**Introduction to Big Data Analytics**

**Term Project Report**

**Topic: Student Academic Performance Data Analysis**

***（Powerful translation by DeepL）***

Yu-Lu Zhang 107AEA002

Zi-Jian Li 108AEA001

Xiang Wang 108AEA008

**December 2020**

**Motivations:**

Academic performance is very important to a student, but we don't know what the most important factors are that affect it. We analyze the dataset and develop models to predict student performance.

**Summary:**

We used this dataset(xAPI-Edu-Data) to analyze the data and visualize it to predict student achievement by analyzing the data and developing a model.

**Research Steps:**

Data pre-processing->data visualization->model building analysis->parameter tuning->prediction effect

**Environment:**

Python 3.7 + Jupyter Notebook

**Python required packages：**

pandas、sklearn、seaborn、matplotlib、numpy

**Reference:**

Students' Academic Performance Dataset (xAPI-Edu-Data)

<https://www.kaggle.com/aljarah/xAPI-Edu-Data>

Seaborn API Website

<https://seaborn.pydata.org/api.html>

**Attributes:**

1 Gender - student's gender (nominal: 'Male' or 'Female’)

2 Nationality- student's nationality (nominal:’ Kuwait’,’ Lebanon’,’ Egypt’,’ SaudiArabia’,’ USA’,’ Jordan’,’  
Venezuela’,’ Iran’,’ Tunis’,’ Morocco’,’ Syria’,’ Palestine’,’ Iraq’,’ Lybia’)

3 Place of birth- student's Place of birth (nominal:’ Kuwait’,’ Lebanon’,’ Egypt’,’ SaudiArabia’,’ USA’,’ Jordan’,’  
Venezuela’,’ Iran’,’ Tunis’,’ Morocco’,’ Syria’,’ Palestine’,’ Iraq’,’ Lybia’)

4 Educational Stages- educational level student belongs (nominal: ‘lowerlevel’,’MiddleSchool’,’HighSchool’)

5 Grade Levels- grade student belongs (nominal: ‘G-01’, ‘G-02’, ‘G-03’, ‘G-04’, ‘G-05’, ‘G-06’, ‘G-07’, ‘G-08’, ‘G-09’, ‘G-10’, ‘G-11’, ‘G-12 ‘)

6 Section ID- classroom student belongs (nominal:’A’,’B’,’C’)

7 Topic- course topic (nominal:’ English’,’ Spanish’, ‘French’,’ Arabic’,’ IT’,’ Math’,’ Chemistry’, ‘Biology’, ‘Science’,’ History’,’ Quran’,’ Geology’)

8 Semester- school year semester (nominal:’ First’,’ Second’)

9 Parent responsible for student (nominal:’mom’,’father’)

10 Raised hand- how many times the student raises his/her hand on classroom (numeric:0-100)

11- Visited resources- how many times the student visits a course content(numeric:0-100)

12 Viewing announcements-how many times the student checks the new announcements(numeric:0-100)

13 Discussion groups- how many times the student participate on discussion groups (numeric:0-100)

14 Parent Answering Survey- parent answered the surveys which are provided from school or not  
(nominal:’Yes’,’No’)

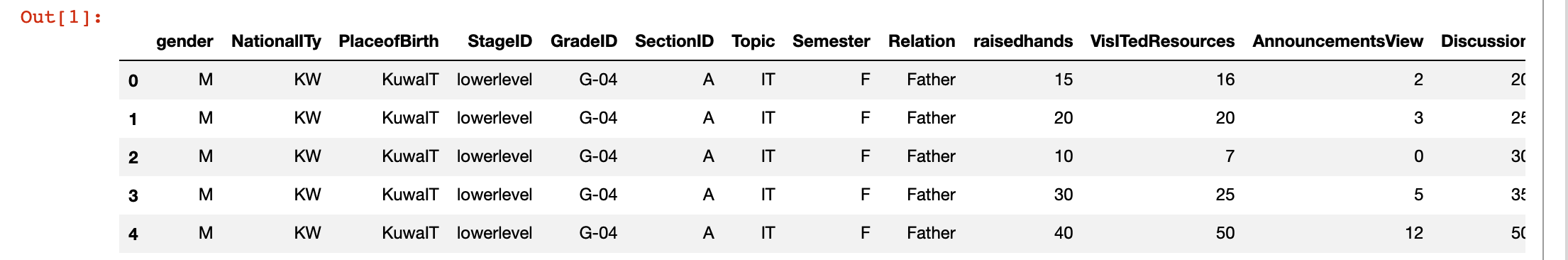
15 Parent School Satisfaction- the Degree of parent satisfaction from school(nominal:’Yes’,’No’)

16 Student Absence Days-the number of absence days for each student (nominal: above-7, under-7)

**Analysis Process:**

**1. Loading packages and dataset**

1. **import** pandas as pd
2. **import** numpy as np
3. **import** seaborn as sns
4. **import** matplotlib.pyplot as plt
6. **from** sklearn **import** preprocessing, svm
7. **from** sklearn.linear\_model **import** Perceptron
8. **from** sklearn.tree **import** DecisionTreeClassifier
10. data = pd.read\_csv('xAPI-Edu-Data.csv')
11. data.head()



View Data

**2. Data Preprocessing**

**2.1** **View Data Sheet**

Input**:** data.info()

Output**:**

1. <**class** 'pandas.core.frame.DataFrame'>
2. RangeIndex: 480 entries, 0 to 479
3. Data columns (total 17 columns):
4. #   Column                    Non-Null Count  Dtype
5. ---  ------                    --------------  -----
6. 0   gender                    480 non-null    object
7. 1   NationalITy               480 non-null    object
8. 2   PlaceofBirth              480 non-null    object
9. 3   StageID                   480 non-null    object
10. 4   GradeID                   480 non-null    object
11. 5   SectionID                 480 non-null    object
12. 6   Topic                     480 non-null    object
13. 7   Semester                  480 non-null    object
14. 8   Relation                  480 non-null    object
15. 9   raisedhands               480 non-null    int64
16. 10  VisITedResources          480 non-null    int64
17. 11  AnnouncementsView         480 non-null    int64
18. 12  Discussion                480 non-null    int64
19. 13  ParentAnsweringSurvey     480 non-null    object
20. 14  ParentschoolSatisfaction  480 non-null    object
21. 15  StudentAbsenceDays        480 non-null    object
22. 16  Class                     480 non-null    object

According to the analysis in the above table, there are no null values and the data does not need to be pre-processed.

**2.2** **View student grades by category**

Input: data.Class.unique()

Output: array(['M', 'L', 'H'], dtype=object)

Student grades are divided into three categories ['L', 'M', 'H'], which will be used as criteria for evaluating students.

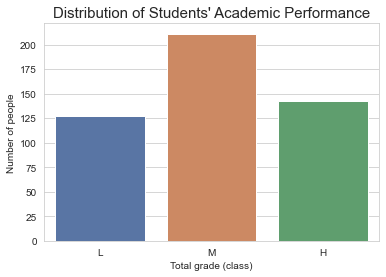
L:0-59 Failure.

M:60-89 Medium.

H:90-100 High scores.

