

Machine Learning Project Report 2019/20

STUDENT ACADEMIC PERFORMANCE CLASSIFICATION

Zeeshan Hussain : Khand (1880109) | Elaheh
Mohammandnezhadhardouroudi (1919827) | Venus Kiani (1916304)
|

Contents

Introduction.....	3
Dataset Information	4
Adaptive Learning	6
Process to be Followed	7
Tools and Libraries	8
Python 3.7	8
Numpy	8
Pandas	8
Seaborn Library	8
Graphviz.....	8
Scikit Learn.....	9
Data Preprocessing.....	9
1. Reading Data into Kaggle Kernel	9
2. Load All Libraries required to process the data	9
3. Load the DataSet.....	10
4. Check missing values:.....	10
5. Check for Duplicates.....	11
6. Check the type of features and label	12
7. Rename Columns	13
8. Distribution of Class Label	13
9. Using Label encoding to convert categorical features into numerical ones.....	14
10. Train and Test Split.....	14
Exploratory Data Analysis.....	15
1. Student Raised Hands in a Certain Time	15
2. Student Visited Resources in a Certain time.....	15
3. Relation Between Gender and Performance	16
4. Participation by Country	18
5. Visualizing Categorical Features with Numerical Features	19
6. Semester Wise Performance	20
Goal of the Project	20
Decision Tree Algorithm	21
Advantages of Decision Tree.....	21

Process of Decision Tree	21
Making the Decision Tree Model	22
1. Importing Required Libraries.....	22
2. Feature Selection.....	22
3. Splitting the Data into Train and Test Set.....	22
4. Building Decision Tree Model.....	22
5. Evaluating Model.....	23
6. Visualizing the Decision Tree.....	23
Optimizing Decision Tree Performance (HyperParameter tuning).....	23
Naïve Bayes Algorithm Learning	24
Creating a Naïve Bayes Classifier	24
Random Forest Classifier.....	25
How the Algorithm Works ?.....	26
Generating a Random Forest Classifier Model.....	26
Feature Importance using Random Forest Classifier	27
Conclusions and Future Work.....	28
Results.....	29
References.....	29
Figure 1: Kaggle Page from where we downloaded xAPI-Edu Dataset.....	5
Figure 2: Adaptive Learning System.....	6
Figure 3: Student Raised Hands in a Certain Time	15
Figure 4: Number of Times Student Visited Resource in Certain Time	16
Figure 5: Relation between gender and performance.....	16
Figure 6 : Gender Wise Participation Visualizations	17
Figure 7: Participation Country Wise	18
Figure 8: Visualizing Categorical Features with numerical features.....	19
Figure 9: Semester Wise Performance	20
Figure 10: Features Importance.....	28

Introduction

Adaptive learning is adapting. The adaptive learning approach, long recognized for its ability to deliver a more personalized version of learning, has a 60-year history in classrooms and corporations around the world. As technology has advanced over the decades, so has adaptive learning – becoming more refined and effective due to leaps in cloud-based managed services, computing power, scalability, and machine learning.

Today, armed with great advancements in data science, artificial intelligence and machine learning, adaptive learning is on the cutting edge of a new, dramatic change.

Adaptive Learning 3.0 is characterized by the application of AI and machine learning to replicate the one-on-one instructor experience more accurately. AI-powered adaptive solutions leverage network knowledge maps to create knowledge and behavioral nodes, forming deeper relationships between content, learning objectives, and persona types, to name a few. This powers a more efficient, effective learning experience and enables:

- Complex, real-time adaptations based on learner performance and behavior
- Data-driven, personalized hints, feedback, remediation, and knowledge reinforcement
- Predictive, forgetting curves and insights into future knowledge application
- Comprehensive application-level mastery of skills and knowledge
- Reduction in learning times

It is not just the learning experience that is amplified. The platforms that are fully embracing AI and machine learning are also able to provide dramatic efficiencies to learning development and content creation. For example, AI-powered platforms can assess the performance of learning content, flag underperforming content and even hide underperforming assessments until they are revised.

The objective of our project is to develop a machine learning model that can predict the grade the Level of a student with a good accuracy so as to assist the system into analyzing what factors contribute to the grade of the student and to predict the level of a student so as to help student improve his/her grade in an Adaptive e-Learning Environment such as KalBoard 360 whose dataset we use for our task.

