Project - Logistic Regression

Abstract:

The dataset consists of 480 student records and 16 features. The features are classified into three major categories: (1) Demographic features such as gender and nationality. (2) Academic background features such as educational stage, grade Level and section. (3) Behavioral features such as raised hand on class, opening resources, answering survey by parents, and school satisfaction.The students are classified into three numerical intervals based on their total grade/mark

Problem Statement:

Using the dataset we are going to which students are in which class using Logistic Regression.

Logistic Regression:

It is used to analyze relationship between categorical dependent variable and categorical or numerical independent variable. It combine the independent varable to estimates the probability that a particular event will occur.

Libraries

In [31]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Load the dataset

In [32]:

```
mydata = pd.read_csv("xAPI-Edu-Data.csv")
mydata.head(10)
```

Out[32]:

	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation	raisedhands	VisITedResources
0	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	15	16
1	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	20	20
2	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	10	7
3	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	30	25
4	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	40	50
5	F	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	42	30
6	M	KW	KuwaIT	MiddleSchool	G-07	A	Math	F	Father	35	12
7	M	KW	KuwaIT	MiddleSchool	G-07	A	Math	F	Father	50	10
8	F	KW	KuwaIT	MiddleSchool	G-07	A	Math	F	Father	12	21
9	F	KW	KuwaIT	MiddleSchool	G-07	B	IT	F	Father	70	80

To display the datatype

In [33]:
```
mydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 17 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   gender                    480 non-null    object
 1   NationalITy               480 non-null    object
 2   PlaceofBirth              480 non-null    object
 3   StageID                   480 non-null    object
 4   GradeID                   480 non-null    object
 5   SectionID                 480 non-null    object
 6   Topic                     480 non-null    object
 7   Semester                  480 non-null    object
 8   Relation                  480 non-null    object
 9   raisedhands               480 non-null    int64 
 10  VisITedResources          480 non-null    int64 
 11  AnnouncementsView         480 non-null    int64 
 12  Discussion                480 non-null    int64 
 13  ParentAnsweringSurvey     480 non-null    object
 14  ParentschoolSatisfaction  480 non-null    object
 15  StudentAbsenceDays        480 non-null    object
 16  Class                     480 non-null    object
dtypes: int64(4), object(13)
memory usage: 63.9+ KB
```

Check the null values

In [34]:
```
mydata.isnull().sum()
```

Out[34]:
```
gender                      0
NationalITy                 0
PlaceofBirth                0
StageID                     0
GradeID                     0
SectionID                   0
Topic                       0
Semester                    0
Relation                    0
raisedhands                 0
VisITedResources            0
AnnouncementsView           0
Discussion                  0
ParentAnsweringSurvey       0
ParentschoolSatisfaction    0
StudentAbsenceDays          0
Class                       0
dtype: int64
```

Label Encoder (convert object datatype into int)

In [35]:
```
from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()
```

In [36]:
```
mydata["gender"]=LE.fit_transform(mydata.gender)
mydata["NationalITy"]=LE.fit_transform(mydata.NationalITy)
mydata["PlaceofBirth"]=LE.fit_transform(mydata.PlaceofBirth)
mydata["StageID"]=LE.fit_transform(mydata.StageID)
mydata["GradeID"]=LE.fit_transform(mydata.GradeID)
mydata["SectionID"]=LE.fit_transform(mydata.SectionID)
mydata["Topic"]=LE.fit_transform(mydata.Topic)
mydata["Semester"]=LE.fit_transform(mydata.Semester)
mydata["Relation"]=LE.fit_transform(mydata.Relation)
mydata["ParentAnsweringSurvey"]=LE.fit_transform(mydata.ParentAnsweringSurvey)
mydata["ParentschoolSatisfaction"]=LE.fit_transform(mydata.ParentschoolSatisfaction)
mydata["StudentAbsenceDays"]=LE.fit_transform(mydata.StudentAbsenceDays)
mydata["Class"]=LE.fit_transform(mydata.Class)
```

In [37]:
```
mydata.head(10)
```

Out[37]:

	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation	raisedhands	VisITedResources	Anno
0	1	4	4	2	1	0	7	0	0	15	16	
1	1	4	4	2	1	0	7	0	0	20	20	
2	1	4	4	2	1	0	7	0	0	10	7	
3	1	4	4	2	1	0	7	0	0	30	25	
4	1	4	4	2	1	0	7	0	0	40	50	
5	0	4	4	2	1	0	7	0	0	42	30	
6	1	4	4	1	4	0	8	0	0	35	12	
7	1	4	4	1	4	0	8	0	0	50	10	
8	0	4	4	1	4	0	8	0	0	12	21	
9	0	4	4	1	4	1	7	0	0	70	80	

```
##### Correlation
To find the relationship between the variables.
```

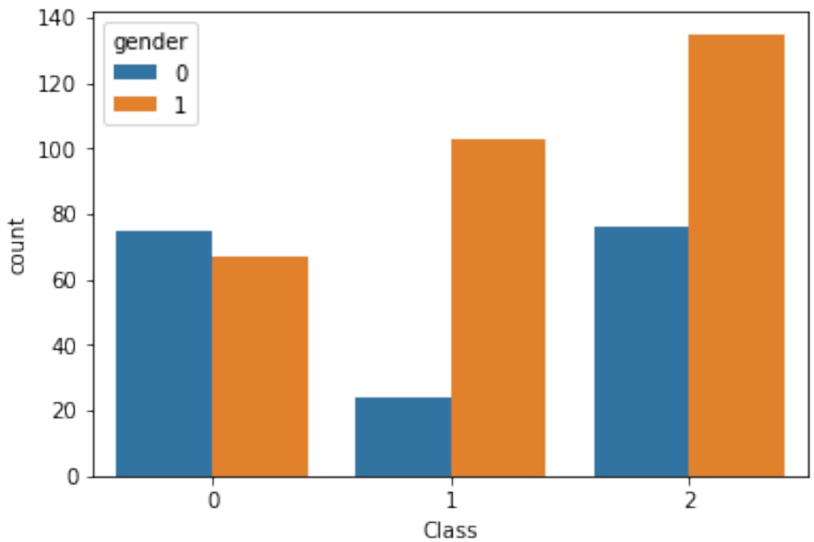
Visualize:

Visualize just the categorical features individually to see what options are included and how each option fares when it comes to count(how many times it appears) and see what can be deduce from t

Graphs:

In [39]:
```
sns.countplot(x="Class",data=mydata,hue="gender")
```

Out[39]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd8dd8b6190>
```

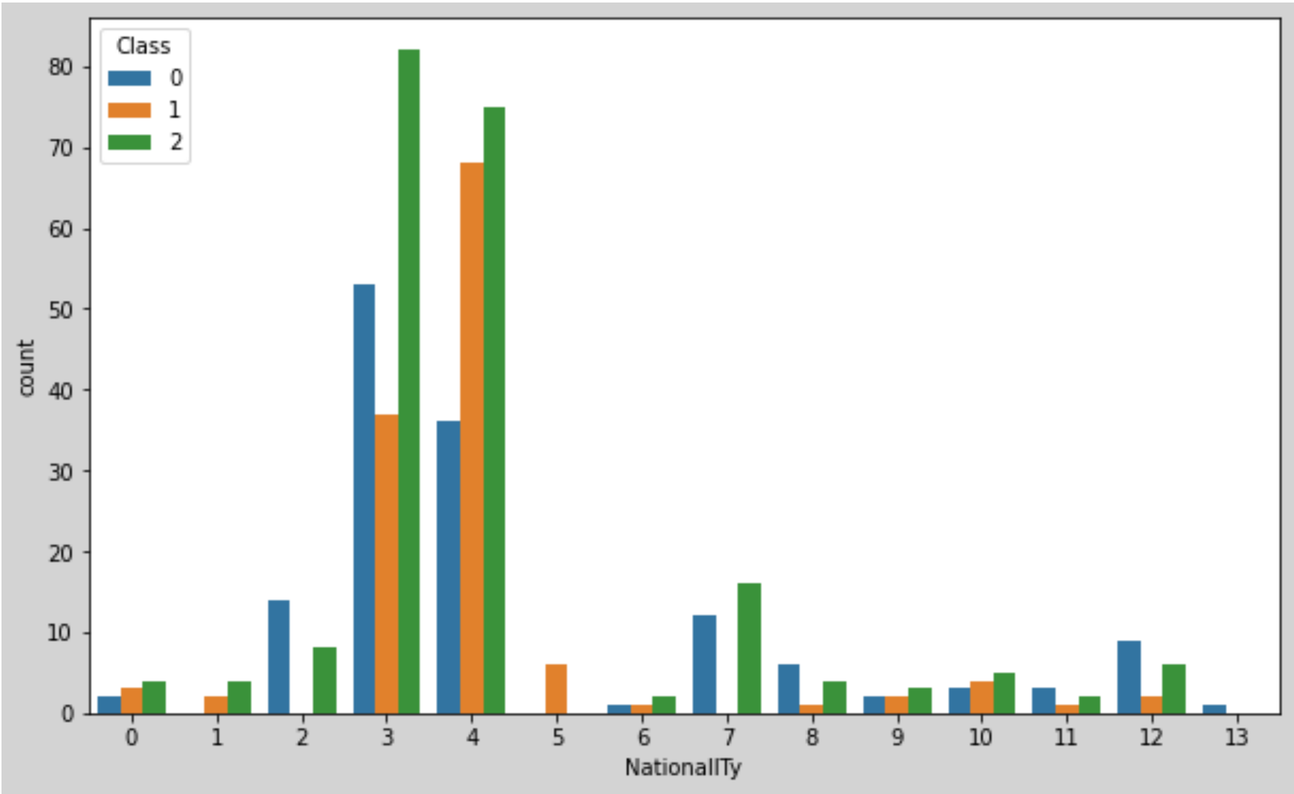


In [40]:
```
mydata.gender.value_counts()
```

Out[40]:
```
1    305
0    175
Name: gender, dtype: int64
```

In [41]:
```
plt.figure(figsize=(10,6),facecolor='lightgrey')
sns.countplot(x='NationalITy',hue='Class',data=mydata)
```

Out[41]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd8dda249a0>
```