**3. Data Visualization**

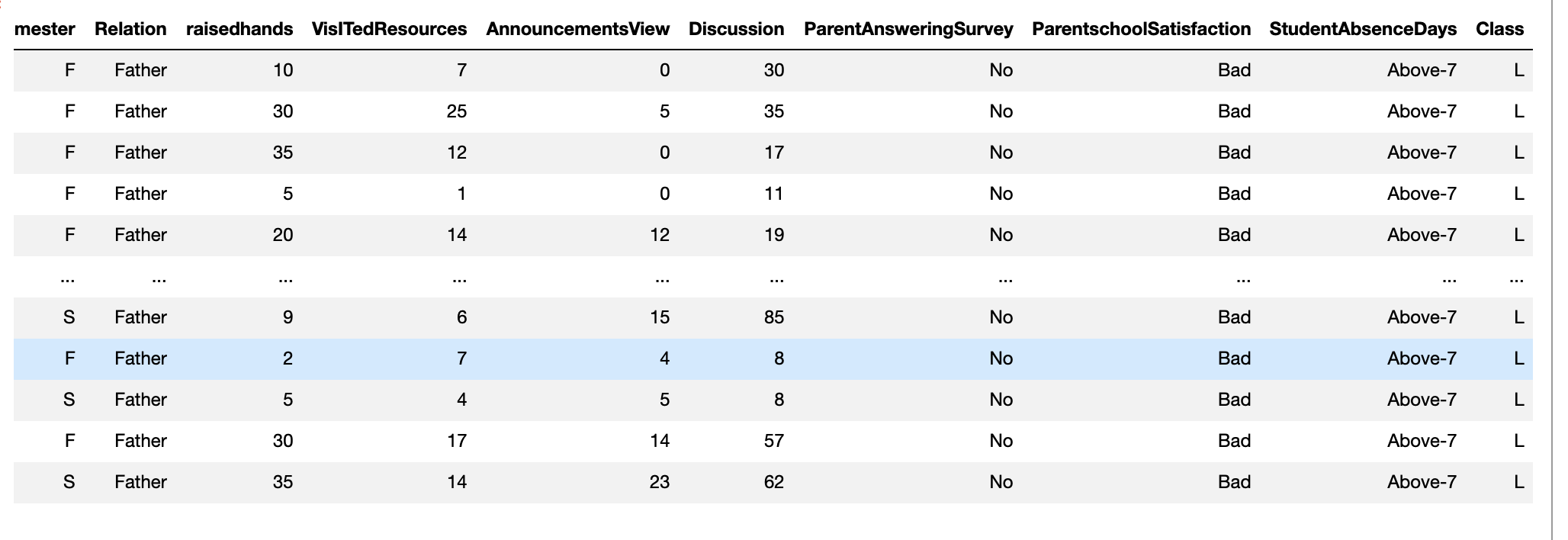
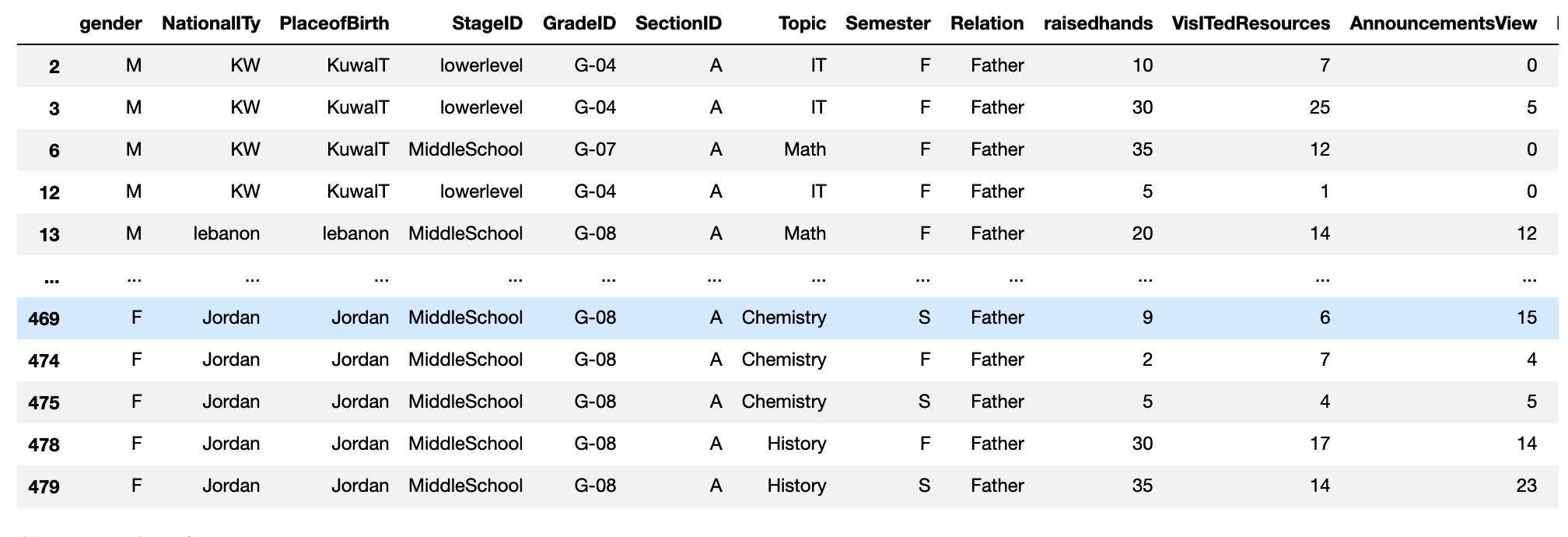
1. sns.set\_style("whitegrid")
2. ax = sns.countplot(x='Class', data=data, order=['L', 'M', 'H'], palette="deep")
3. plt.xlabel('Total grade (class)')
4. plt.ylabel('Number of people')
5. plt.title("Distribution of Students' Academic Performance", size=15)
6. plt.show()



According to the chart above, most students were in the middle of the range, followed by high scores and the fewest number of failing grades.

**3.1** **View information for failing students**

1. data.loc[data["Class"]=='L']

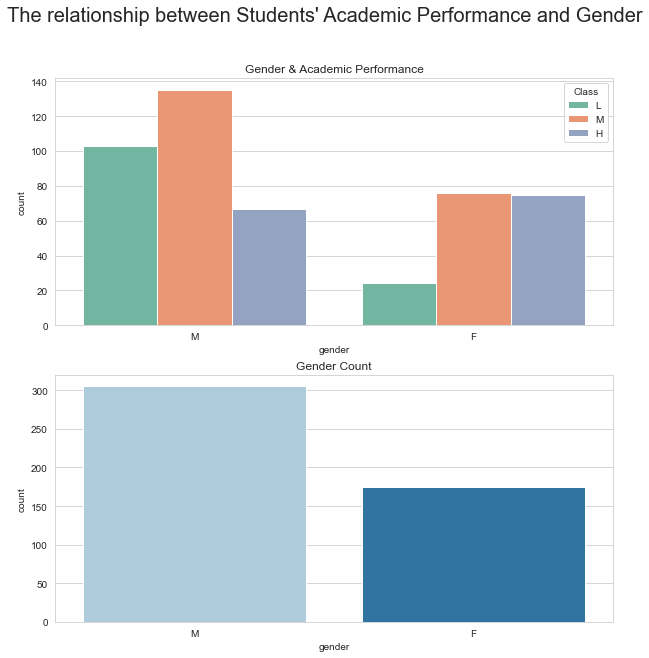


Based on the table above, it appears that the failing students missed more than seven days of school, and all values were concentrated in a low area, such as those who raised their hands less often and did not participate in discussions (see the graph below).

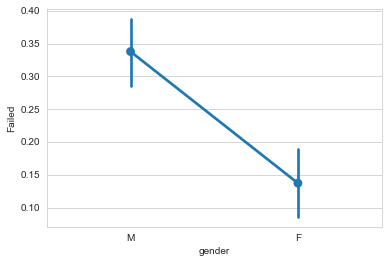
We can also observe that the guardians of the failing students were usually the fathers, and they did not take the school survey and were not very satisfied with the school.

**3.2** **The relationship between Students' Academic Performance and Gender**

1. fig, axarr  = plt.subplots(2,figsize=(10,10))
2. sns.countplot(x='gender', hue='Class', data=data, order=['M', 'F'],hue\_order = ['L', 'M', 'H'], ax=axarr[0], palette="Set2")
3. sns.countplot(x='gender', data=data, order=['M','F'], ax=axarr[1], palette="Paired")
4. axarr[0].set\_title('Gender & Academic Performance')
5. axarr[1].set\_title('Gender Count')
6. fig.suptitle("The relationship between Students' Academic Performance and Gender", size=20)
7. plt.show()



1. sns.pointplot(x='gender', y='Failed', data=data)

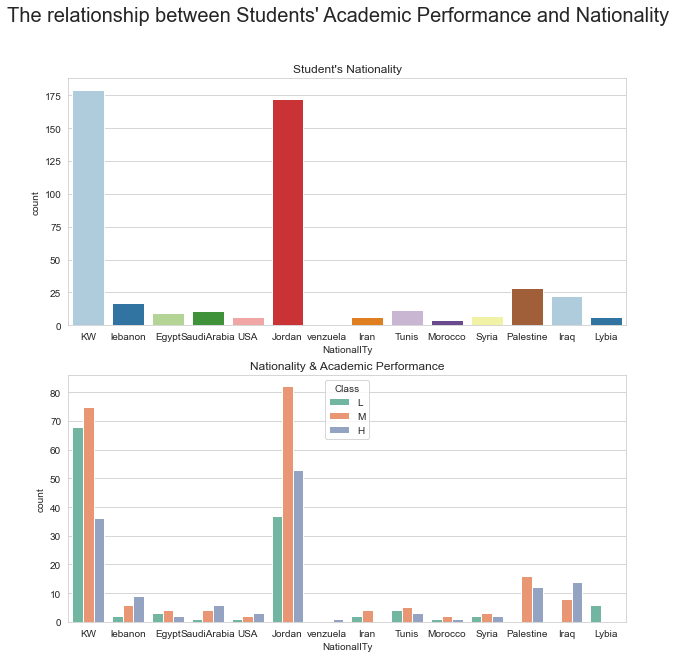


Based on this analysis, we can clearly see that the number of female students failing is much smaller than that of male students, and that the number of female students in the middle and high grades is almost equal.

The difference in the number of male and female students at the high end of the scale is not large. From this we can conclude that gender may affect students' performance.

**3.3The relationship between Students' Academic Performance and Nationality**

1. fig, axarr  = plt.subplots(2,figsize=(10,10))
2. axarr[0].set\_title("Student's Nationality")
3. axarr[1].set\_title('Nationality & Academic Performance')
4. fig.suptitle("The relationship between Students' Academic Performance and Nationality", size=20)
5. sns.countplot(x='NationalITy', data=data, ax=axarr[0], palette="Paired")
6. sns.countplot(x='NationalITy', hue='Class', data=data,hue\_order = ['L', 'M', 'H'], ax=axarr[1], palette="Set2")
7. plt.show()



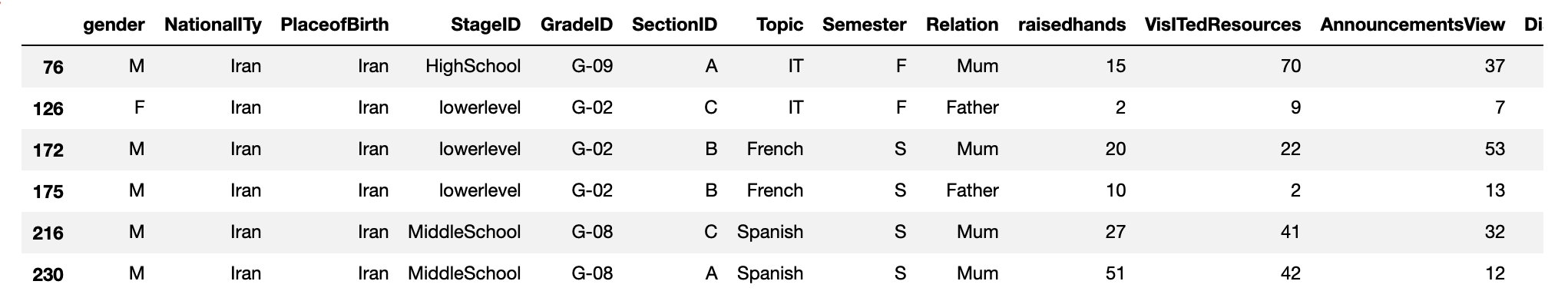
Based on this graph, except for the KW and Jordan students, the sample size for the other nationalities is relatively small and no valid information can be deduced from this analysis.