Educational Machine Learning concerns of developing methods to discover hidden patterns from educational data. We collect the data from an e-Learning system called Kalboard 360 using Experience API Web service (XAPI). After that, we use some supervised machine learning classification techniques such as Decision Tree, Naïve Bayes, and Random Forest classifiers to evaluate the impact of such features on student's academic performance. The results reveal that there is a strong relationship between learner behaviors and its academic achievement. Furthermore, our new model that will be developed from scratch is capable to be used with real time data to perform classification of student performance and make predictions.

Dataset Information

This Dataset of student academic performances is collected from Learning Management System called Kalboard 360. Such systems provide users access to cutting edge technology and education resources for any device with Internet Connection. The data is collected using a learner activity tracker tool, which called experience API (xAPI). The xAPI is a component of the training and learning architecture (TLA) that enables to monitor learning progress and learner's actions like reading an article or watching a training video. The experience API helps the learning activity providers to determine the learner, activity and objects that describe a learning experience.

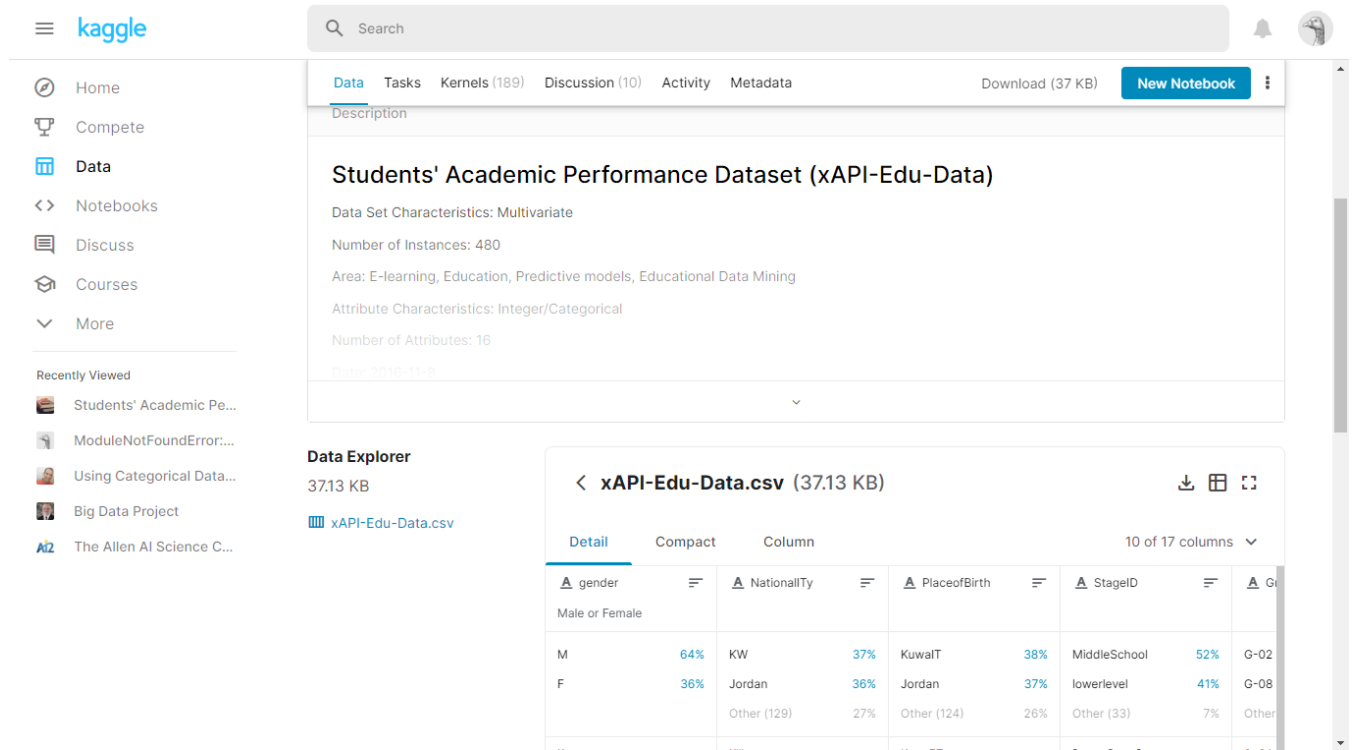
The dataset consists of 480 records and 18 features. The features are classified into three major categories :

- 1) Demographic features – Gender and Nationality
- 2) Academic Background- Educational stage, Grade level and section
- 3) Behavioral features- Raised hands in class, opening resources ,answering survey by parents, school satisfaction.