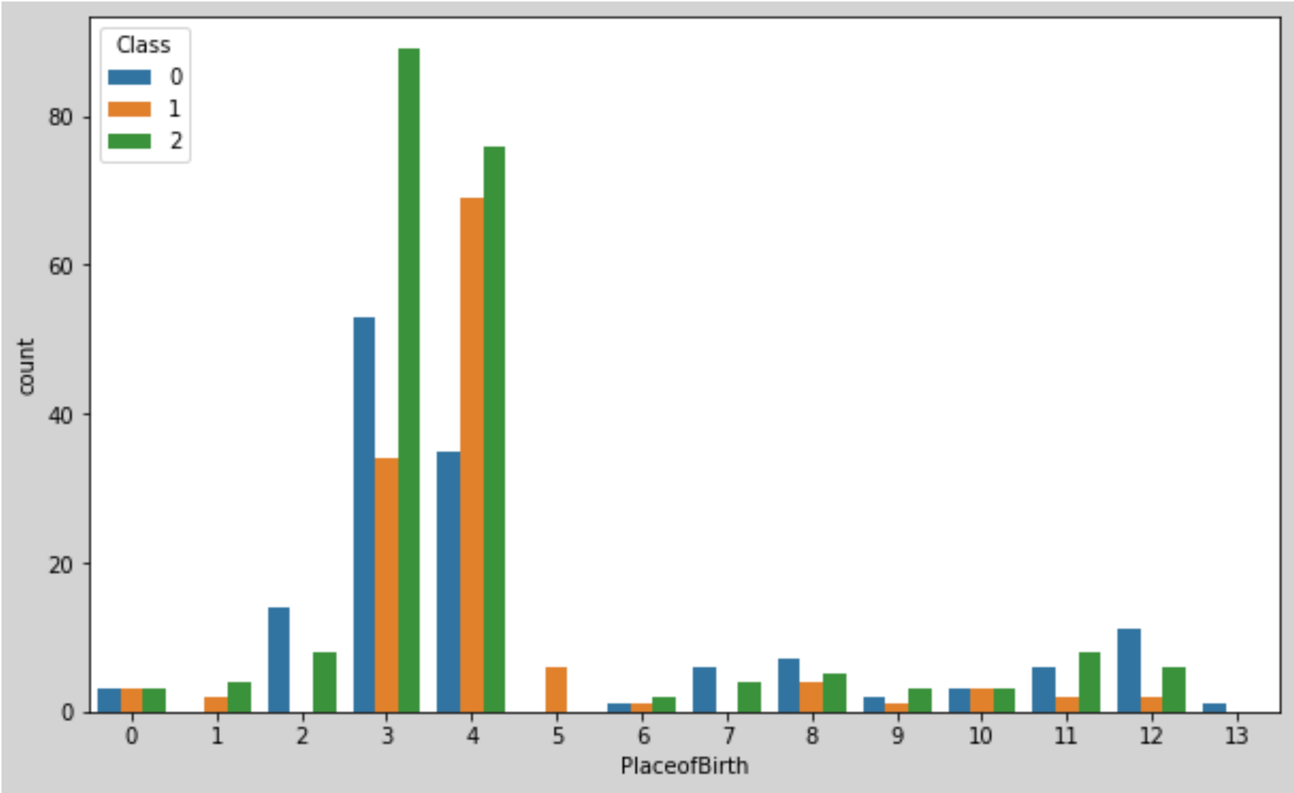


In [42]:
```
mydata.NationalITy.value_counts()
```

Out[42]:
```
4     179
3     172
7      28
2      22
12     17
10     12
8      11
0       9
9       7
11      6
5       6
1       6
6       4
13      1
Name: NationalITy, dtype: int64
```

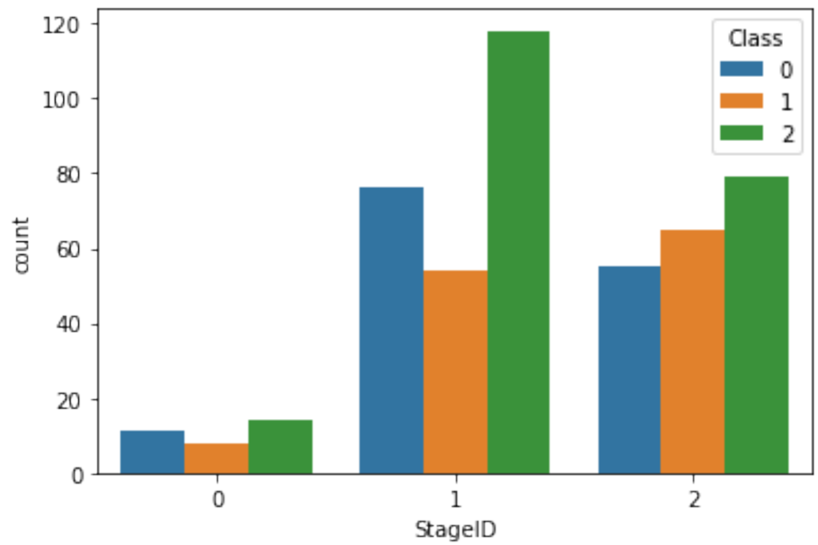
In [43]:
```
plt.figure(figsize=(10,6),facecolor='lightgrey')
sns.countplot(x='PlaceofBirth',hue='Class',data=mydata)
```

