= 12)
plt.ylabel('Accuracy', size= 12)

```

```
Text(0, 0.5, 'Accuracy')
```

Bar Plot for Threshold Value wrt Accuracy of the model




```
Bernouli_model = BernoulliNB(binarize=0.33)
Bernouli_model.fit(mini_train_data, mini_train_labels)

```

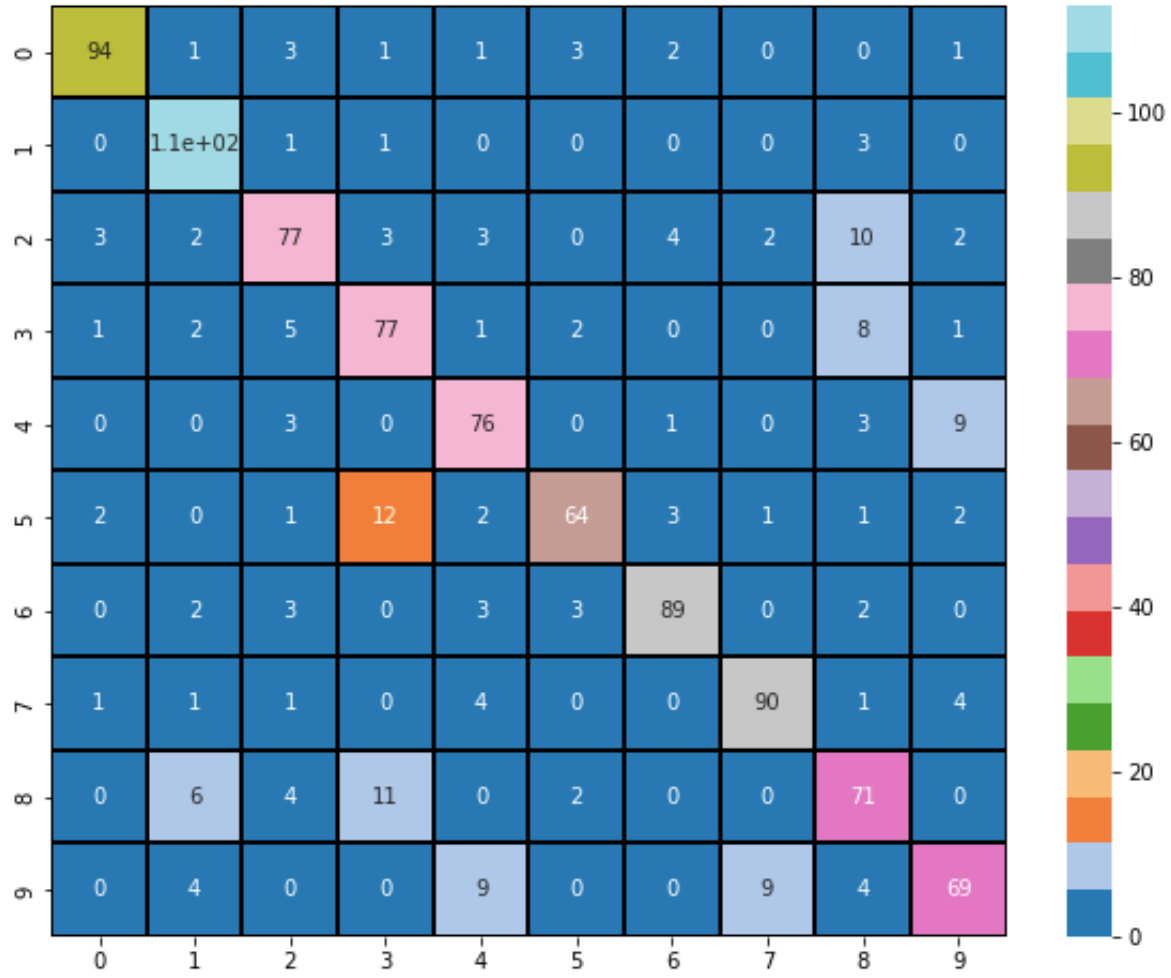
```
BernoulliNB
BernoulliNB(binarize=0.33)
```