The dataset consists of analysis on the 305 males and 175 females. The countries include Kuwait, Jordan, Palestine, Iraq , Lebanon, Tunis, Saudi Arabia, Morocco, Egypt , Syria , USA, Iran and Libya where the software was used. The data set also includes the school attendance feature such as the students are classified into two categories based on their absence days: 191 students exceed 7 absence days and 289 students their absence days under 7.

The students are classified into three numerical intervals based on their total grade/mark:

- Low-Level: interval includes values from 0 to 69,
- Middle-Level: interval includes values from 70 to 89,
- High-Level: interval includes values from 90-100.



Kaggle

Search

[Data](#) [Tasks](#) [Kernels \(189\)](#) [Discussion \(10\)](#) [Activity](#) [Metadata](#) [Download \(37 KB\)](#) [New Notebook](#)

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

Description

Data Set Characteristics: Multivariate

Number of Instances: 480

Area: E-learning, Education, Predictive models, Educational Data Mining

Attribute Characteristics: Integer/Categorical

Number of Attributes: 16

Date: 2016-11-8

Data Explorer

37.13 KB

[xAPI-Edu-Data.csv](#)

xAPI-Edu-Data.csv (37.13 KB)

[Detail](#) [Compact](#) [Column](#) 10 of 17 columns

gender	Nationality	PlaceofBirth	StageID	Grade
Male or Female				
M 64%	KW 37%	KuwaIT 38%	MiddleSchool 52%	G-02
F 36%	Jordan 36%	Jordan 37%	lowerlevel 41%	G-08
	Other (129) 27%	Other (124) 26%	Other (33) 7%	Other

Figure 1: Kaggle Page from where we downloaded xAPI-Edu Dataset

Adaptive Learning

Adaptive learning technology is any system that collects information on a student's skill, knowledge and confidence levels, and uses it to change the material or tasks presented to that student. The goal of adaptive learning technology is to personalize the course to suit the individual's needs, by helping them improve in their areas of weakness, better retain what they have learned and progress through the course at a pace and in a direction uniquely suited to them.

Adaptive learning falls under the umbrella of what are known as Intelligent Tutor Systems (ITS). These represent the most prominent application of deep learning in EdTech. They work by tracking the mental steps of learners during problem-solving to analyse their understanding of a field. Through their emulation of human behavior to gain insight, they provide guidance, feedback and explanations to the learner in real-time, and more interestingly, they can recommend learning activities specifically suited to the individual learner.

Adaptive learning is an educational method which uses computers as interactive teaching devices, and to orchestrate the allocation of human and mediated resources according to the unique needs of each learner. Computers adapt the presentation of educational material according to students' learning needs, as indicated by their responses to questions, tasks and experiences

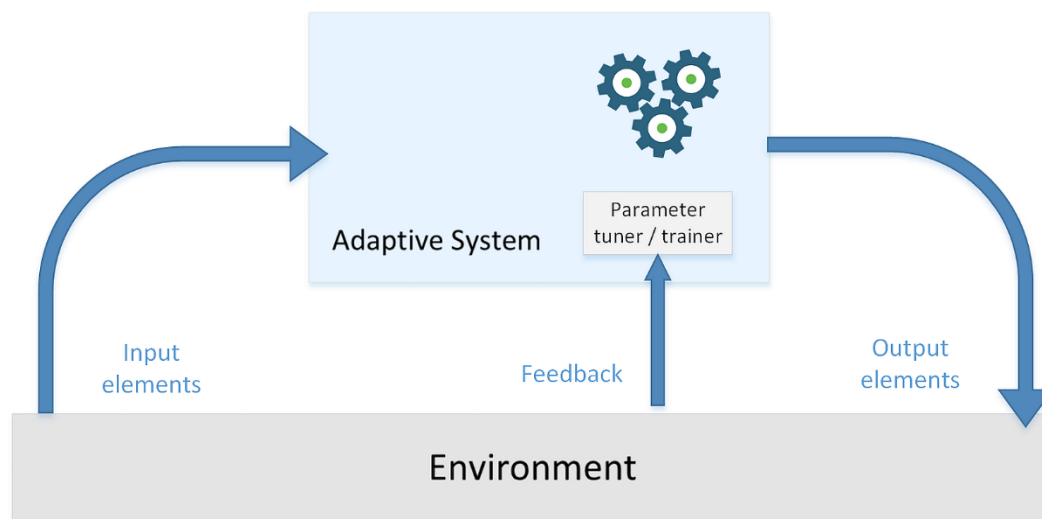


Figure 2: Adaptive Learning System

Process to be Followed

From dataset above, we can conclude it is a supervised classification learning task with 3 labels (Low-level , Middle-Level and High-Level) to be predicted for the student. So , we choose 3 algorithms for this task :