What we can deduce is that Jordan's students failed half as many as KW's students and had better overall grades than KW's students, while Iranian and Lybia's students did not get high grades.

Let's look at a sample of the Iranian and Lybia students and try to analyze the factors that affect their academic performance.

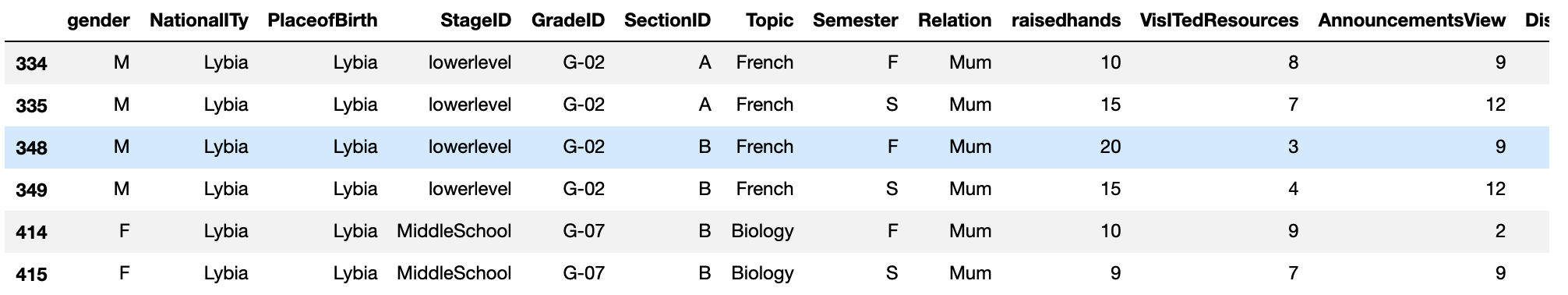
To see the samples of Iranian students:

1. data.loc[data['NationalITy'] == 'Iran']



To see the samples of Lybia students:

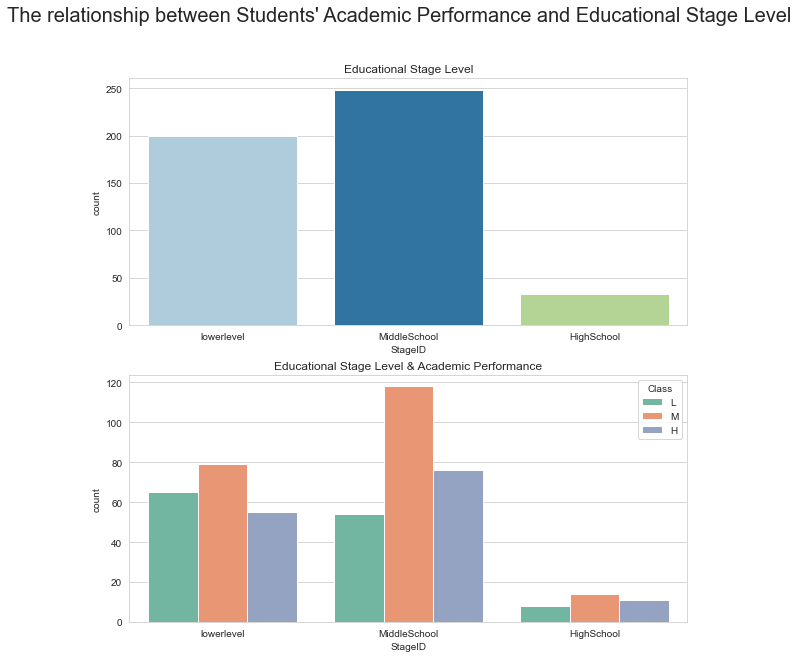
1. data.loc[data['NationalITy'] == 'Lybia']



From our observations, we know that there seems to be a high degree of overlap between the data for Lybia students and all students who did not pass the exam (missing more than 7 days, low values, no school survey, etc.).

**3.4The relationship between Students' Academic Performance and Educational Stage Level**

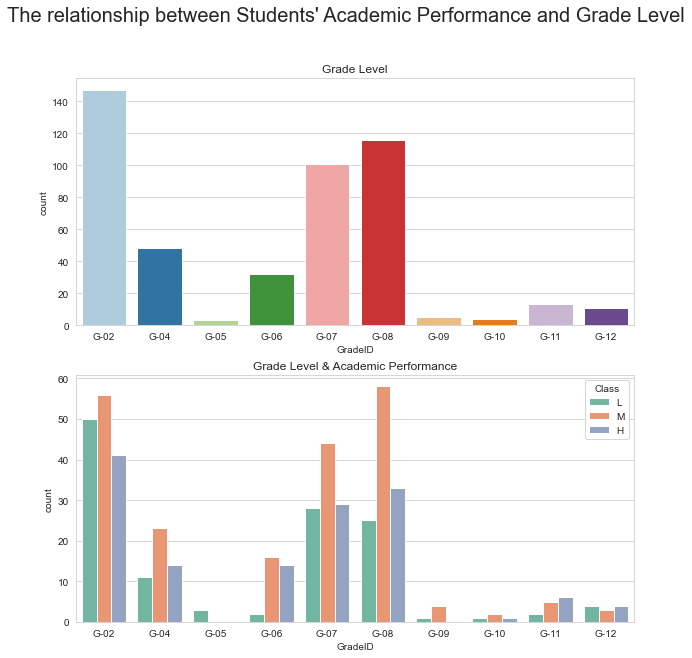
1. fig, axarr  = plt.subplots(2,figsize=(10,10))
2. axarr[0].set\_title('Educational Stage Level')
3. axarr[1].set\_title('Educational Stage Level & Academic Performance')
4. fig.suptitle("The relationship between Students' Academic Performance and Educational Stage Level", size=20)
5. sns.countplot(x='StageID', data=data, ax=axarr[0], palette="Paired")
6. sns.countplot(x='StageID', hue='Class', data=data, hue\_order = ['L', 'M', 'H'], ax=axarr[1], palette="Set2")
7. plt.show()



Based on this graph, we can see that students are over-represented in the middle grades regardless of their educational level.

**3.5 The relationship between Students' Academic Performance and Grade Level**

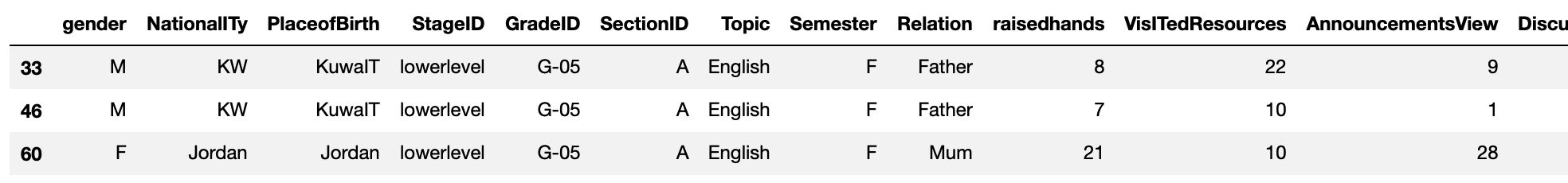
1. fig, axarr  = plt.subplots(2,figsize=(10,10))
2. axarr[0].set\_title('Grade Level')
3. axarr[1].set\_title('Grade Level & Academic Performance')
4. fig.suptitle("The relationship between Students' Academic Performance and Grade Level", size=20)
5. sns.countplot(x='GradeID',
6. data=data,
7. order=['G-02', 'G-04', 'G-05', 'G-06', 'G-07', 'G-08', 'G-09', 'G-10', 'G-11', 'G-12'],
8. ax=axarr[0], palette="Paired")
9. sns.countplot(x='GradeID',
10. hue='Class',
11. data=data,
12. order=['G-02', 'G-04', 'G-05', 'G-06', 'G-07', 'G-08', 'G-09', 'G-10', 'G-11', 'G-12'],
13. hue\_order = ['L', 'M', 'H'],
14. ax=axarr[1], palette="Set2")
15. plt.show()



Based on this analysis, we can see that the number of students in grades 5, 9, and 10 is very small. In addition, no fifth-grader passed and no ninth-grader scored high marks.

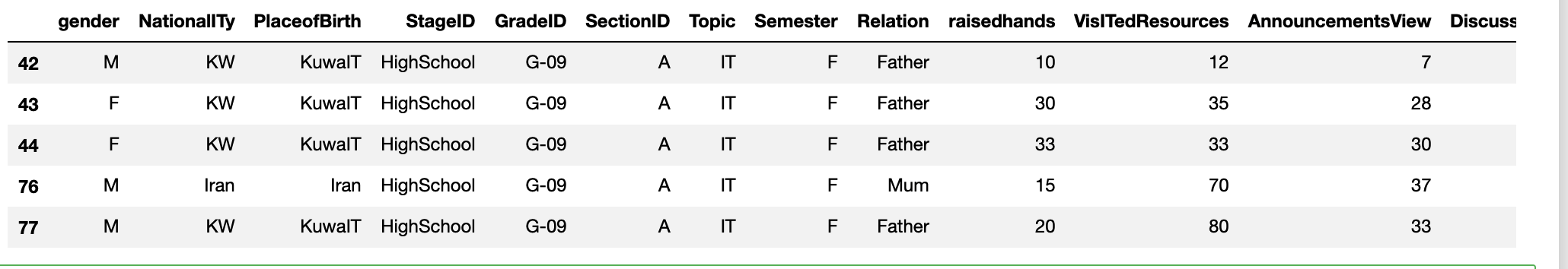
To see the samples of fifth-grade students:

1. data.loc[data['GradeID']=='G-05']



To see the samples of ninth-grade students:

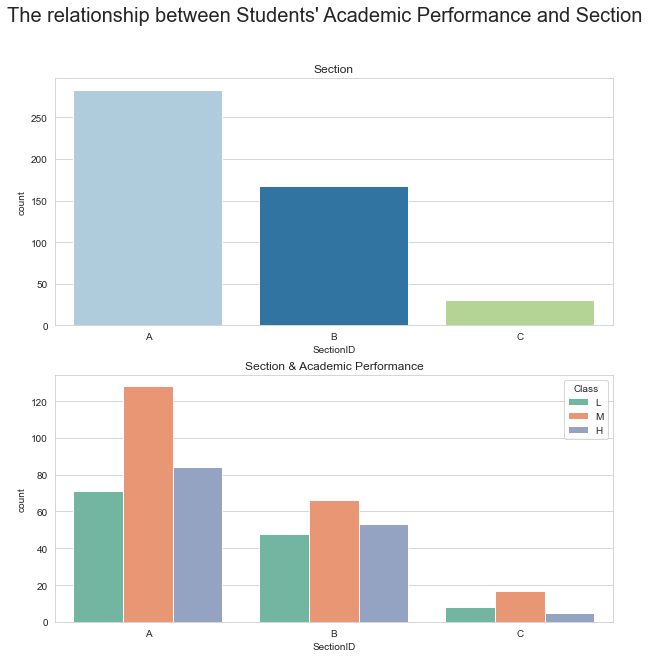
1. data.loc[data['GradeID']=='G-09']



After observation, there seems to be a high degree of overlap between the data for 5th and 9th graders and all students who did not pass the test (missed more than 7 days, low values, no school survey, etc.).