```
pred1= Bernouli_model.predict(dev_data)
```

```
from sklearn.metrics import confusion_matrix

```

```
plt.figure(figsize=(10,8))
sns.heatmap(confusion_matrix(dev_labels,pred1), cmap= "tab20", annot=True, linewidths= 2, linecolor= 'black')
plt.show()
```

Confusion Matrix




```
1 from sklearn.metrics import precision_score
2 from sklearn.metrics import recall_score
3 from sklearn.metrics import classification_report
```

```
1 round(precision_score(dev_labels,pred, average='macro')*100,2)
```

```
82.0
```

```
1 round(recall_score(dev_labels,pred, average='macro')*100,2)
```

```
81.55
```

MultinomialNB

```
from sklearn.naive_bayes import MultinomialNB
```

```
X_mapped_train =  pd.DataFrame(mini_train_data)
X_mapped_test =  pd.DataFrame(dev_data)

def scaled(val):
    if val >= 0.66:
      return 2.0
    elif val >= 0.33 and val<0.66:
      return 1.0
    elif val<0.33:
      return 0.0
    else:
      return val

value = np.vectorize(scaled)

for column in X_mapped_train.columns.to_list():
    column_data_train = X_mapped_train[column].to_numpy()
    column_data_test = X_mapped_test[column].to_numpy()
    result_train = value(column_data_train)
    result_test = value(column_data_test)
    X_mapped_train[column].update(pd.Series(result_train))
    X_mapped_test[column].update(pd.Series(result_test))
```

```
multinomialnb_model = MultinomialNB()
```

```
multinomialnb_model.fit(X_mapped_train, mini_train_labels)

```

▾ MultinomialNB

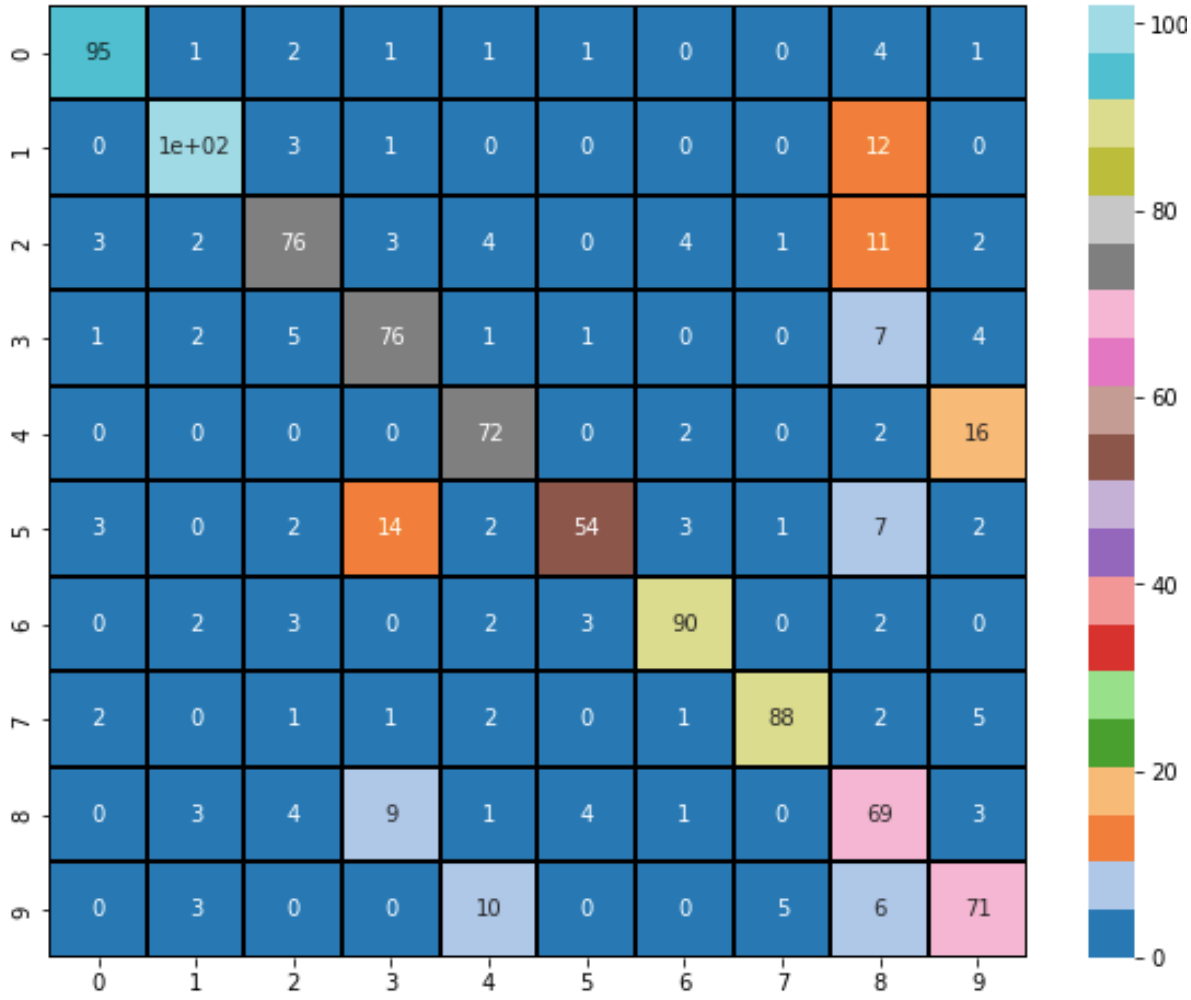
MultinomialNB()

