Lab 3

Nakkina Vinay(B21AI023)

Question 1

Subpart 1: Data Preprocessing, Visualization and Finding important features in the dataset

- Data was loaded and processed. We can see the data using data.head() and data.info()
- Checking the empty values in the features of the dataset
- Using the countplot and histogram plots I am visualizing the different columns and plotting the graphs
- Filling the missing values in AGE column by taking a function and filling the column based on the average age of the persons from different Pclass
- Due to the number of missing values in ‘Cabin’ feature, column was dropped.
- Unnecessary features like ‘Name’, ‘Ticket’ and ‘PassengerId’ were also dropped immediately.
- Converting the string values of columns (Sex, Pclass, Embarked) into dummy values of 0 and 1
- After converting the values to dummies, we should add these newly formed columns to the original dataset
- After adding the columns to the original dataset, we can now drop the columns of Sex, Pclass, Embarked from the dataset because the dummy value columns are added
- Dependency of Survivability with respect to various features can be summarized
- Clearly, the most important features are:

    Sex (converted to male)

    Pclass (converted to 2 and 3)

    Age

    Embarked (converted to Q and S)

    Fare

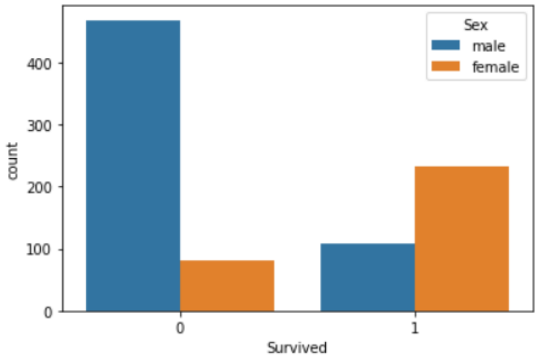
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Name         891 non-null    object
 2   Pclass       891 non-null    int64
 3   Sex          891 non-null    object
 4   Age          714 non-null    float64
 5   Ticket       891 non-null    object
 6   Fare         891 non-null    float64
 7   Cabin        204 non-null    object
 8   Embarked     889 non-null    object
 9   Survived     891 non-null    int64
dtypes: float64(2), int64(3), object(5)
memory usage: 69.7+ KB
```

Converted To →

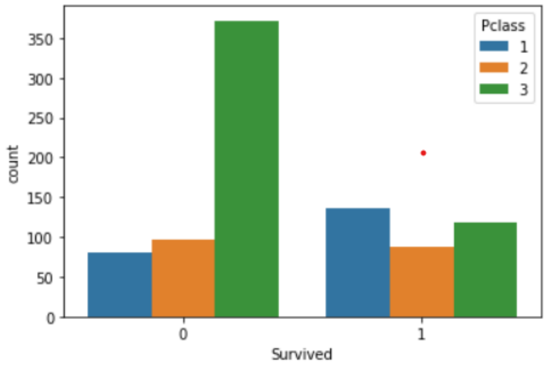
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Age       891 non-null    float64
 1   Fare      891 non-null    float64
 2   Survived  891 non-null    int64
 3   male      891 non-null    uint8
 4   Q         891 non-null    uint8
 5   S         891 non-null    uint8
 6   2         891 non-null    uint8
 7   3         891 non-null    uint8
dtypes: float64(2), int64(1), uint8(5)
memory usage: 25.4 KB
```

Visualisation of Dataset

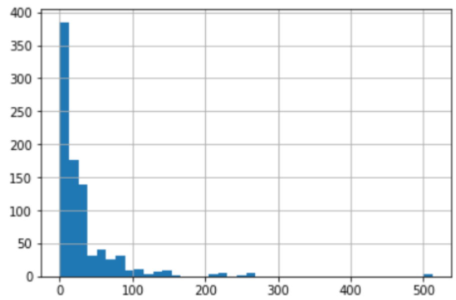
<matplotlib.axes._subplots.AxesSubplot at 0x7fca332c3e20>



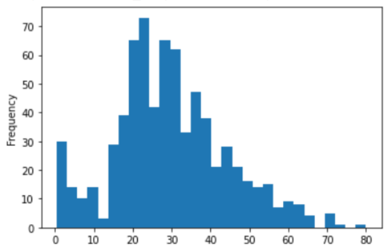
<matplotlib.axes._subplots.AxesSubplot at 0x7fca30098340>