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd8dd8c8250>



In [44]:
```
sns.countplot(x='StageID',hue='Class',data=mydata)
```

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd8dde62a00>

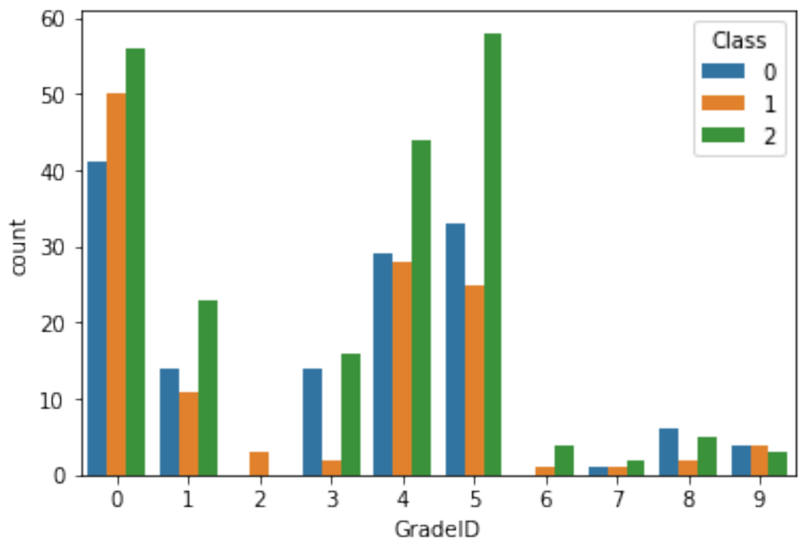


In [45]:
```
mydata.StageID.value_counts()
```

Out[45]:
```
1    248
2    199
0     33
Name: StageID, dtype: int64
```

In [46]:
```
sns.countplot(x='GradeID',hue='Class',data=mydata)
```

Out[46]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd8ddfa4f10>
```

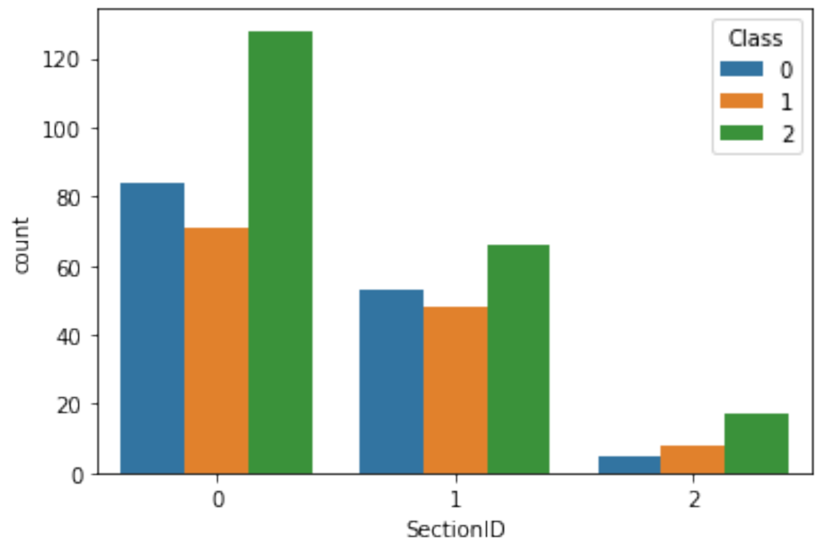


In [47]:
```
mydata.GradeID.value_counts()
```

Out[47]:
```
0    147
5    116
4    101
1     48
3     32
8     13
9     11
6      5
7      4
2      3
Name: GradeID, dtype: int64
```

In [48]:
```
sns.countplot(x='SectionID',hue='Class',data=mydata)
```

Out[48]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd8de0c5fd0>
```

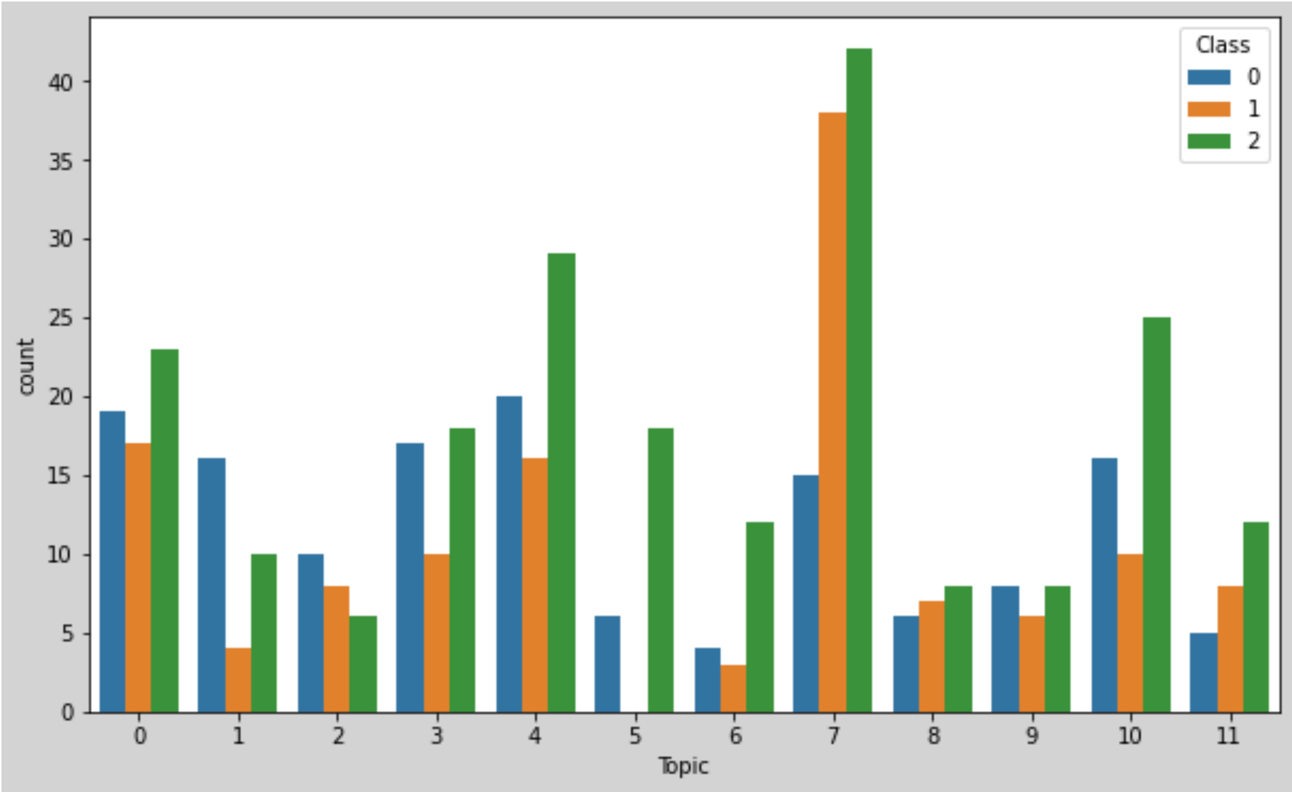


In [49]:
```
mydata.SectionID.value_counts()
```

Out[49]:
```
0    283
1    167
2     30
Name: SectionID, dtype: int64
```

In [50]:
```
plt.figure(figsize=(10,6),facecolor='lightgrey')
sns.countplot(x='Topic',hue='Class',data=mydata)
```

Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd8de088370>

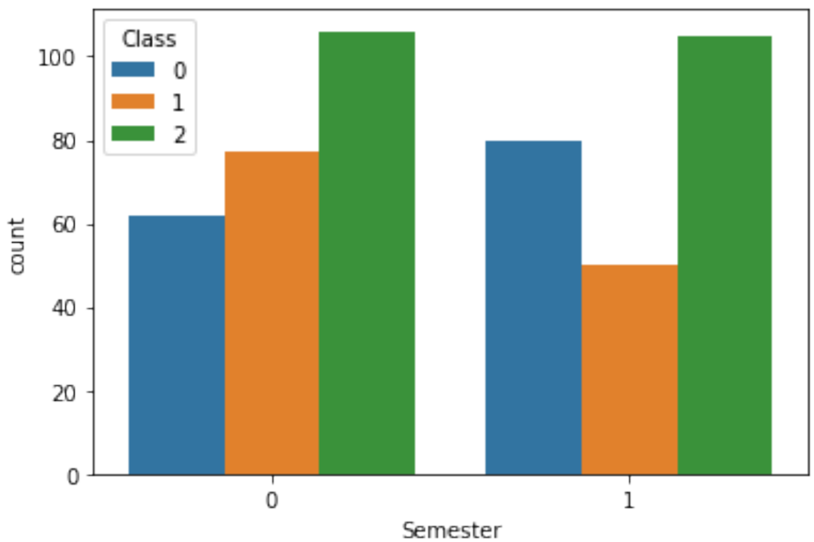


In [51]:
```
mydata.Topic.value_counts()
```

Out[51]:
```
7     95
4     65
0     59
10    51
3     45
1     30
11    25
5     24
2     24
9     22
8     21
6     19
Name: Topic, dtype: int64
```

In [52]:
```
sns.countplot(x='Semester',hue='Class',data=mydata)
```

Out[52]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd8de1ac940>
```

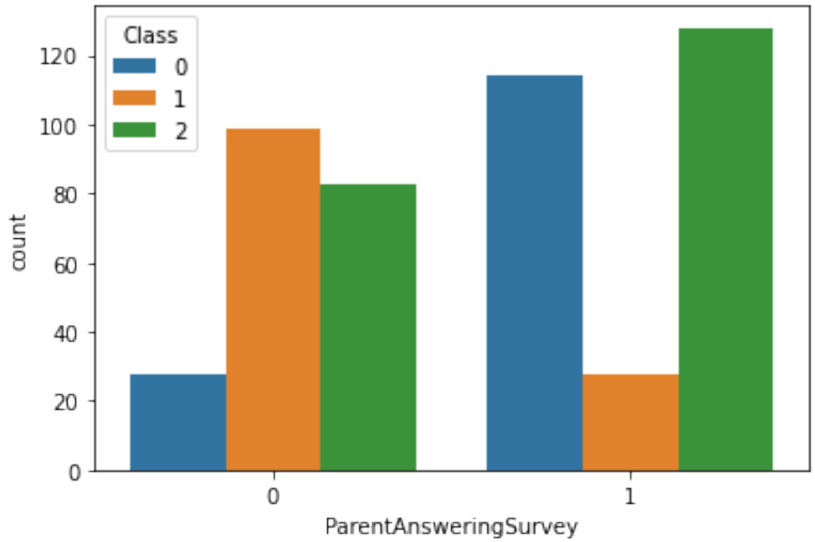


In [53]:
```
mydata.Semester.value_counts()
```

Out[53]:
```
0    245
1    235
Name: Semester, dtype: int64
```

In [55]:
```
sns.countplot(x='ParentAnsweringSurvey',hue='Class',data=mydata)
```

Out[55]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd8de510730>
```