```
pred= multinomialnb_model.predict(X_mapped_test)
```

```
print(round(accuracy_score(dev_labels,pred) * 100.0,2))
```

79.3

Confusion Matrix




```
1 round(precision_score(dev_labels,pred, average='macro')*100,2)
```

```
79.97
```

```
1 round(recall_score(dev_labels,pred, average='macro')*100,2)
```

```
78.84
```

Results

For BernouliNB we checked the threshold at different levels and the accuracy was close to 84% in all cases.we also plotted the confusion matrix and precision and recall score was around 82%. For MultinomialNB we mapped the features in the range 0,1,2 by defining the function and the accuracy was slightly low as compared to BernouliNB ie. around 80%. The accuracy didn't approve with multinomialnb because multinomialnb is more suitable for occurrence counts. However, the bernouli is designed for binary data that's why the accuracy is more on bernoulinb model.

C. Try training a model using GuassianNB, which is intended for real-valued features, and evaluate on the dev data. You'll notice that it doesn't work so well. Could you explain why?

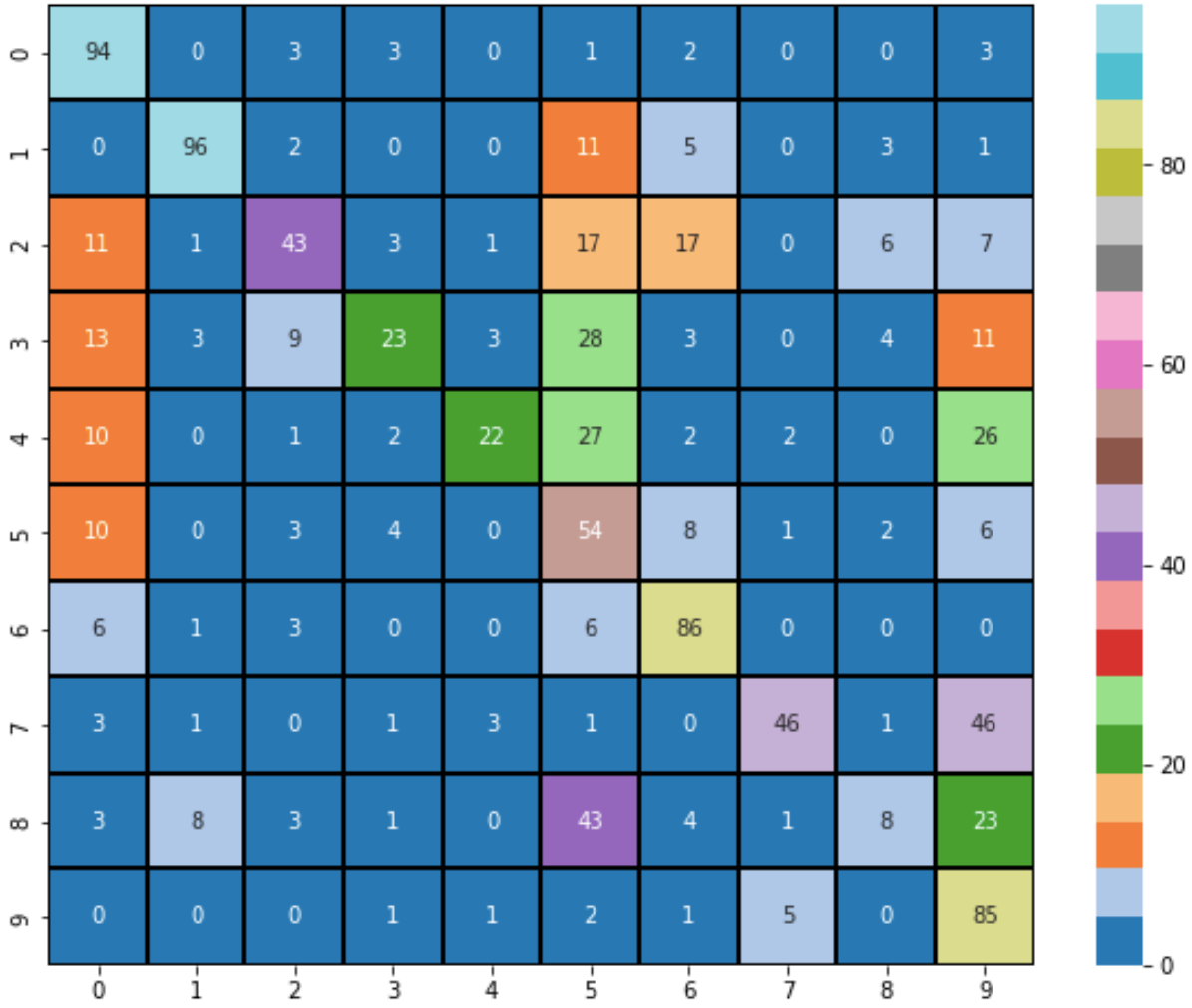
```
gaussiannb_model = GaussianNB()
# Fit the model with the mini training set
gaussiannb_model.fit(mini_train_data, mini_train_labels)
# Report accuracy for the dev set
print('Accuracy of Gaussian NB: {0:.3f}'.format(GaussianNB_model.score(dev_data,dev_labels)))

```

```
Accuracy of Gaussian NB: 0.589
```

```
pred= gaussiannb_model.predict(dev_data)
```

```
plt.figure(figsize=(10,8))
sns.heatmap(confusion_matrix(dev_labels,pred), cmap= "tab20", annot=True, linewidths= 2, linecolor= 'black')
plt.show()
```

Confusion Matrix