**3.6 The relationship between Students' Academic Performance and Section**

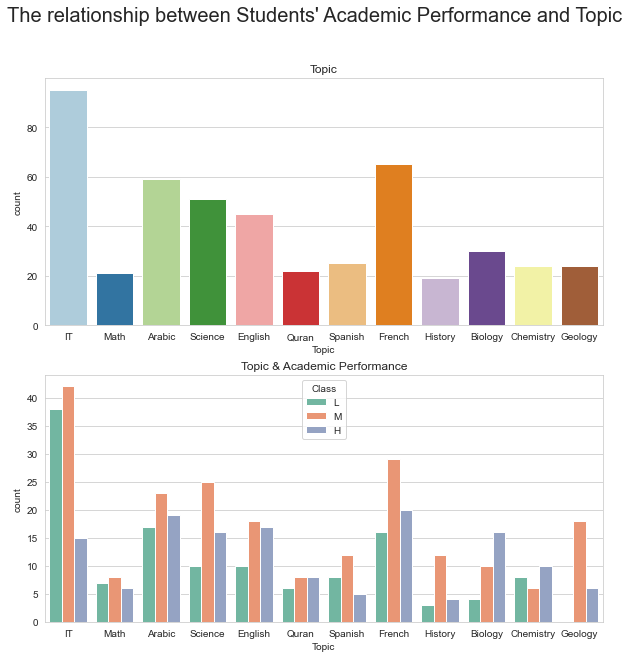
1. fig, axarr  = plt.subplots(2,figsize=(10,10))
2. axarr[0].set\_title('Section')
3. axarr[1].set\_title('Section & Academic Performance')
4. fig.suptitle("The relationship between Students' Academic Performance and Section", size=20)
5. sns.countplot(x='SectionID', data=data,
6. order=['A', 'B', 'C'], ax = axarr[0], palette="Paired")
7. sns.countplot(x='SectionID', hue='Class',
8. data=data, order=['A', 'B', 'C'],
9. hue\_order = ['L', 'M', 'H'], ax = axarr[1], palette="Set2")
10. plt.show()



Based on this graphical analysis, we know that the overall trend for all three classes is similar, and we cannot derive any valid information.

**3.7 The relationship between Students' Academic Performance and Topic**

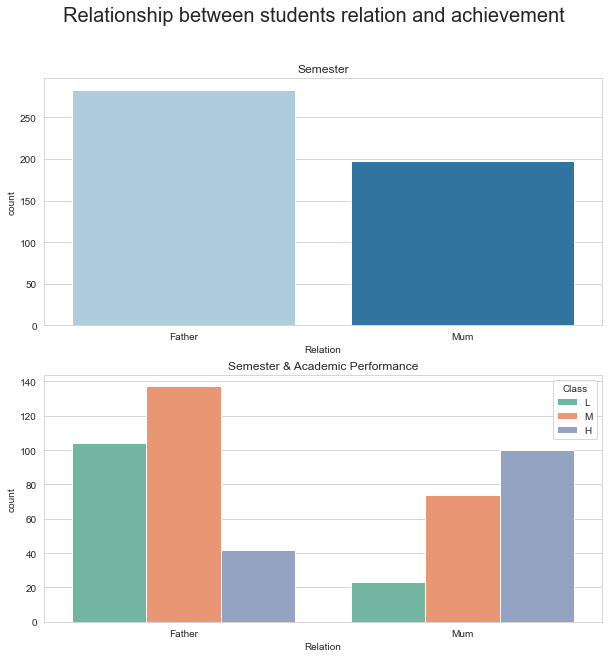
1. fig, axarr  = plt.subplots(2,figsize=(10,10))
2. axarr[0].set\_title('Topic')
3. axarr[1].set\_title('Topic & Academic Performance')
4. fig.suptitle("The relationship between Students' Academic Performance and Topic", size=20)
5. sns.countplot(x='Topic', data=data, ax = axarr[0], palette="Paired")
6. sns.countplot(x='Topic', hue='Class', data=data,hue\_order = ['L', 'M', 'H'], ax = axarr[1], palette="Set2")
7. plt.show()



Based on this graph analysis, we can see an interesting phenomenon that there are no failing students in Geology. Why is that?

**3.8 The relationship between Students' Academic Performance and Semester**

1. fig, axarr  = plt.subplots(2,figsize=(10,10))
2. axarr[0].set\_title('Semester')
3. axarr[1].set\_title('Semester & Academic Performance')
4. fig.suptitle("The relationship between Students' Academic Performance and Semester", size=20)
5. sns.countplot(x='Semester', data=data, ax = axarr[0], palette="Paired")
6. sns.countplot(x='Semester', hue='Class', data=data,hue\_order = ['L', 'M', 'H'], ax = axarr[1], palette="Set2")
7. plt.show()

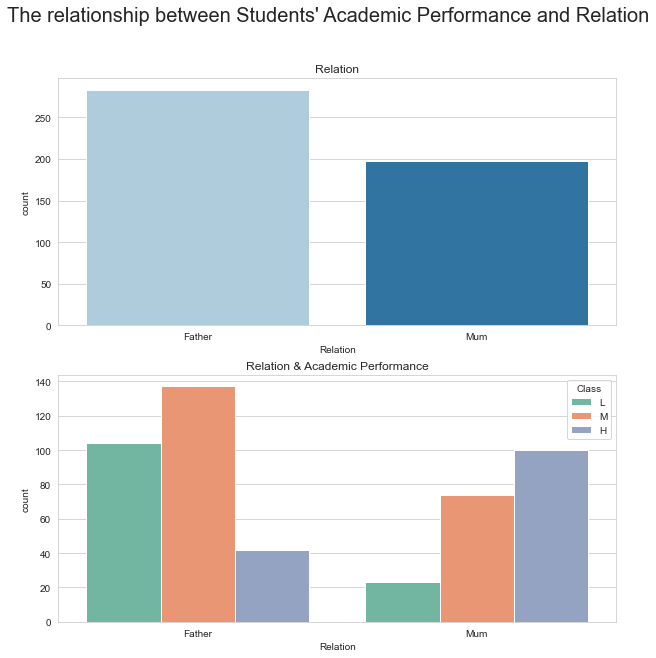


According to this graph, there were fewer failures in the second year than in the first year, and more high scorers than in the first year.

From this, we can conclude that the academic year may affect the grade.

**3.9 The relationship between Students' Academic Performance and Relation**

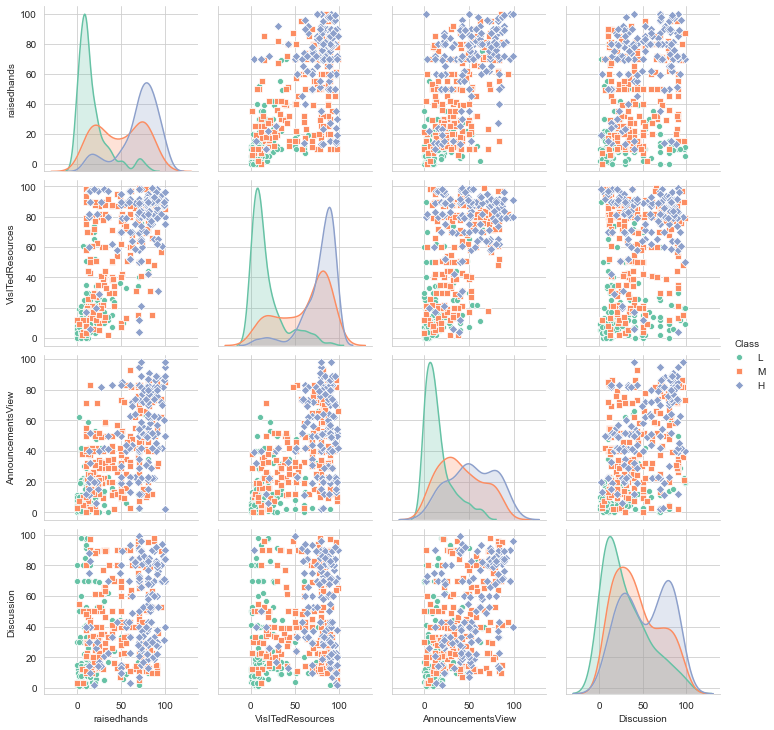
1. fig, axarr  = plt.subplots(2,figsize=(10,10))
2. axarr[0].set\_title('Relation')
3. axarr[1].set\_title('Relation & Academic Performance')
4. fig.suptitle("The relationship between Students' Academic Performance and Relation", size=20)
5. sns.countplot(x='Relation', data=data, ax = axarr[0], palette="Paired")
6. sns.countplot(x='Relation', hue='Class', data=data,hue\_order = ['L', 'M', 'H'], ax = axarr[1], palette="Set2")
7. plt.show()



According to the analysis of the two figures above, there seems to be a correlation between the mother as guardian and the student passing, and a correlation between the father as guardian and the student failing.