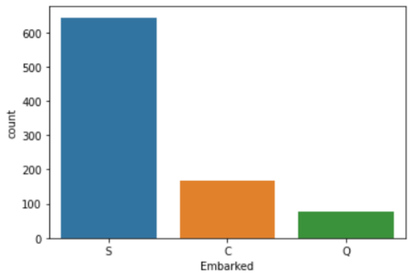
<matplotlib.axes._subplots.AxesSubplot at 0x7fca335e



<matplotlib.axes._subplots.AxesSubplot at 0x7fca332e



<matplotlib.axes._subplots.AxesSubplot at 0x7fca3010d



Data Preprocessing is completed and dropping of the un-used columns is also completed and we can now split our dataset into train and test sets

- Assuming values for X and Y, we can split the dataset into train and test sets using train_test_split
- This is our dataset after the subpart one:

	Age	Fare	Survived	male	Q	S	2	3
0	22.0	7.2500	0	1	0	1	0	1
1	38.0	71.2833	1	0	0	0	0	0
2	26.0	7.9250	1	0	0	1	0	1
3	35.0	53.1000	1	0	0	1	0	0
4	35.0	8.0500	0	1	0	1	0	1

Subpart 2: Identifying best variant of NB

There are three variants of Naive Bayes we can choose for classification:-

1) Gaussian Naive Bayes - continuous features
2) Bernoulli Naive Bayes -binary features
3) Multinomial Naive Bayes - categorical features

After calculating the accuracy using all the three variants of Naive Bayes Classifier we are getting the highest accuracy for Gaussian Naïve Bayes

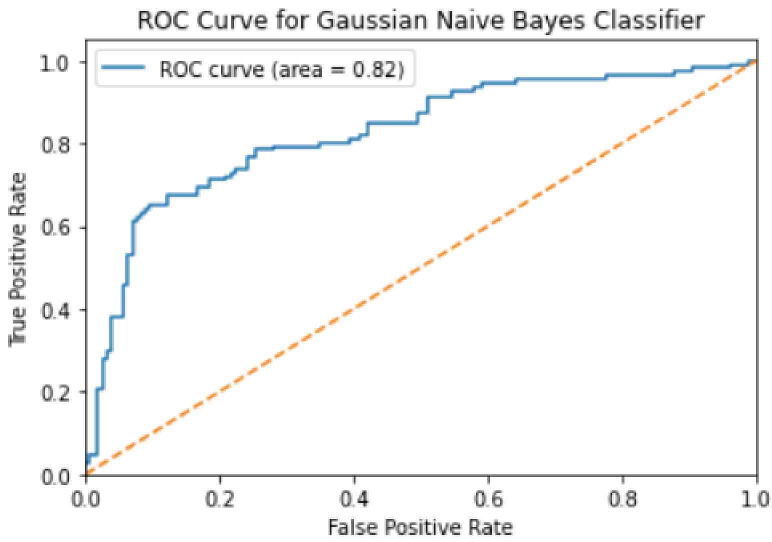
```
Gaussian Naive Bayes Classifier Accuracy: 0.7835820895522388

Multinomial Naive Bayes Classifier Accuracy: 0.7014925373134329

Bernoulli Naive Bayes Classifier Accuracy: 0.7798507462686567
```

Subpart 3: Implementing the above Gaussian variant

- Our Model had following methods:
  - fit(X, y)
  - predict(X):
  - accuracy_score(y_test, y_prediction)

After calculating the accuracy score we are now printing the ROC curve using the roc_curve function and finding the area under the ROC curve

```
Area under ROC Curve is:  0.8237791932059448
```



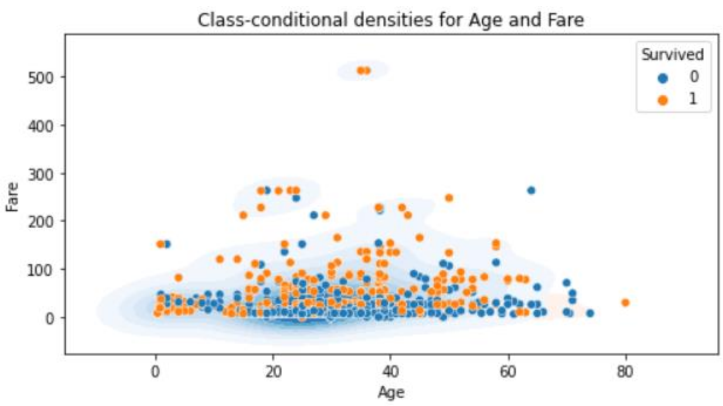
Subpart 4: Performing 5 fold cross validation