```
round(precision_score(dev_labels,pred, average='macro')*100,2)
```

```
60.14
```

```
round(recall_score(dev_labels,pred, average='macro')*100,2)
```

```
54.7
```

Results

The accuracy is low on GaussianNB because the data is not normally distributed. Gaussian Model expects features to be normally distributed but the features are almost binary. Due to this, the accuracy is low ie. around 60%. Also there are many false positives and false negatives as the precision and recall is low. This shows the model is not working properly as compared to BernouliNB and MultinomialNB.
To prove this, we also draw the histrogram showing distribution of features and we can clearly see that the distribution is not normal. However, after adding noise using np.random.normal the features are now normally distributed. To prove our point, we again fit the model on this data with noise and the accuracy of the model is now 80%. Also the precision and recall has improved.

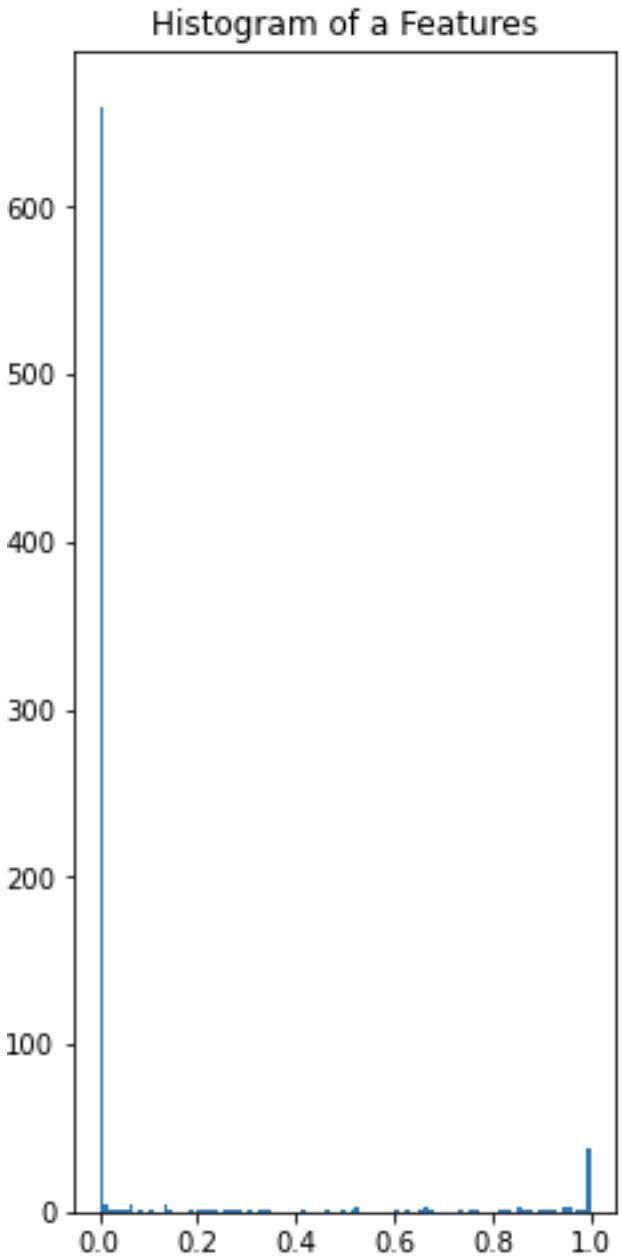
```
import numpy as np
```