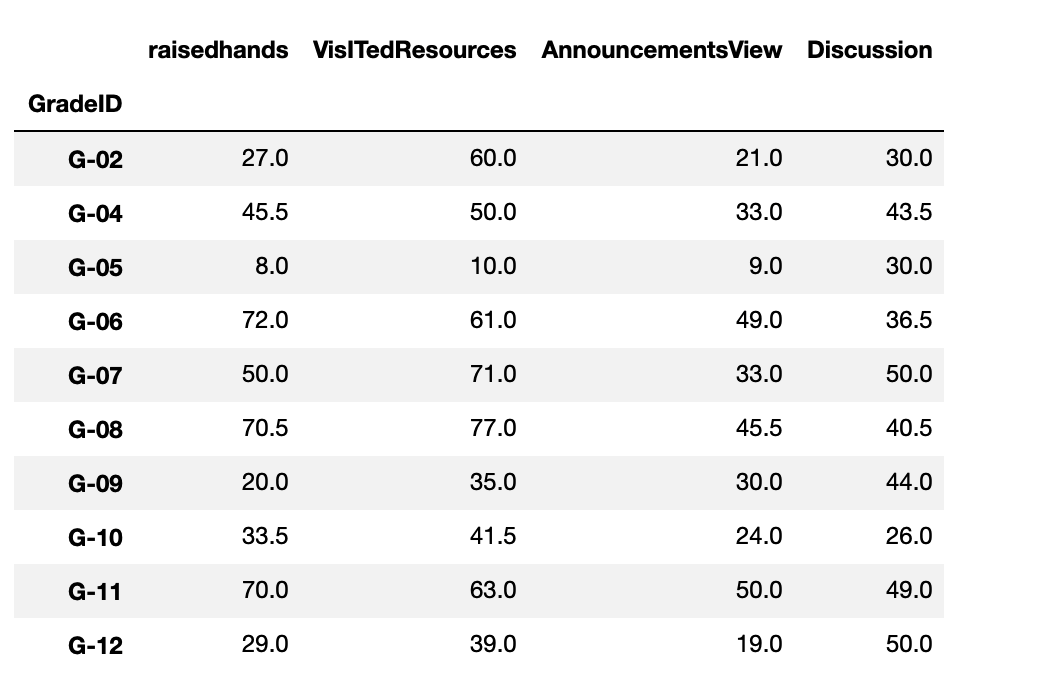
**3.10** **The relationship between the number of times students raised their hands in class, the number of times they visited course content, the number of times they checked new announcements, the number of times they participated in discussions, and their grades**

1. sns.pairplot(data, hue="Class",
2. diag\_kind="kde",
3. hue\_order = ['L', 'M', 'H'],
4. markers=["o", "s", "D"], palette="Set2")
5. plt.show()



View raisedhands, VisiTedResources, AnnouncementsView, Discussion at different educational levels, and get the median here:

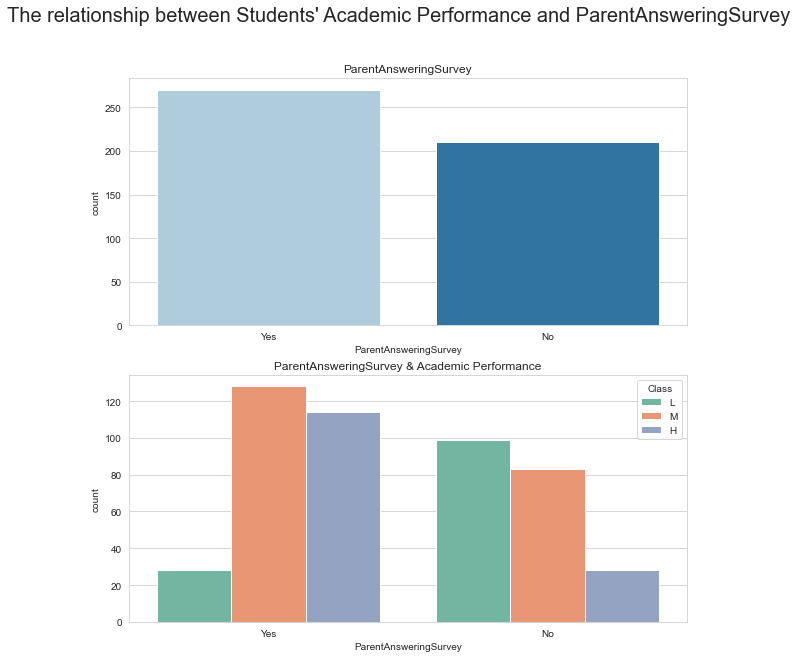
1. data.groupby('GradeID').median()



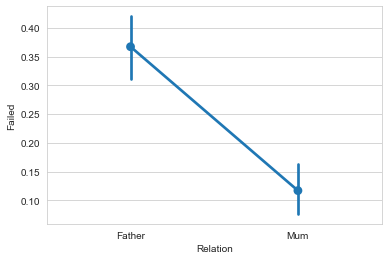
Here we can see that the data for grades 5 and 9 are much lower than most other grades.

**3.11 The relationship between Students' Academic Performance and ParentAnsweringSurvey**

1. fig, axarr  = plt.subplots(2,figsize=(10,10))
2. axarr[0].set\_title('ParentAnsweringSurvey')
3. axarr[1].set\_title('ParentAnsweringSurvey & Academic Performance')
4. fig.suptitle("The relationship between Students' Academic Performance and ParentAnsweringSurvey", size=20)
5. sns.countplot(x='ParentAnsweringSurvey', data=data,
6. order=['Yes', 'No'], ax = axarr[0], palette="Paired")
7. sns.countplot(x='ParentAnsweringSurvey', hue='Class',
8. data=data, order=['Yes', 'No'], hue\_order = ['L', 'M', 'H'],
9. ax = axarr[1], palette="Set2")
10. plt.show()



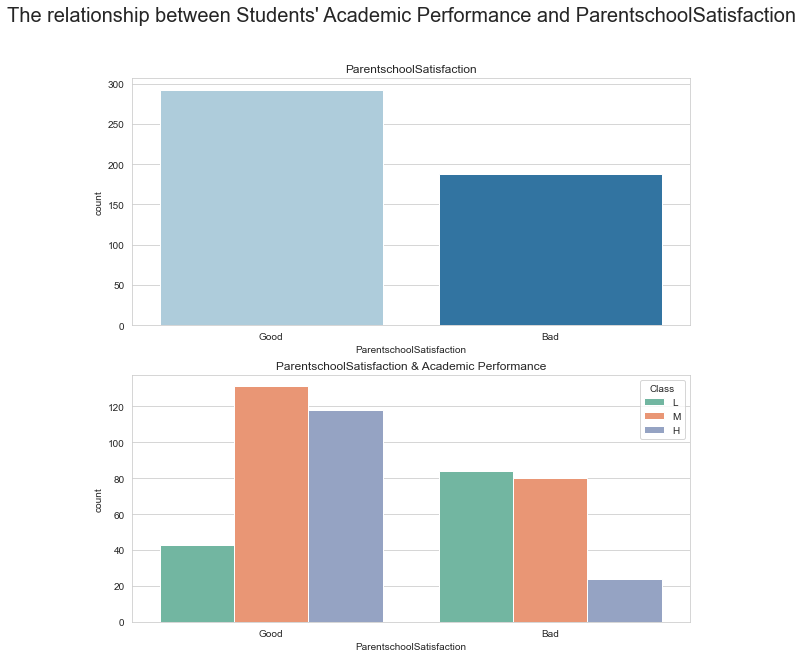
1. sns.pointplot(x='Relation', y='Failed', data=data)



Among high-achieving students, the vast majority of parents were satisfied with the education they received. Students whose parents were least satisfied with the school fared much worse. Students whose mothers were responsible for them fared better.

**3.12The relationship between Students' Academic Performance and ParentschoolSatisfaction**

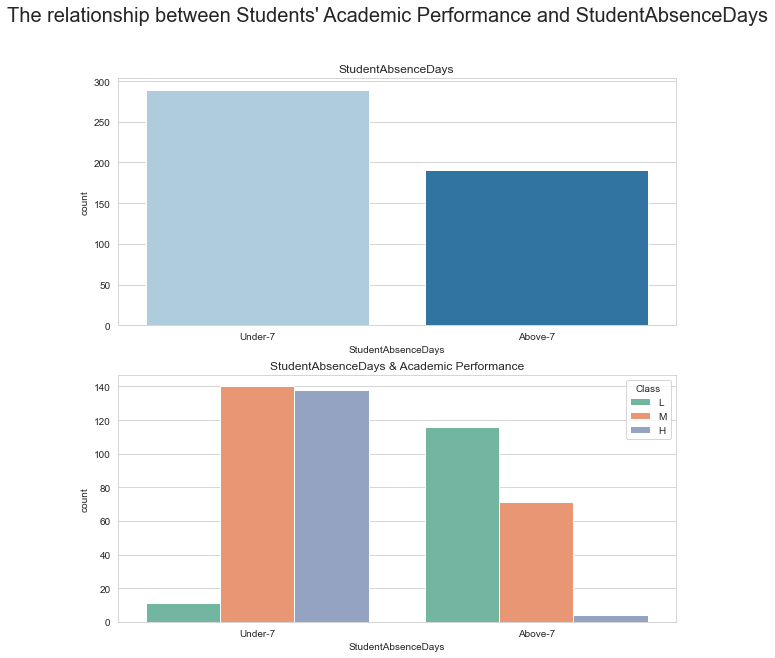
1. fig, axarr  = plt.subplots(2,figsize=(10,10))
2. axarr[0].set\_title('ParentschoolSatisfaction')
3. axarr[1].set\_title('ParentschoolSatisfaction & Academic Performance')
4. fig.suptitle("The relationship between Students' Academic Performance and ParentschoolSatisfaction", size=20)
5. sns.countplot(x='ParentschoolSatisfaction', data=data,
6. order=['Good', 'Bad'], ax = axarr[0], palette="Paired")
7. sns.countplot(x='ParentschoolSatisfaction', hue='Class',
8. data=data, order=['Good', 'Bad'],
9. hue\_order = ['L', 'M', 'H'], ax = axarr[1], palette="Set2")
10. plt.show()



The same observations as in 3.11 are not described.