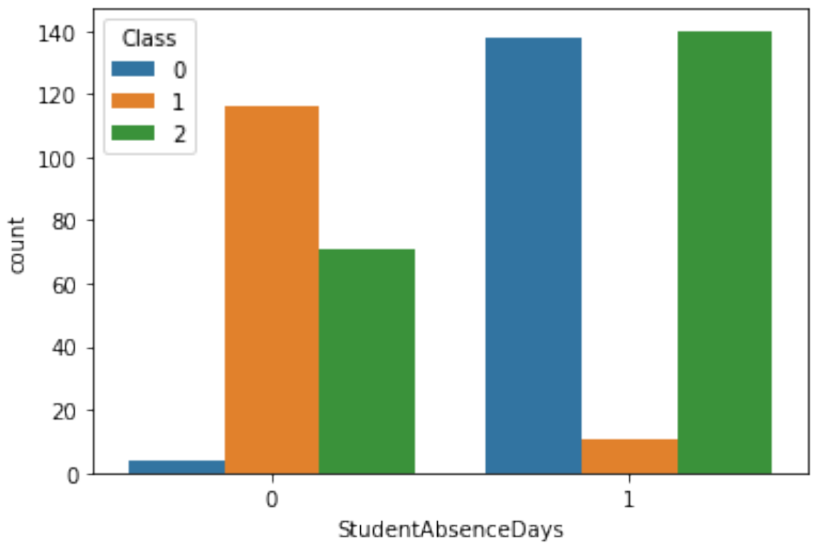


In [56]:
```
mydata.ParentAnsweringSurvey.value_counts()
```

Out[56]:
```
1    270
0    210
Name: ParentAnsweringSurvey, dtype: int64
```

In [57]:
```
sns.countplot(x='StudentAbsenceDays',hue='Class',data=mydata)
```

Out[57]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd8de5e6280>
```

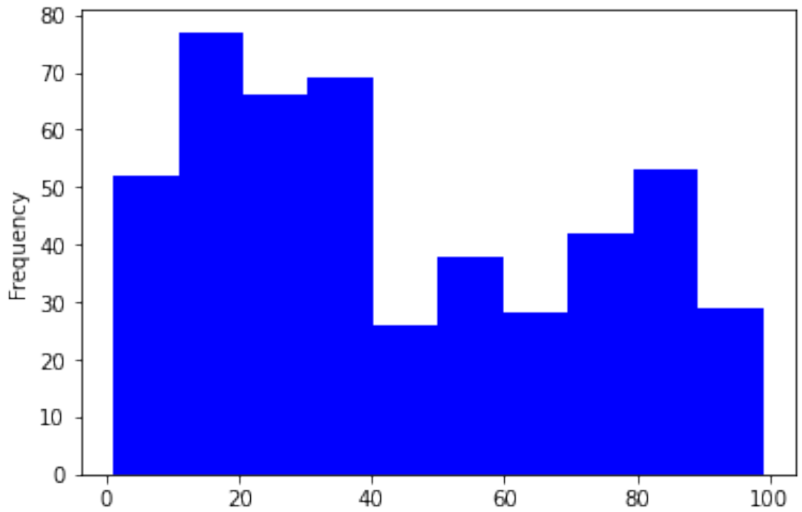


In [58]:
```
mydata.StudentAbsenceDays.value_counts()
```

Out[58]:
```
1    289
0    191
Name: StudentAbsenceDays, dtype: int64
```

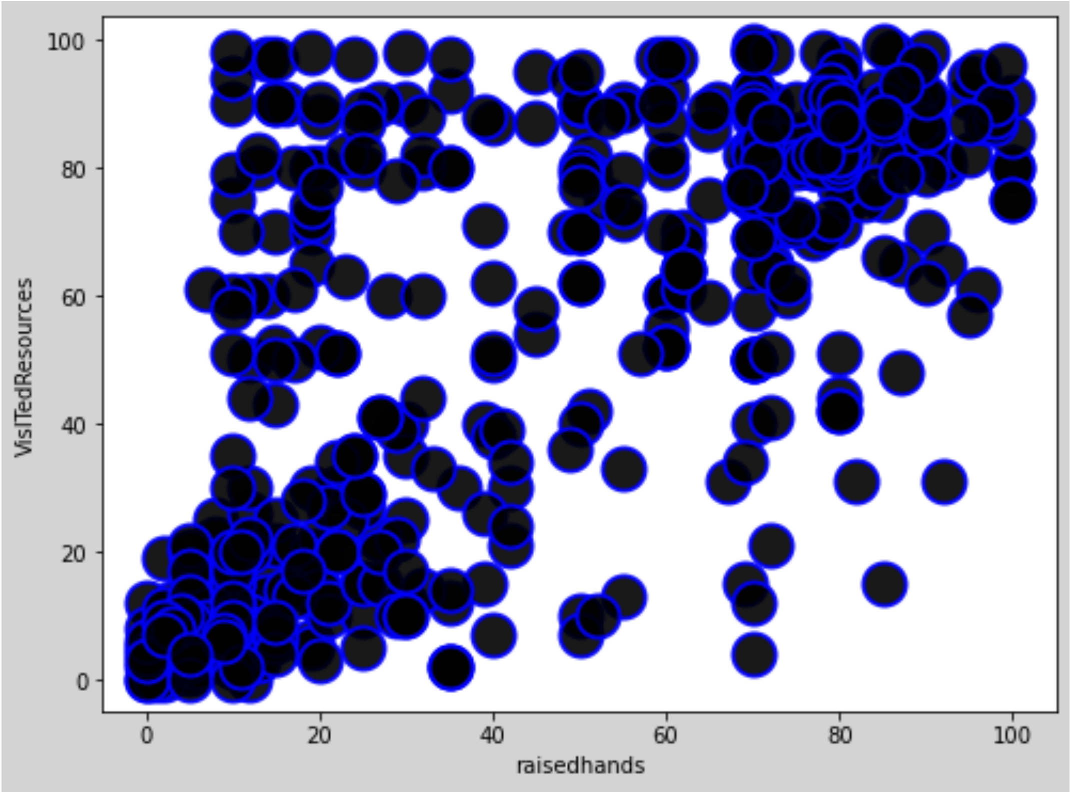
In [63]:
```
mydata.Discussion.plot.hist(color ='Blue')
```

Out[63]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd8dee5e490>
```



In [59]:
```
plt.figure(figsize=(8,6),facecolor="lightgrey")
plt.scatter(mydata.raisedhands,mydata.VisITedResources,
            color="black",alpha=0.9,edgecolors="blue",linewidths=2,s=400)
plt.xlabel("raisedhands")
plt.ylabel("VisITedResources")
```

Out[59]:
```
Text(0, 0.5, 'VisITedResources')
```



Correlation:

Look at some categorical features in relation to each other, to see what insights could be possibly read To find the relationship between the variables.

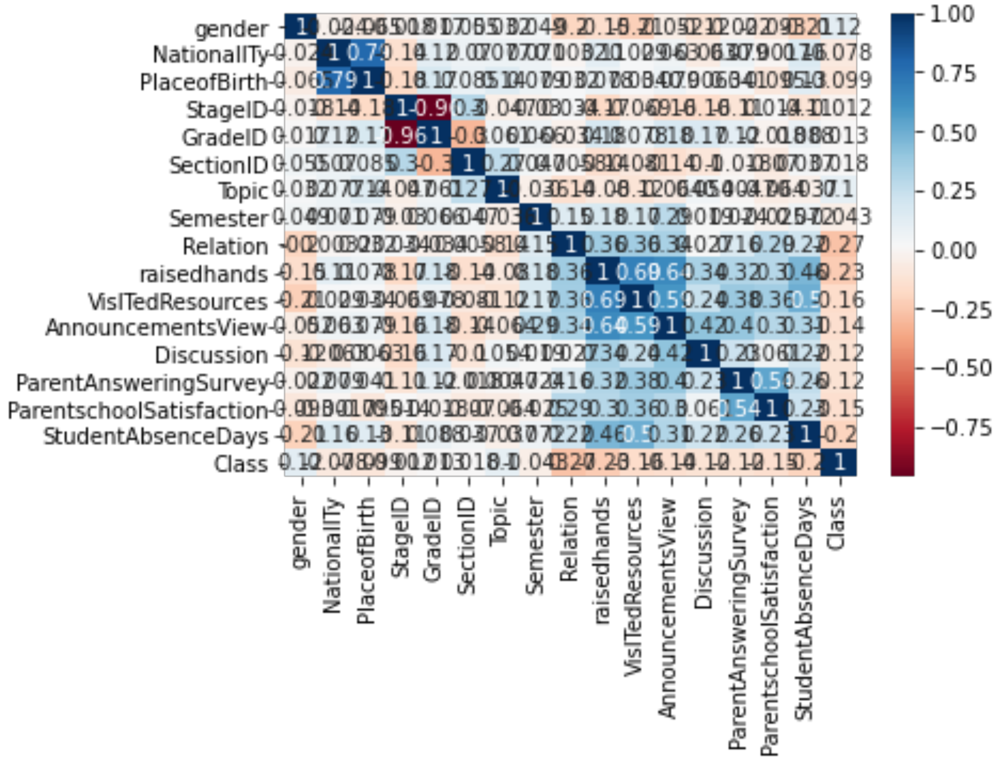
In [68]:
```
mydata_corr= mydata.corr()
mydata_corr
```