```
X.shape[0]
```

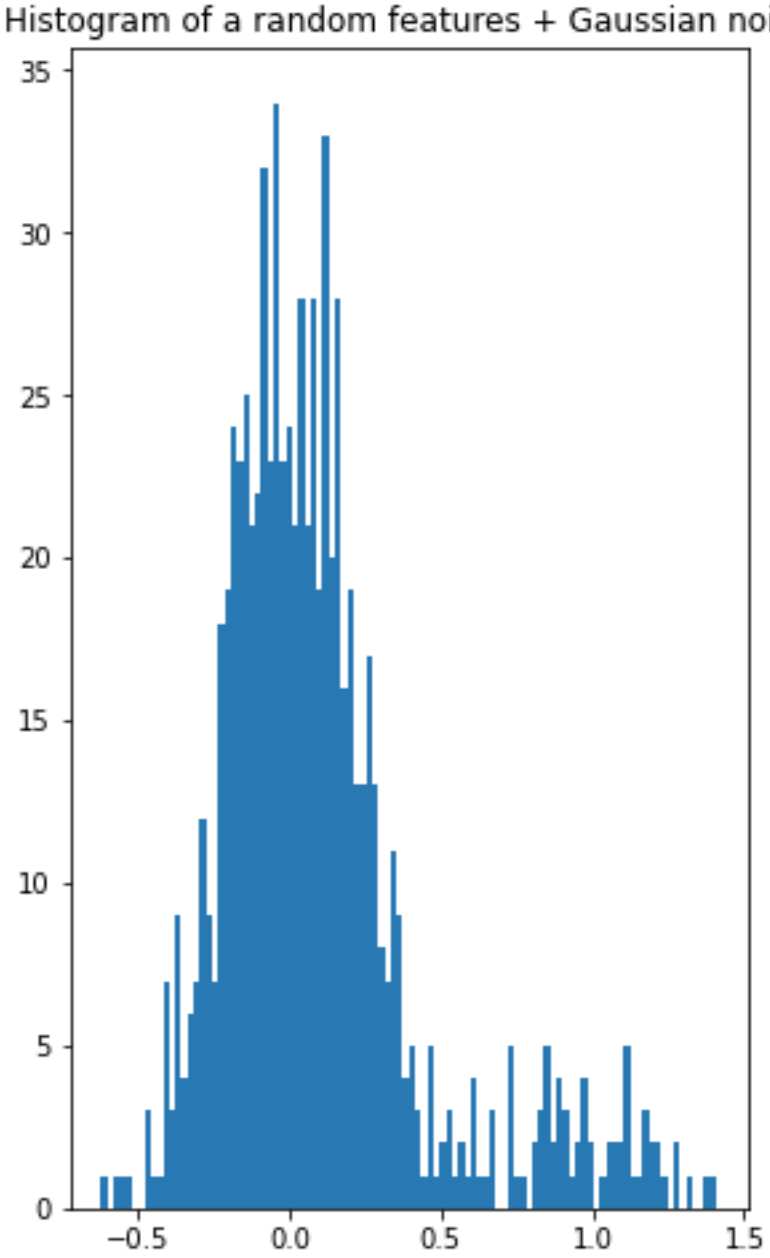
```
70000
```

```
val_random = X.iloc[np.random.choice(X.shape[0])]

```

```
plt.figure(figsize=(8,8))
plt.subplot(1, 2, 1)
fig = plt.hist(val_random, 100)
plt.title('Histogram of a Features')
```