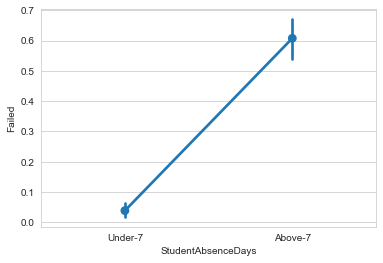
**3.13 The relationship between Students' Academic Performance and StudentAbsenceDays**

1. fig, axarr  = plt.subplots(2,figsize=(10,10))
2. axarr[0].set\_title('StudentAbsenceDays')
3. axarr[1].set\_title('StudentAbsenceDays & Academic Performance')
4. fig.suptitle("The relationship between Students' Academic Performance and StudentAbsenceDays", size=20)
5. sns.countplot(x='StudentAbsenceDays', data=data,
6. order=['Under-7', 'Above-7'],
7. ax = axarr[0], palette="Paired")
8. sns.countplot(x='StudentAbsenceDays', hue='Class',
9. data=data, order=['Under-7', 'Above-7'],
10. hue\_order = ['L', 'M', 'H'],
11. ax = axarr[1], palette="Set2")
12. plt.show()



According to this graph, there is a strong correlation between study time and student performance. Students who have missed more than seven days of school rarely get a high grade, and students who have missed less than seven days of school rarely fail.

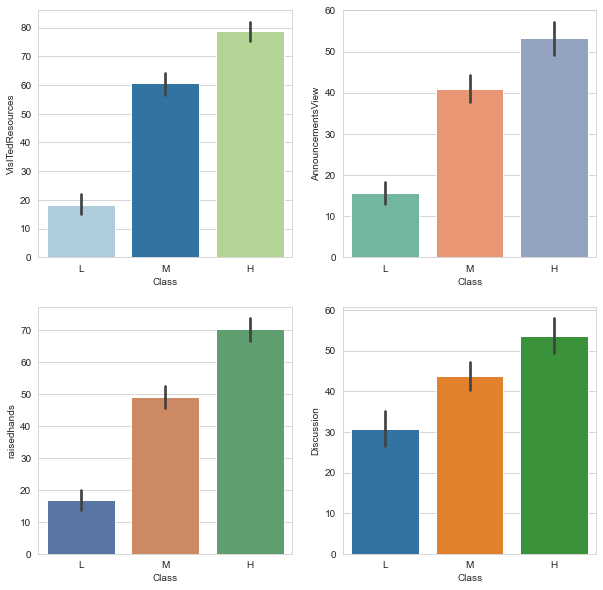
1. data['Failed'] = np.where(data['Class']=='L',1,0)
2. sns.pointplot(x='StudentAbsenceDays', y='Failed', data=data)



The largest intuitive trend can be seen in the frequency of student absences. Low-performing students had more than seven absences, while high-performing students hardly ever had more than seven absences.

**3.14** **A bar graph of the number of times students raised their hands in class, the number of times they visited course content, the number of times they checked new announcements, the number of discussions they attended, and their grades**

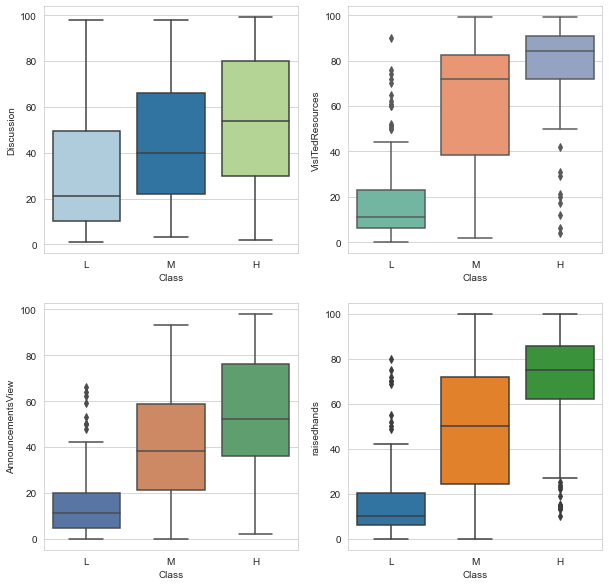
1. fig, axarr  = plt.subplots(2,2,figsize=(10,10))
2. sns.barplot(x='Class', y='VisITedResources', data=data, order=['L','M','H'], ax=axarr[0,0], palette="Paired")
3. sns.barplot(x='Class', y='AnnouncementsView', data=data, order=['L','M','H'], ax=axarr[0,1], palette="Set2")
4. sns.barplot(x='Class', y='raisedhands', data=data, order=['L','M','H'], ax=axarr[1,0], palette="deep")
5. sns.barplot(x='Class', y='Discussion', data=data, order=['L','M','H'], ax=axarr[1,1], palette="tab10")



As expected, those who participate more (more discussions, more hands, more bulletin views, more hands) perform better ...... This is the correlation and causality thing.

**3.15** **Classroom Activity Contrast**

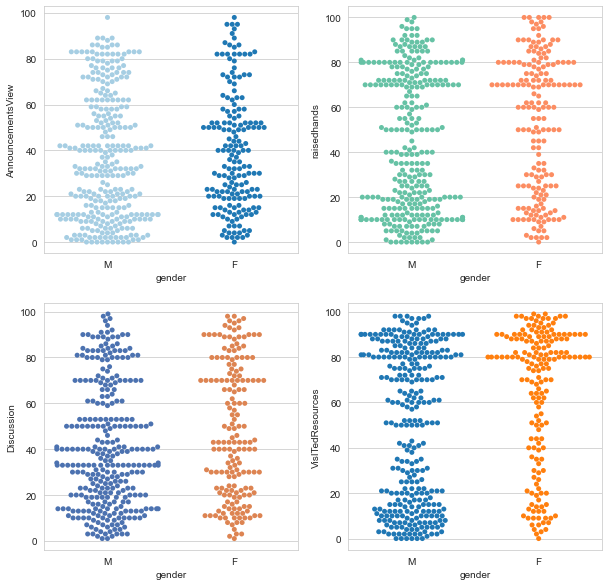
1. fig, axarr = plt.subplots(2, 2,figsize=(10,10))
2. sns.boxplot(x='Class', y='Discussion', data=data, order=['L','M','H'], ax=axarr[0,0], palette="Paired")
3. sns.boxplot(x='Class', y='VisITedResources', data=data, order=['L','M','H'], ax=axarr[0,1], palette="Set2")
4. sns.boxplot(x='Class', y='AnnouncementsView', data=data, order=['L','M','H'], ax=axarr[1,0], palette="deep")
5. sns.boxplot(x='Class', y='raisedhands', data=data, order=['L','M','H'], ax=axarr[1,1], palette="tab10")



According to the above analysis, access to course content may not be as necessary for good performance as discussion, and raising hands may not be as necessary for good performance as checking for new announcements.

**3.16** **Contrast of Gender and Classroom Participation**

1. fig, axarr = plt.subplots(2, 2,figsize=(10,10))
2. sns.swarmplot(x='gender', y='AnnouncementsView', data=data, ax=axarr[0,0], palette="Paired")
3. sns.swarmplot(x='gender', y='raisedhands', data=data, ax=axarr[0,1], palette="Set2")
4. sns.swarmplot(x='gender', y='Discussion', data=data, ax=axarr[1,0], palette="deep")
5. sns.swarmplot(x='gender', y='VisITedResources', data=data, ax=axarr[1,1], palette="tab10")



This swarm diagram tells us that students with low scores (L) access a much hotter resource than students with M or H. In addition, women with high scores (H) access almost exclusively online resources. In addition, women with high scores (H) visited almost exclusively a large number of online resources.

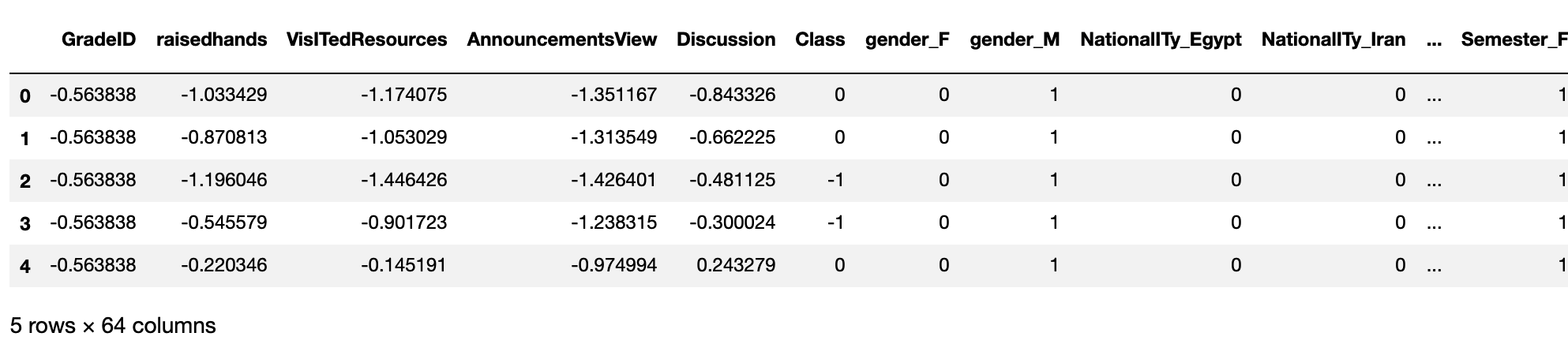
In summary, student grades were related to attributes such as number of visits to course content, days absent, number of hands raised in class, number of times checked for new announcements, participation in discussions, gender, guardian, and semester.