- Performing % fold cross validation using Kfold
- Making five folds from the dataset and calculating the accuracy from the Gaussian Naïve Bayes and storing it in an array

```
Mean Validation NB Score:  0.7832387096774195
Variance in NB validation Score:  0.004523861644120709
```

```
Accuracy results for 5 fold Cross-validation
 [0.770949720670391, 0.7584269662921348, 0.8202247191011236, 0.8089887640449438, 0.7808988764044944]

Average accuracy: 0.7878978093026175
```

Subpart 5: Contour plot with data points to visualize the class conditionals



Using the kdeplot function from seabour plotting the class conditional densities for age and fare confidences of Naïve Bayes classifier are high and accuracy is higher than compared to other variants of Naïve Bayes Classifier

Subpart 6: Comparison with another Model (Decision Tree)

- Decision Tree classifier gives higher accuracy than the Naïve Bayes Classifier
- So Decision tree classifier works better on this dataset

```
Mean Validation DT Score:  0.7753161290322581
Variance in DT validation Score:  0.00047162139438085336
Accuracy results for 5 fold Cross-validation
 [0.8379888268156425, 0.8258426966292135, 0.7865168539325843, 0.7808988764044944, 0.7359550561797753]

Average accuracy of Decision Tree: 0.793440461992342
```