```
plt.figure(figsize=(10,8))
plt.subplot(1, 2, 2)
fig = plt.hist(val_random + np.random.normal(0, 0.2, random_image.shape), 100)
plt.title('Histogram of a random features + Gaussian noise')
```

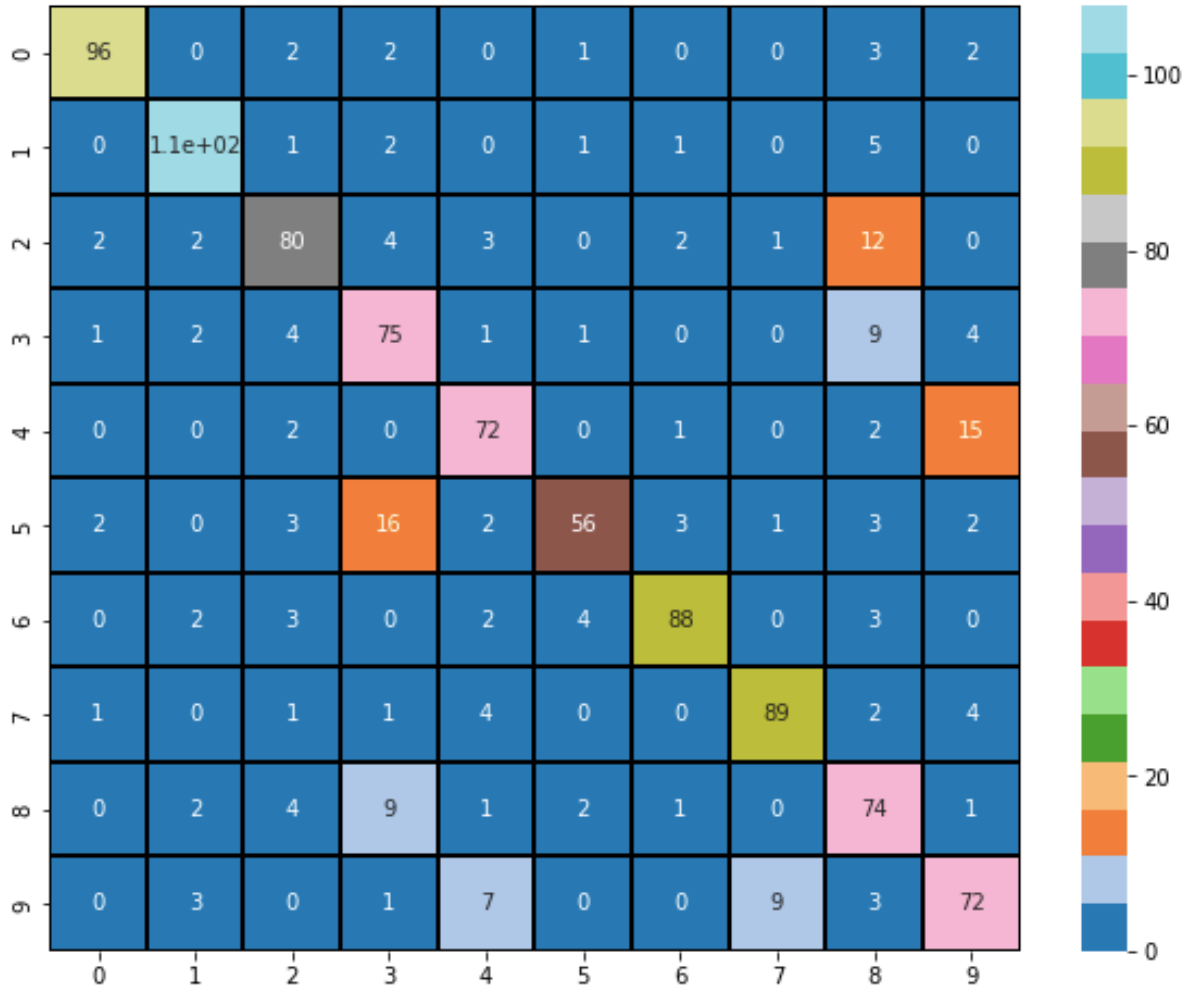


Accuracy of Model after adding noise

```
new_train_data = mini_train_data + np.random.normal(0,0.2,mini_train_data.shape)
new_dev_data = dev_data + np.random.normal(0,0.2,mini_train_data.shape)
gaussiannb_model.fit(new_train_data, mini_train_data_labels)
print('Accuracy of new GaussianNB after adding noise = {0}'.format(gaussiannb_model.score(new_dev_data, dev_labels)
```

```
Accuracy of new GaussianNB after adding noise = 81.0
```

Confusion Matrix




```
1 round(precision_score(dev_labels,pred, average='macro')*100,2)
```

```
81.39
```

```
1 round(recall_score(dev_labels,pred, average='macro')*100,2)
```

```
80.48
```