**4. Modeling the relationship between predictions and student performance**

**4.1** **Processing Data**

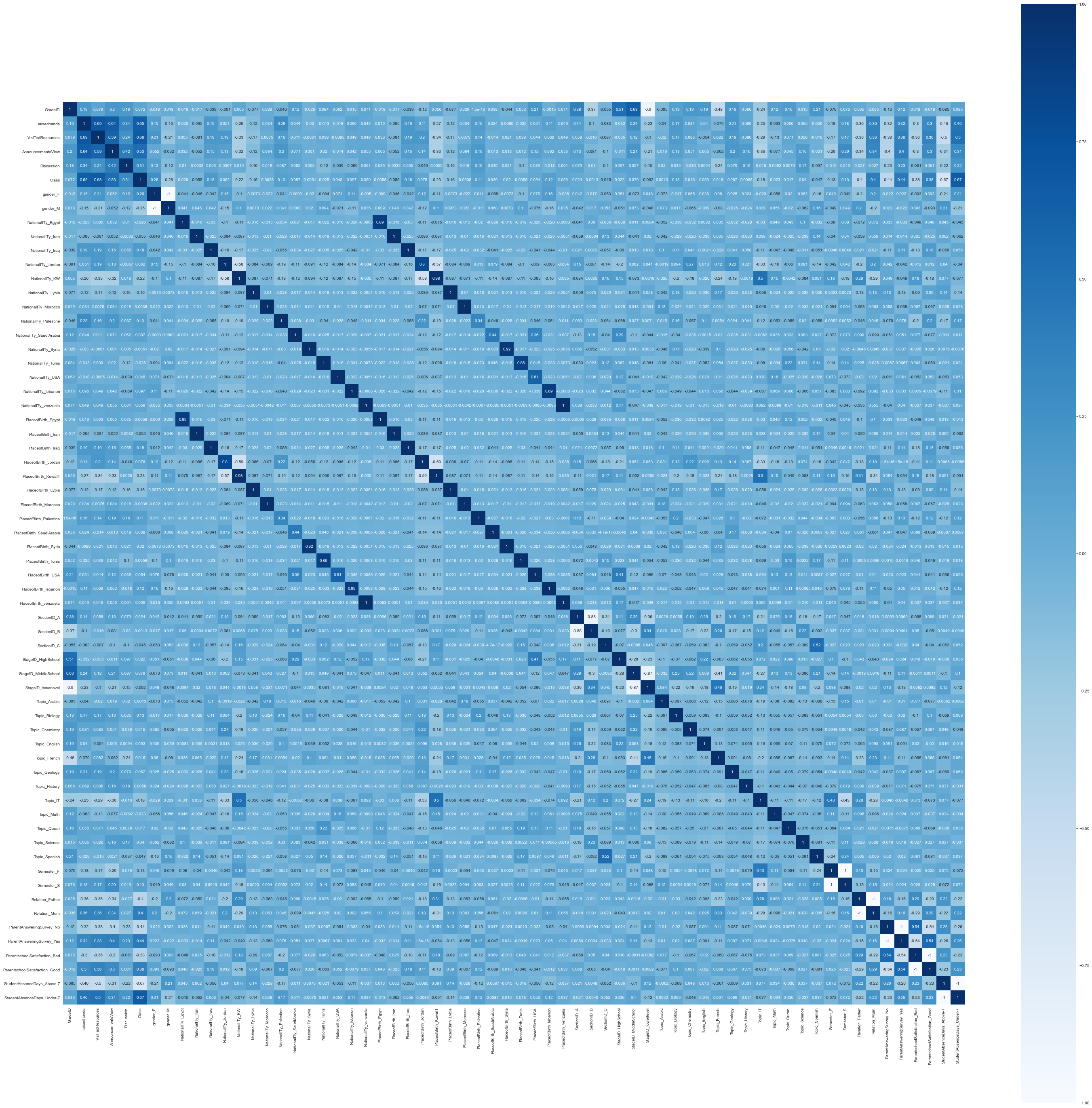
1. # Convert grades into data
2. gradeID\_dict = {"G-01" : 1,
3. "G-02" : 2,
4. "G-03" : 3,
5. "G-04" : 4,
6. "G-05" : 5,
7. "G-06" : 6,
8. "G-07" : 7,
9. "G-08" : 8,
10. "G-09" : 9,
11. "G-10" : 10,
12. "G-11" : 11,
13. "G-12" : 12}
15. data = data.replace({"GradeID" : gradeID\_dict})
16. # Convert scores into data
17. class\_dict = {"L" : -1,
18. "M" : 0,
19. "H" : 1}
20. data = data.replace({"Class" : class\_dict})
22. # Convert to Scale data
23. data["GradeID"] = preprocessing.scale(data["GradeID"])
24. data["raisedhands"] = preprocessing.scale(data["raisedhands"])
25. data["VisITedResources"] = preprocessing.scale(data["VisITedResources"])
26. data["AnnouncementsView"] = preprocessing.scale(data["AnnouncementsView"])
27. data["Discussion"] = preprocessing.scale(data["Discussion"])
29. # Use virtual code conversion to convert 11 columns into 64 columns
30. data = pd.get\_dummies(data, columns=["gender",
31. "NationalITy",
32. "PlaceofBirth",
33. "SectionID",
34. "StageID",
35. "Topic",
36. "Semester",
37. "Relation",
38. "ParentAnsweringSurvey",
39. "ParentschoolSatisfaction",
40. "StudentAbsenceDays"])

43. data.head()

****

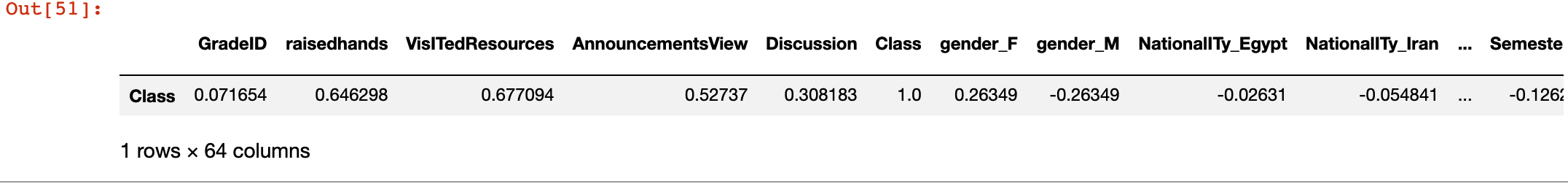
**4.2** **List the correlation between grades and other attributes**

1. corr = data.corr()
2. mask = np.triu(np.ones\_like(corr, dtype=bool))
3. f, ax = plt.subplots(figsize=(11, 9))
4. cmap = sns.diverging\_palette(230, 20, as\_cmap=True)
5. sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
6. square=True, linewidths=.5, cbar\_kws={"shrink": .5})



We look at Class in relation to the other columns individually:

1. corr = data.corr()
2. corr.iloc[[5]]



Based on the above graphs and tables, we can see that the number of visits to the course content, the number of days absent, the number of hands raised in class, the number of times new announcements were checked, whether or not discussions were held, gender, guardians, and semesters all have strong correlations with Class, as in our previous analysis.

**5. Training and Forecasting**

**5.1** **Find the most accurate classifier**

1. X = data.drop('Class', axis=1)
2. y = data['Class']
4. # Encoding our categorical columns in X
5. labelEncoder = LabelEncoder()
6. cat\_columns = X.dtypes.pipe(**lambda** x: x[x == 'object']).index
7. **for** col **in** cat\_columns:
8. X[col] = labelEncoder.fit\_transform(X[col])
10. # Train Test Split
11. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=52)
13. # Create the radial basis function kernel version of a Support Vector Machine classifier
14. rbf\_clf = svm.SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,
15. decision\_function\_shape='ovo', degree=3, gamma='auto', kernel='rbf',
16. max\_iter=-1, probability=False, random\_state=None, shrinking=True,
17. tol=0.001, verbose=False)
18. # Create the linear kernel version of a Support Vector Machine classifier
19. lin\_clf = svm.SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,
20. decision\_function\_shape='ovo', degree=3, gamma='auto', kernel='linear',
21. max\_iter=-1, probability=False, random\_state=None, shrinking=True,
22. tol=0.001, verbose=False)
23. # Create the polynomial kernel version of a Support Vector Machine classifier
24. poly\_clf = svm.SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,
25. decision\_function\_shape='ovo', degree=3, gamma='auto', kernel='poly',
26. max\_iter=-1, probability=False, random\_state=None, shrinking=True,
27. tol=0.001, verbose=False)
28. # Create the sigmoid kernel version of a Support Vector Machine classifier
29. sig\_clf = svm.SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,
30. decision\_function\_shape='ovo', degree=3, gamma='auto', kernel='sigmoid',
31. max\_iter=-1, probability=False, random\_state=None, shrinking=True,
32. tol=0.001, verbose=False)
33. keys = []
34. scores = []
35. models = {'Logistic Regression': LogisticRegression(max\_iter=3000), 'Decision Tree': DecisionTreeClassifier(),
36. 'Random Forest': RandomForestClassifier(n\_estimators=300, random\_state=52),'Perceptron':Perceptron(eta0=0.1, random\_state=15),'RBF':rbf\_clf,'Linear':lin\_clf,
37. 'Polynomial':poly\_clf,'Sigmoid':sig\_clf}
39. **for** k,v **in** models.items():
40. mod = v
41. mod.fit(X\_train, y\_train)
42. pred = mod.predict(X\_test)
43. **print**('Results for: ' + str(k) + '\n')
44. **print**(confusion\_matrix(y\_test, pred))
45. **print**(classification\_report(y\_test, pred))
46. acc = accuracy\_score(y\_test, pred)
47. **print**("accuracy is "+ str(acc))
48. **print**('\n' + '\n')
49. keys.append(k)
50. scores.append(acc)
51. table = pd.DataFrame({'model':keys, 'accuracy score':scores})
53. **print**(table)