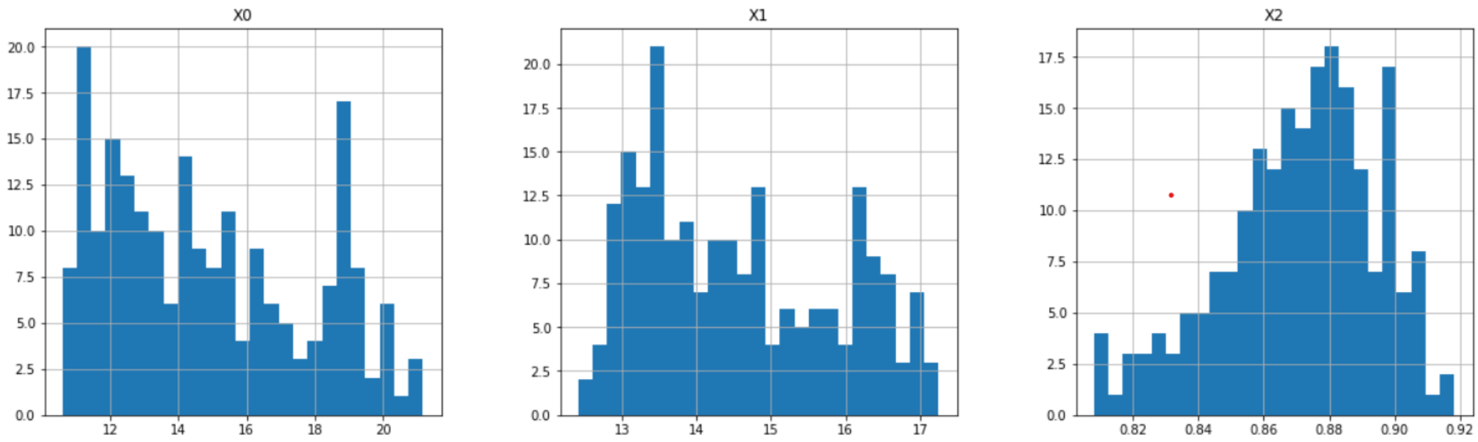
Question 2

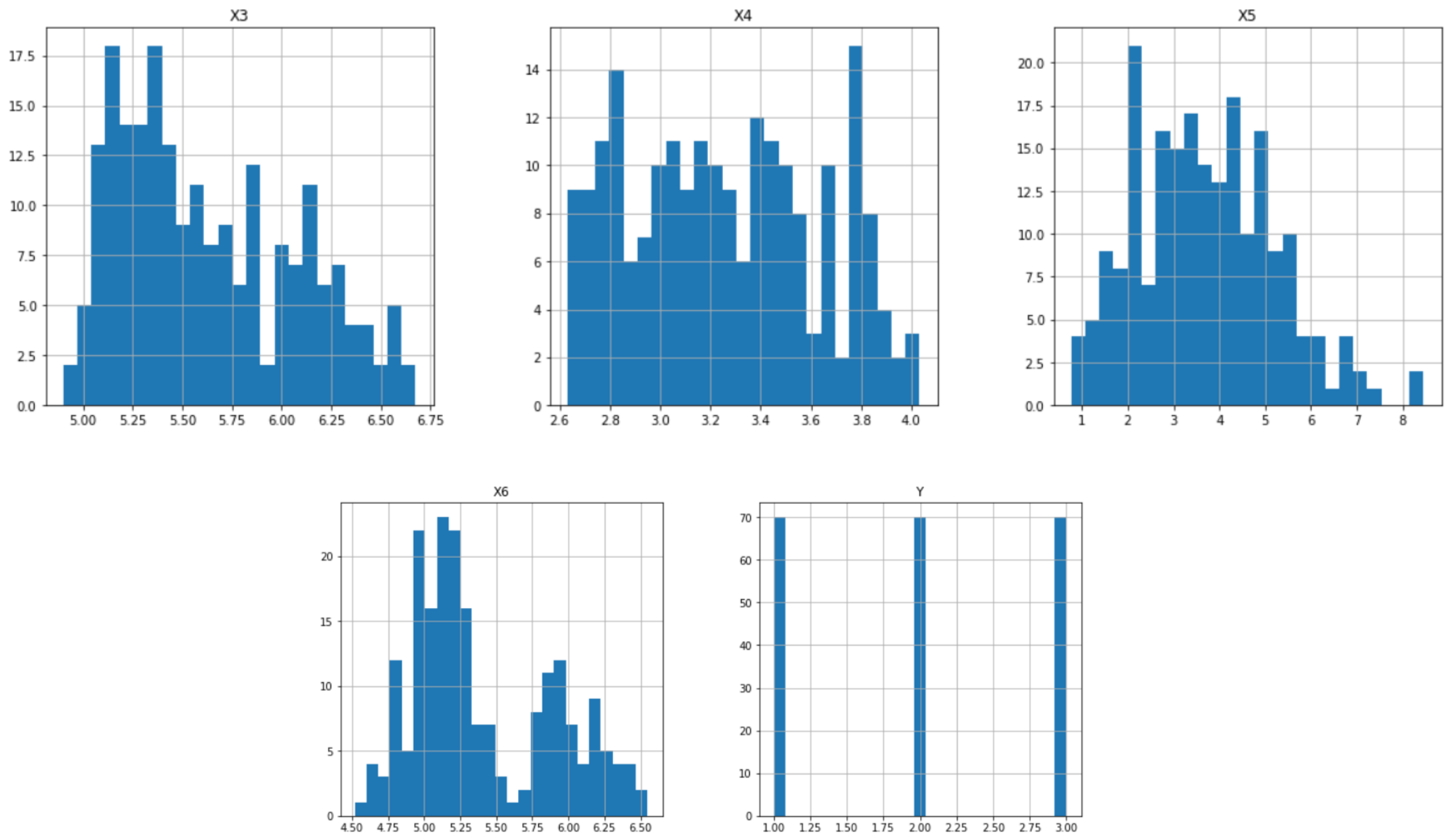
Subpart 1: Plotting histograms

- Necessary preprocessing is performed and graphs are plotted

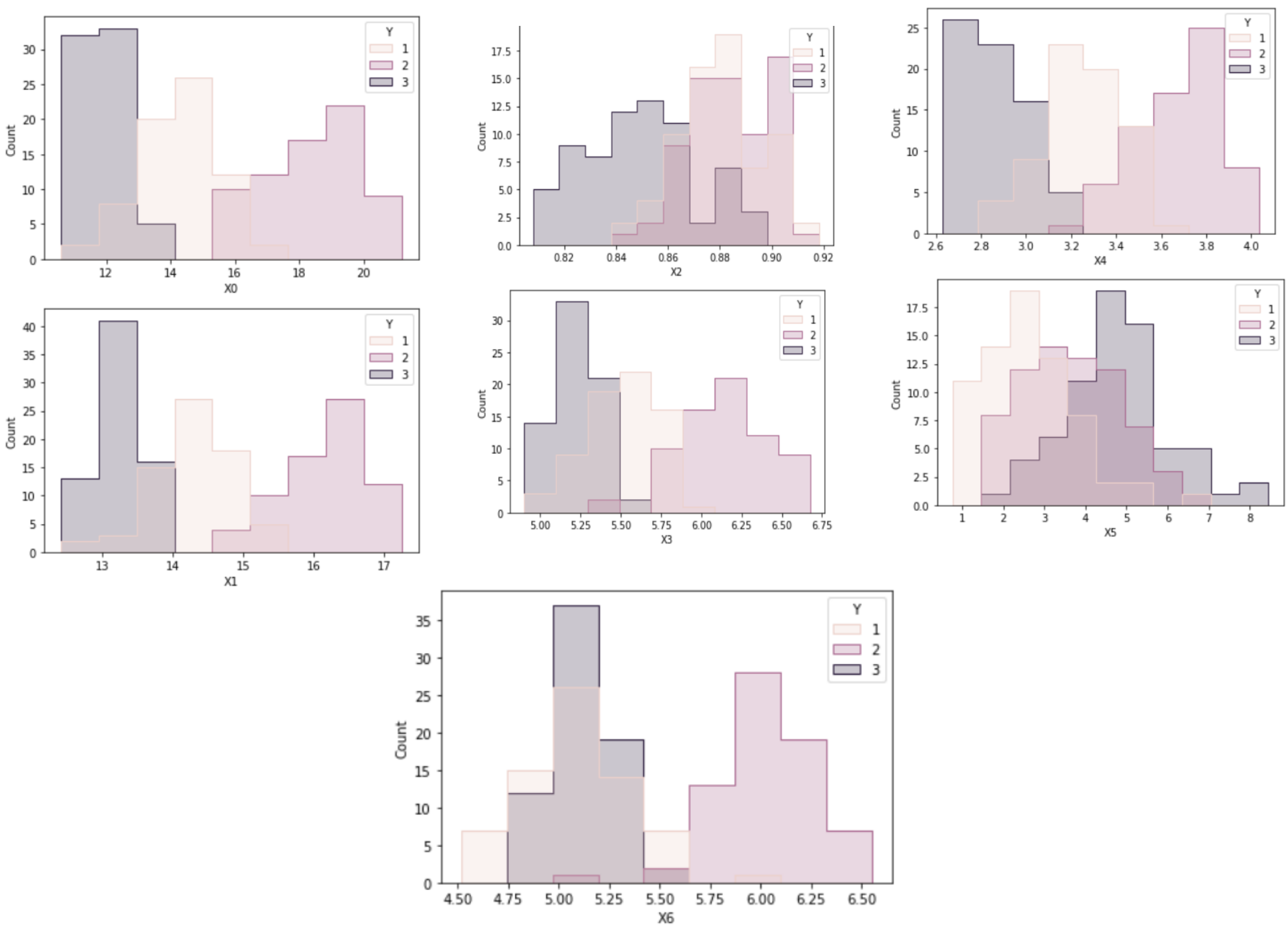
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210 entries, 0 to 209
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   X0      210 non-null    float64
 1   X1      210 non-null    float64
 2   X2      210 non-null    float64
 3   X3      210 non-null    float64
 4   X4      210 non-null    float64
 5   X5      210 non-null    float64
 6   X6      210 non-null    float64
 7   Y       210 non-null    int64
dtypes: float64(7), int64(1)
memory usage: 13.2 KB
```

	X0	X1	X2	X3	X4	X5	X6	Y
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	1
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	1
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	1
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	1
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	1





- Further histplots for individual classes was also plotted and we found the maximum features too be retained at 5 and 7 bins.
- 3 is unable to capture the variations in density across the data while 9+ bins are capturing way too minute features.
- On this analysis, 5 bins are chosen.



b: calculating priors

Calculation of priors was straight forward