1. Decision Tree
2. Naïve Bayes
3. Random Forest Classifier

The machine learning process to followed is as below :

1. **Data Pre-processing** : We load the data , transform it into a state from which our machine learning algorithm can process it
2. **Exploratory Data Analysis on Data** : exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task. **Exploratory Data Analysis (EDA)** helps to answer all these questions, ensuring the best outcomes for the project. It is an approach for summarizing, visualizing, and becoming intimately familiar with the important characteristics of a data set.
3. **Choosing the Model (Decision Tree, Naïve Bayes, Random Forest Classifier)** : since it is a supervised classification task with multivariate classes to predicted, we test three algorithms that best fit the task.
4. **Training the model** : train the model to predict the grade/level of a student. As specified , we will develop three models and compare their results to select which one is best. Each model will be trained on the same split of test and train data so as to compare their performance.
5. **Evaluation** : Once training is complete, it's time to see if the model is any good, using Evaluation. This is where that dataset that we set aside earlier comes into play. Evaluation allows us to test our model against data that has never been used for training. This metric allows us to see how the model might perform against data that it has not yet seen. This is meant to be representative of how the model might perform in the real world. We use accuracy as our metric for the test data to measure the performance of the models trained.

6. **Hyperparameter tuning** : we will tune parameters such as max_depth , criterion in decision trees to how our model performs with changes in the parameters and if it improves the accuracy. We will do parameter tuning for all models used.
7. **Prediction on test data** : we will use the model on the test data and check its accuracy on the test data to make sure if it predicts the right results.
8. **Calculate feature importance** : Inspecting the **importance** score provides insight into that specific model and which **features** are the most **important** and least **important** to the model when making a prediction.

Tools and Libraries

I have used several libraries to achieve the task. Following is the description of all the libraries and tools I have used in the project.

Python 3.7

This is latest version of python which is embedded in Anaconda navigator. Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. Package versions are managed by the package management system conda. Python is an interpreted, high-level, general-purpose programming language.

Numpy

I have used numpy to manage my dataset and to calculate metrics for evaluation. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

Pandas

Pandas is a library that is used to read and write the data from .csv , excel files and do a whole lot of manipulation. We use it in our project to read data from .csv file and do certain manipulation on the training data.

Seaborn Library

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. It is built on top of matplotlib and closely integrated with pandas data structures. We use it in our project for performing Exploratory Data Analysis on our dataset to get new insights from Data and make conclusions as to what factors contribute to the grade of the student. We use seaborn to draw different graphical visualizations of the data.

Graphviz

This is a library that I used in my project to visualize my decision tree. This provides a clear graphical representation of the decision tree that is generated after fitting our model on training data.

Scikit Learn

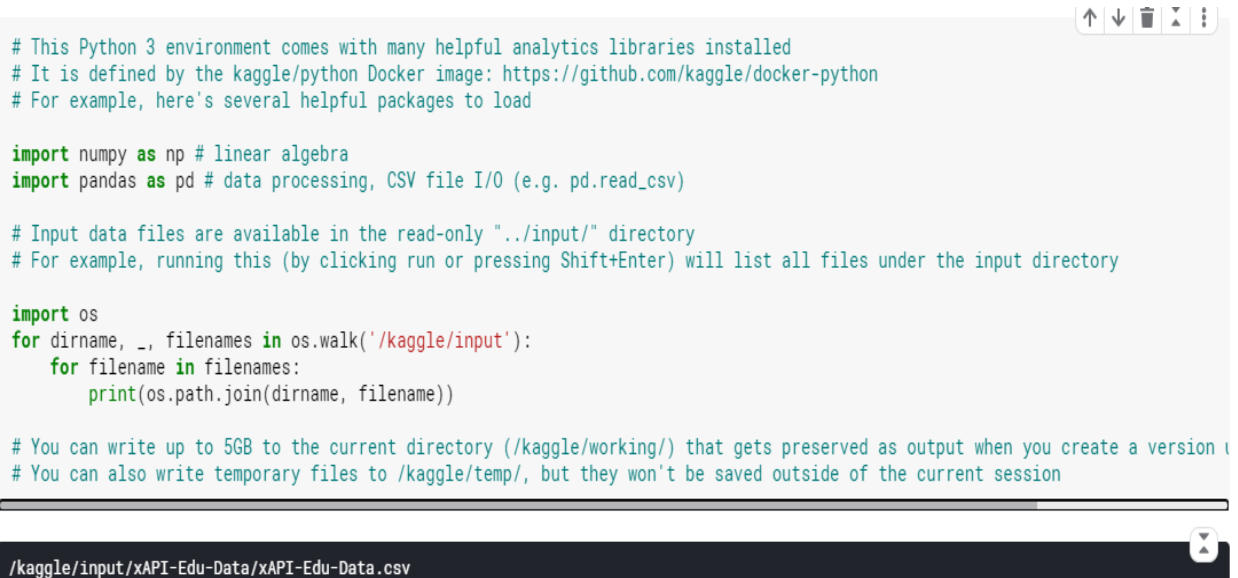
Scikit Learn is a python library that provides us support for implementation of the machine learning algorithms. It is an open source machine learning library for the Python programming language. It provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. For our project purpose, we have Scikit Learn to implement the decision tree, naïve bayes and random forest classifier on our dataset.