Out[68]:

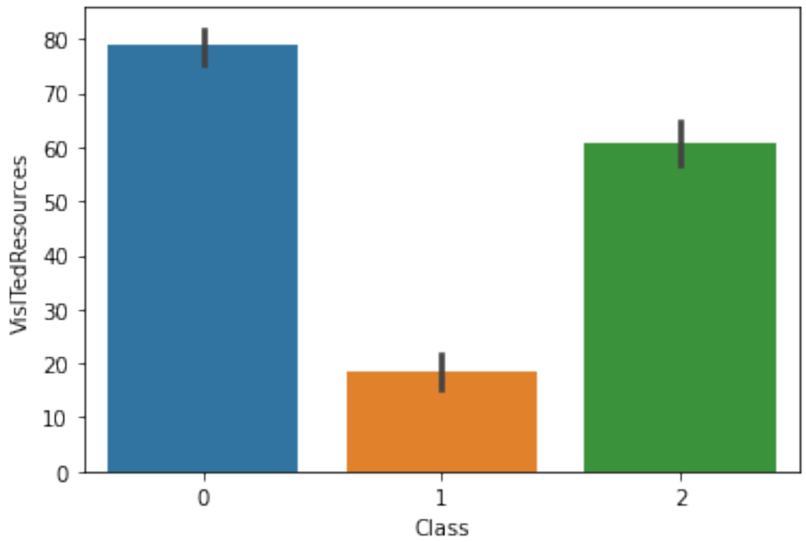
	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation	raise
gender	1.000000	-0.023653	-0.064895	-0.017793	0.016869	0.054907	0.031769	0.049156	-0.195142	-0
NationalITy	-0.023653	1.000000	0.786798	-0.139212	0.124049	0.069712	0.076718	0.070503	0.003212	0
PlaceofBirth	-0.064895	0.786798	1.000000	-0.176368	0.174026	0.085178	0.143477	0.078554	0.031632	0
StageID	-0.017793	-0.139212	-0.176368	1.000000	-0.961835	0.296416	-0.047493	-0.029512	0.034205	-0
GradeID	0.016869	0.124049	0.174026	-0.961835	1.000000	-0.303949	0.061389	0.066079	-0.033602	0
SectionID	0.054907	0.069712	0.085178	0.296416	-0.303949	1.000000	0.267445	0.046763	0.005783	-0
Topic	0.031769	0.076718	0.143477	-0.047493	0.061389	0.267445	1.000000	-0.035975	-0.139487	-0
Semester	0.049156	0.070503	0.078554	-0.029512	0.066079	0.046763	-0.035975	1.000000	0.148705	0
Relation	-0.195142	0.003212	0.031632	0.034205	-0.033602	0.005783	-0.139487	0.148705	1.000000	0
raisedhands	-0.149978	0.111533	0.077986	-0.172751	0.182621	-0.143862	-0.080418	0.178358	0.364237	1
VisITedResources	-0.210932	0.028793	0.033798	-0.068621	0.078262	-0.080909	-0.118144	0.173219	0.360240	0
AnnouncementsView	-0.052139	0.062827	0.078636	-0.163666	0.183033	-0.144955	-0.063856	0.287066	0.339505	0
Discussion	-0.124703	-0.063386	0.006262	-0.161406	0.168462	-0.102538	0.054064	0.019083	0.026720	0
ParentAnsweringSurvey	-0.022359	0.079380	0.040887	-0.114025	0.118246	-0.018449	0.004730	0.023628	0.163811	0
ParentschoolSatisfaction	-0.093478	-0.001701	-0.094594	0.014272	-0.018421	-0.070405	-0.064087	-0.025258	0.287698	0
StudentAbsenceDays	-0.209011	0.157116	0.134554	-0.112536	0.088342	0.037062	-0.036537	0.072462	0.219687	0
Class	0.123675	-0.077785	-0.098975	-0.011696	0.013483	0.017597	0.103610	-0.043287	-0.272111	-0