Output:

1. Results **for**: Logistic Regression
3. [[28  7  1]
4. [ 4 43  9]
5. [ 0 12 40]]
6. precision    recall  f1-score   support
8. -1       0.88      0.78      0.82        36
9. 0       0.69      0.77      0.73        56
10. 1       0.80      0.77      0.78        52
12. accuracy                           0.77       144
13. macro avg       0.79      0.77      0.78       144
14. weighted avg       0.78      0.77      0.77       144
16. accuracy **is** 0.7708333333333334


20. Results **for**: Decision Tree
22. [[29  6  1]
23. [ 4 39 13]
24. [ 0 21 31]]
25. precision    recall  f1-score   support
27. -1       0.88      0.81      0.84        36
28. 0       0.59      0.70      0.64        56
29. 1       0.69      0.60      0.64        52
31. accuracy                           0.69       144
32. macro avg       0.72      0.70      0.71       144
33. weighted avg       0.70      0.69      0.69       144
35. accuracy **is** 0.6875


39. Results **for**: Random Forest
41. [[31  4  1]
42. [ 3 49  4]
43. [ 0 12 40]]
44. precision    recall  f1-score   support
46. -1       0.91      0.86      0.89        36
47. 0       0.75      0.88      0.81        56
48. 1       0.89      0.77      0.82        52
50. accuracy                           0.83       144
51. macro avg       0.85      0.84      0.84       144
52. weighted avg       0.84      0.83      0.83       144
54. accuracy **is** 0.8333333333333334


58. Results **for**: Perceptron
60. [[33  2  1]
61. [16 34  6]
62. [ 3 25 24]]
63. precision    recall  f1-score   support
65. -1       0.63      0.92      0.75        36
66. 0       0.56      0.61      0.58        56
67. 1       0.77      0.46      0.58        52
69. accuracy                           0.63       144
70. macro avg       0.66      0.66      0.64       144
71. weighted avg       0.65      0.63      0.62       144
73. accuracy **is** 0.6319444444444444


77. Results **for**: RBF
79. [[32  4  0]
80. [ 5 48  3]
81. [ 0 14 38]]
82. precision    recall  f1-score   support
84. -1       0.86      0.89      0.88        36
85. 0       0.73      0.86      0.79        56
86. 1       0.93      0.73      0.82        52
88. accuracy                           0.82       144
89. macro avg       0.84      0.83      0.83       144
90. weighted avg       0.83      0.82      0.82       144
92. accuracy **is** 0.8194444444444444


96. Results **for**: Linear
98. [[29  6  1]
99. [ 4 43  9]
100. [ 0 12 40]]
101. precision    recall  f1-score   support
103. -1       0.88      0.81      0.84        36
104. 0       0.70      0.77      0.74        56
105. 1       0.80      0.77      0.78        52
107. accuracy                           0.78       144
108. macro avg       0.79      0.78      0.79       144
109. weighted avg       0.78      0.78      0.78       144
111. accuracy **is** 0.7777777777777778


115. Results **for**: Polynomial
117. [[ 0 36  0]
118. [ 0 56  0]
119. [ 0 52  0]]
120. precision    recall  f1-score   support
122. -1       0.00      0.00      0.00        36
123. 0       0.39      1.00      0.56        56
124. 1       0.00      0.00      0.00        52
126. accuracy                           0.39       144
127. macro avg       0.13      0.33      0.19       144
128. weighted avg       0.15      0.39      0.22       144
130. accuracy **is** 0.3888888888888889

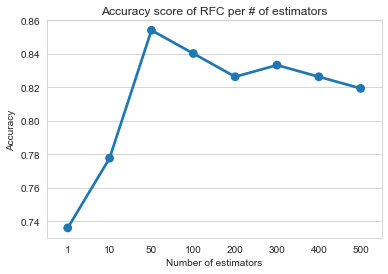

134. Results **for**: Sigmoid
136. [[32  4  0]
137. [ 6 46  4]
138. [ 1 18 33]]
139. precision    recall  f1-score   support
141. -1       0.82      0.89      0.85        36
142. 0       0.68      0.82      0.74        56
143. 1       0.89      0.63      0.74        52
145. accuracy                           0.77       144
146. macro avg       0.80      0.78      0.78       144
147. weighted avg       0.79      0.77      0.77       144
149. accuracy **is** 0.7708333333333334


153. model  accuracy score
154. 0  Logistic Regression        0.770833
155. 1        Decision Tree        0.687500
156. 2        Random Forest        0.833333
157. 3           Perceptron        0.631944
158. 4                  RBF        0.819444
159. 5               Linear        0.777778
160. 6           Polynomial        0.388889
161. 7              Sigmoid        0.770833

As can be seen in the table above, Random Forest is the most accurate classifier, with an accuracy of 83.3%. Let's further explore the number of estimators in the forest. As a general rule, when the number of estimators increases, the classifier performs better.

**5.2** **Exploratory Tonalities Random Forest Classifier**

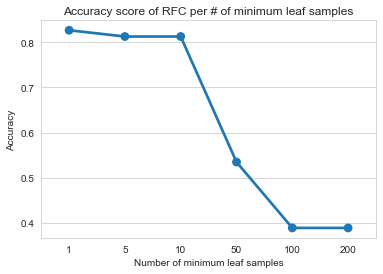
1. # Exploring the number of estimators in the random forest
2. score = []
3. est = []
4. estimators = [1, 10, 50, 100, 200, 300, 400, 500]
5. **for** e **in** estimators:
6. rfc1 = RandomForestClassifier(n\_estimators=e, random\_state=52)
7. pred1 = rfc1.fit(X\_train, y\_train).predict(X\_test)
8. accuracy = accuracy\_score(y\_test, pred1)
9. score.append(accuracy)
10. est.append(e)
11. plot = sns.pointplot(x=est, y=score)
12. plot.set(xlabel='Number of estimators', ylabel='Accuracy',
13. title='Accuracy score of RFC per # of estimators')
14. plt.show()



In fact, when the number of estimators increases, the RFC performs better. However, at 200 estimators, it tends to stabilize. Obviously, 200 estimators are sufficient for this dataset.

Another variable can be explored, such as the minimum number of samples required for a leaf node.

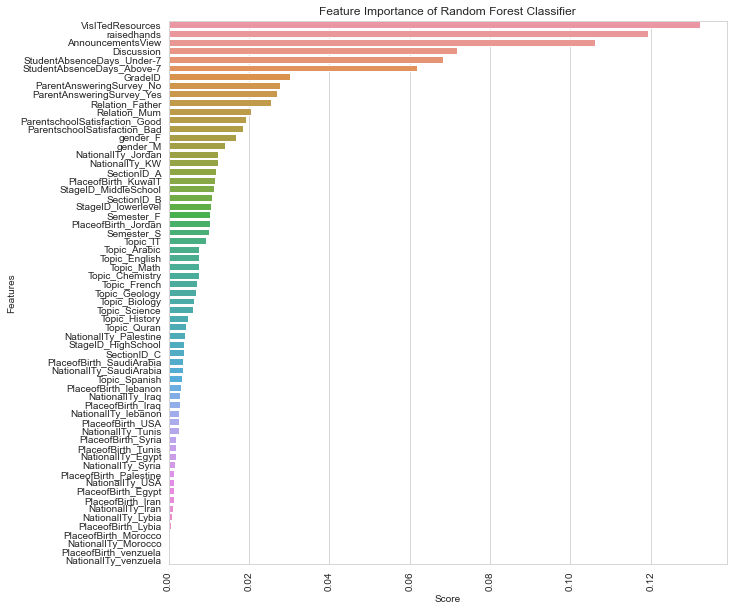
1. # Exploring minimum leaf samples
2. score = []
3. leaf = []
4. leaf\_options = [1, 5, 10, 50, 100, 200]
5. **for** l **in** leaf\_options:
6. rfc2 = RandomForestClassifier(n\_estimators=200, random\_state=52, min\_samples\_leaf=l)
7. pred2 = rfc2.fit(X\_train, y\_train).predict(X\_test)
8. accuracy = accuracy\_score(y\_test, pred2)
9. score.append(accuracy)
10. leaf.append(l)
11. plot = sns.pointplot(x=leaf, y=score)
12. plot.set(xlabel='Number of minimum leaf samples', ylabel='Accuracy',
13. title='Accuracy score of RFC per # of minimum leaf samples')
14. plt.show()



In this case, we can see that the accuracy fraction simply decreases as the minimum leaf sample increases. Therefore, it is better to keep this value at the default value of 1.

Let's evaluate the importance of RFC features.

1. rfc = RandomForestClassifier(n\_estimators=200, random\_state=52)
2. pred = rfc.fit(X\_train, y\_train).predict(X\_test)
3. dn = {'features':X.columns, 'score':rfc.feature\_importances\_}
4. df = pd.DataFrame.from\_dict(data=dn).sort\_values(by='score', ascending=False)
5. plot = sns.barplot(x='score', y='features', data=df, orient='h')
6. plot.set(xlabel='Score', ylabel='Features',
7. title='Feature Importance of Random Forest Classifier')
8. plt.rcParams['figure.figsize']=(20,20)
9. plt.setp(plot.get\_xticklabels(), rotation=90)
10. plt.show()



The number of visits to the course content is the most important feature.

**Conclusion:**

From our results, the number of visits to the course content, the number of days absent, the number of times students raised their hands in class, the number of times they checked for new announcements, whether or not they participated in discussions, gender, guardianship, and semester were indeed factors that influenced students' academic performance.