Data Preprocessing

I use a Kaggle Notebook to do the project coding

1. Reading Data into Kaggle Kernel

I use a Kaggle Kernel Notebook to run and execute my project. Therefore , the first step is to read the data and assign it to this project directory on Kaggle. The following is the illustration of code to do this.



```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 5GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

/kaggle/input/xAPI-Edu-Data/xAPI-Edu-Data.csv

2. Load All Libraries required to process the data

We use numpy, pandas, seaborn and sklearn in our project. In this step I import them.

```
[2]: import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
sns.set_style('whitegrid')

from sklearn.impute import SimpleImputer as Imputer
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
```

3. Load the DataSet

We then load our dataset as a dataframe object to do processing on the data. We use pandas for this purpose. As we can see the data is now available in the kernel. The shape of our data is (480,17) which means it has 480 rows(aka values) and 17 features/columns.

```
[3]: df = pd.read_csv("/kaggle/input/xAPI-Edu-Data/xAPI-Edu-Data.csv") #at first we import the dataset
```

```
df.head()
```

Out[4]:

	gender	Nationality	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation	raisedhands	VisiTedResources	AnnouncementsView	Discussion	ParentAn:
0	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	15	16	2	20	
1	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	20	20	3	25	
2	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	10	7	0	30	
3	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	30	25	5	35	
4	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	40	50	12	50	

```
[5]: print (df.shape)
```

```
(480, 17)
```

4. Check missing values:

The first thing to check is if our data has any missing values. So we run the following command. In data preprocessing, one of the major task is to check if the data has any missing values and to handle them before processing the data.

Our dataset doesnot have any missing value. Next we check for duplicates.

```
[6]: df.isnull().sum() #so we can consider that we don't have null values, so we can go fast through dataset
```

```
Out[6]: 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
```

5. Check for Duplicates

```
In [16]: # drop any duplicate value if exists
df.drop_duplicates()
```

```
Out[16]:
```

There are no duplicate values too in the dataset.

6. Check the type of features and label

```
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
```

We see that we have 13 categorical features and 4 numerical. Before we process the categorical features by encoding them. We first rename the mis-spelled columns.

7. Rename Columns

```
df.columns

Out[9]: Index(['gender', 'NationalITy', 'PlaceofBirth', 'StageID', 'GradeID',
              'SectionID', 'Topic', 'Semester', 'Relation', 'raisedhands',
              'VisITedResources', 'AnnouncementsView', 'Discussion',
              'ParentAnsweringSurvey', 'ParentschoolSatisfaction',
              'StudentAbsenceDays', 'Class'],
              dtype='object')

[10]: df.rename(index=str, columns={'gender':'Gender',
                                   'NationalITy':'Nationality',
                                   'raisedhands':'RaisedHands',
                                   'VisITedResources':'VisitedResources'},
               inplace=True)

df.columns #here we want to make the dataset neat as some words had capital and small alphabets together

Out[10]: Index(['Gender', 'Nationality', 'PlaceofBirth', 'StageID', 'GradeID',
               'SectionID', 'Topic', 'Semester', 'Relation', 'RaisedHands',
               'VisitedResources', 'AnnouncementsView', 'Discussion',
               'ParentAnsweringSurvey', 'ParentschoolSatisfaction',
               'StudentAbsenceDays', 'Class'],
               dtype='object')
```

Now the columns are renamed and the spelling mistakes have been corrected.

Now lets see the distribution of the class label.

8. Distribution of Class Label