In [69]:
```
sns.heatmap(mydata_corr, annot= True, cmap = 'RdBu')
```

Out[69]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd8de43eb20>

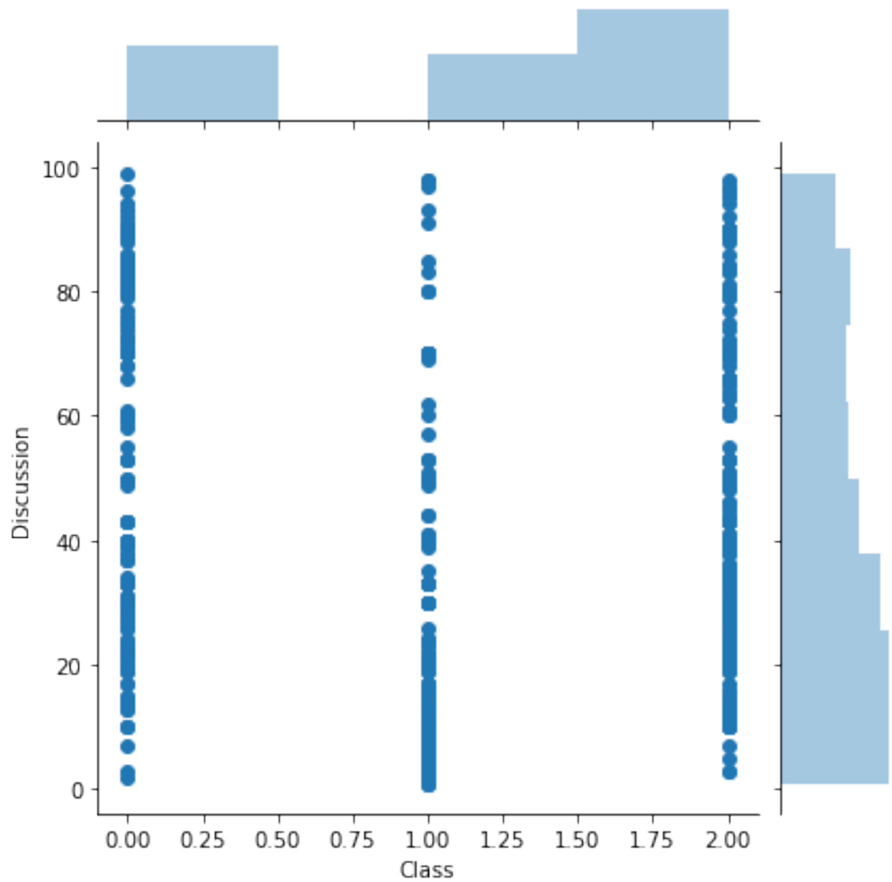


In [70]:
```
sns.barplot(x="Class",y='VisITedResources',data=mydata);
```



In [71]:
```
sns.jointplot(x="Class",y='Discussion',data=mydata)
```

Out[71]:
```
<seaborn.axisgrid.JointGrid at 0x7fd8dfa3a400>
```



Separate independent and dependent variables:

dependent variables

In [13]:
```
y_dep=mydata.iloc[:,16]   # the dependent variable is cateriozed into 3
y_dep
```

Out[13]:
```
0      2
1      2
2      1
3      1
4      2
      ..
475    1
476    2
477    2
478    1
479    1
Name: Class, Length: 480, dtype: int64
```

Independent variable:

In [14]:
```
x_ind = mydata.iloc[:,0:16]
x_ind
```

Out[14]:

	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation	raisedhands	VisITedResources	An
0	1	4	4	2	1	0	7	0	0	15	16	
1	1	4	4	2	1	0	7	0	0	20	20	
2	1	4	4	2	1	0	7	0	0	10	7	
3	1	4	4	2	1	0	7	0	0	30	25	
4	1	4	4	2	1	0	7	0	0	40	50	
...	...	...	...	...	...	...	...	...	...	...	...	
475	0	3	3	1	5	0	2	1	0	5	4	
476	0	3	3	1	5	0	5	0	0	50	77	
477	0	3	3	1	5	0	5	1	0	55	74	
478	0	3	3	1	5	0	6	0	0	30	17	
479	0	3	3	1	5	0	6	1	0	35	14	

480 rows × 16 columns

Machine Learning

Train and test split

In [15]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_ind,y_dep,train_size = 0.8, random_state = 86)
```

Build classification model and present it's classification report

Logistic regression model

In [29]:
```
# The class is cateroize into 3 we can use multinomial in logistic regression.
from sklearn.linear_model import LogisticRegression
model1=LogisticRegression(multi_class ='multinomial', solver ='lbfgs')
model1
```

Out[29]:
```
LogisticRegression(multi_class='multinomial')
```

Model fitting:

In [17]:
```
model1.fit(x_train,y_train)
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:762: ConvergenceWa
rning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable
/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
(https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[17]:
```
LogisticRegression(multi_class='multinomial')
```

Predict the x_test

In [18]:
```
y_pred=model1.predict(x_test)
y_pred
```

Out[18]:
```
array([1, 1, 2, 1, 2, 0, 1, 0, 1, 1, 2, 2, 0, 0, 0, 2, 0, 1, 0, 1, 1, 2,
       0, 2, 2, 1, 2, 0, 2, 1, 2, 0, 2, 0, 1, 2, 2, 1, 2, 2, 1, 2, 2, 2,
       2, 0, 0, 0, 2, 0, 0, 2, 0, 1, 1, 2, 2, 1, 1, 0, 2, 2, 0, 1, 1, 0,
       2, 2, 1, 2, 1, 2, 0, 1, 2, 0, 1, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2,
       2, 2, 1, 2, 2, 2, 1, 2])
```

Performance measures:

Confusion matrix:

It is used to calculate the follwoing performance measures like accuracy,f1 score,precision,recall

In [30]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [20]:
```
confusion_matrix(y_test,y_pred)
```

Out[20]:
```
array([[20,  0, 11],
       [ 0, 23,  5],
       [ 4,  3, 30]])
```

In [21]:
```
from sklearn.preprocessing import StandardScaler
```

In [22]:
```
norm=StandardScaler()
```

In [23]:
```
x_train=norm.fit_transform(x_train)
x_test=norm.fit_transform(x_test)
```

In [24]:
```
accuracy_score(y_test,y_pred)
```

Out[24]:
```
0.7604166666666666
```

Accuracy score:

My model accuracy for this data set is *76%*.

Classification report:

In [25]:
```
from sklearn.metrics import classification_report
```

In [26]:
```
class_report = classification_report(y_test,y_pred)
```

In [27]:
```
print(class_report)
```

```
              precision    recall  f1-score   support

           0       0.83      0.65      0.73        31
           1       0.88      0.82      0.85        28
           2       0.65      0.81      0.72        37

    accuracy                           0.76        96
   macro avg       0.79      0.76      0.77        96
weighted avg       0.78      0.76      0.76        96
```

Conclusion:

The overall accuracy for this dataset is **76%**. As our target variable is categorized into 3 classes. We used to multiclass logistic regression. 76 % of system provides users with a synchronous access to educational resources from any device with Internet connection.

In [ ]:

```
```

In [ ]:

```
```