```
Prior probability of class 1:  0.3333333333333333
Prior probability of class 2:  0.3333333333333333
Prior probability of class 3:  0.3333333333333333
```

Subtask c: Discretizing into bins (and Subtask e)

- Number of bins chosen is 5.
- Here we bin the continous features on the principle of **equal width binning**.
- Visualization of our implementation:

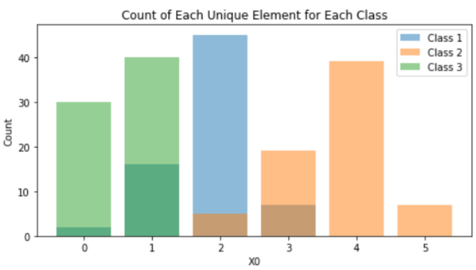
	X0	X1	X2	X3	X4	X5	X6	Y
0	2	3	3	2	2	1	2	1
1	2	2	3	2	3	0	1	1
2	2	2	4	1	3	1	1	1
3	2	2	4	1	3	1	1	1
4	3	3	4	2	3	0	2	1
..	..	..	..	..	..	..	..	..
205	1	1	3	1	1	2	1	3
206	0	0	2	1	1	2	1	3
207	1	1	4	1	2	5	1	3
208	1	1	2	1	1	2	1	3
209	1	1	3	1	1	3	1	3

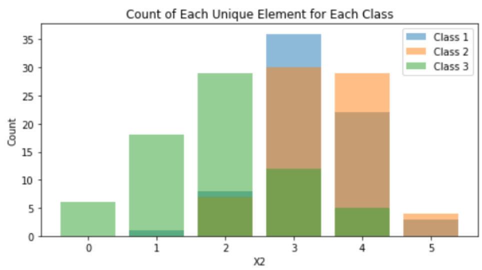
[210 rows x 8 columns]

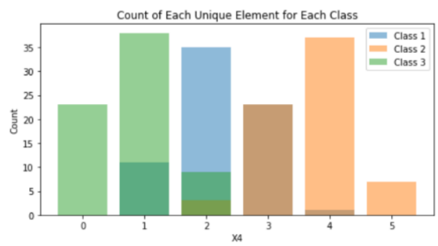
Subtask d: calculating likelihood and plotting

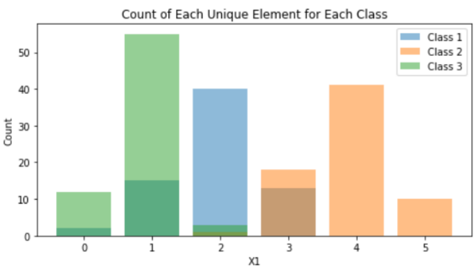
Calculation od likelihood was straight-forward

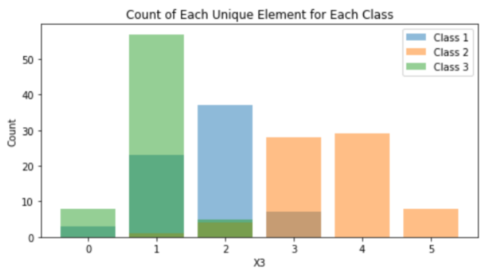
Subtask e: Plot the count of unique element for each class

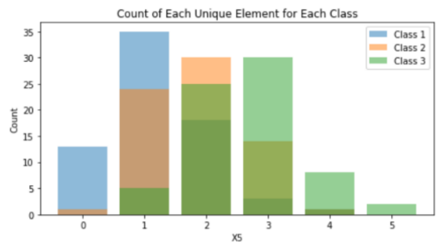


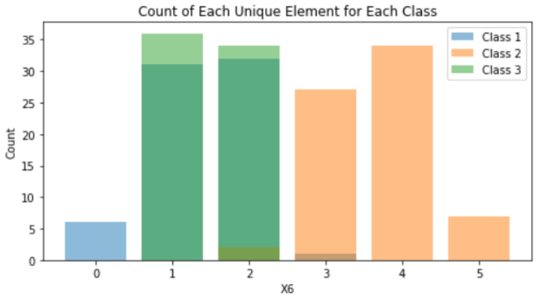












Subtask f: Calculating and plotting posteriors

Posteriors was calculated feature-wise.

$$posterior = \frac{likelihood \times prior}{evidence}$$

The calculation was straight-forward and formula based.