```
In [14]: df.groupby('Class').size()

Out[14]:
Class
H      142
L      127
M      211
dtype: int64
```

In our data, we have 142 students who scored High Score, 127 students who scored low score and 211 students were in Medium Category. The data doesnt look imbalanced, therefore we can say that test accuracy could be a good metric to evaluate performance on the dataset.

9. Using Label encoding to convert categorical features into numerical ones

```
In [34]:
replace_map = {'Gender': {'F': 1, 'M': 2}}
replace_map1 = {'Nationality': {'Egypt': 1, 'Iran': 2, 'Iraq': 3, 'Jordan': 4, 'KW': 5, 'Lybia': 6, 'Morocco': 7, 'Palestine': 8, 'SaudiArabia': 9, 'Syria': 10, 'Tunis': 11, 'USA': 12, 'lebanon': 13, 'venzuela': 14}}
replace_map2 = {'PlaceofBirth': {'Egypt': 1, 'Iran': 2, 'Iraq': 3, 'Jordan': 4, 'KuwaIT': 5, 'Lybia': 6, 'Morocco': 7, 'Palestine': 8, 'SaudiArabia': 9, 'Syria': 10, 'Tunis': 11, 'USA': 12, 'lebanon': 13, 'venzuela': 14}}
replace_map3 = {'StageID': {'HighSchool': 1, 'MiddleSchool': 2, 'lowerlevel': 3}}
replace_map4 = {'GradeID': {'G-02': 1, 'G-04': 2, 'G-05': 3, 'G-06': 4, 'G-07': 5, 'G-08': 6, 'G-09': 7, 'G-10': 8, 'G-11': 9, 'G-12': 10}}
replace_map5 = {'SectionID': {'A': 1, 'B': 2, 'C': 3}}
replace_map6 = {'Topic': {'Arabic': 1, 'Biology': 2, 'Chemistry': 3, 'English': 4, 'French': 5, 'Geology': 6, 'History': 7, 'IT': 8, 'Math': 9, 'Quran': 10, 'Science': 11, 'Spanish': 12}}
replace_map7 = {'Semester': {'F': 1, 'S': 2}}
replace_map8 = {'Relation': {'Father': 1, 'Mum': 2}}
replace_map9 = {'ParentAnsweringSurvey': {'Yes': 1, 'No': 2}}
replace_map10 = {'ParentschoolSatisfaction': {'Bad': 1, 'Good': 2}}
replace_map11 = {'StudentAbsenceDays': {'Above-7': 1, 'Under-7': 2}}
replace_map12 = {'Class': {'M': 1, 'L': 2, 'H': 3}}
```

```
In [35]:
df_dt.replace(replace_map,inplace=True)
df_dt.replace(replace_map1,inplace=True)
df_dt.replace(replace_map2,inplace=True)
df_dt.replace(replace_map3,inplace=True)
df_dt.replace(replace_map4,inplace=True)
df_dt.replace(replace_map5,inplace=True)
df_dt.replace(replace_map6,inplace=True)
df_dt.replace(replace_map7,inplace=True)
df_dt.replace(replace_map8,inplace=True)
df_dt.replace(replace_map9,inplace=True)
df_dt.replace(replace_map10,inplace=True)
df_dt.replace(replace_map11,inplace=True)
df_dt.replace(replace_map12,inplace=True)
```

10. Train and Test Split

In Machine Learning model it is necessary to split our dataset into two separate sets, Training set and Test set. We do so because from the training set our model will learn and from test set it will classify

our model. In general it is good practice to split dataset into 80-20 ratio where 80 percent dataset for training and 20 percent for testing.

```
124]: X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.2,random_state=1)
      indextoCheckDecisionTree=y_test.index
```

We will use the same split in all our algorithms to better evaluate the performance of algorithm on the same proportion of the test data.

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analyzing datasets to summarize their main characteristics, often with visual methods. EDA is used for seeing what the data can tell us before the modeling task. It is not easy to look at a column of numbers or a whole spreadsheet and determine important characteristics of the data. It may be tedious, boring, and/or overwhelming to derive insights by looking at plain numbers. Exploratory data analysis techniques have been devised as an aid in this situation. In other words, we use EDA to get helpful insights about our data. We use SeaBorn and matplotlib for the visualizations.

1. Student Raised Hands in a Certain Time

By this graph we want to visualize the participation of the students during learning.

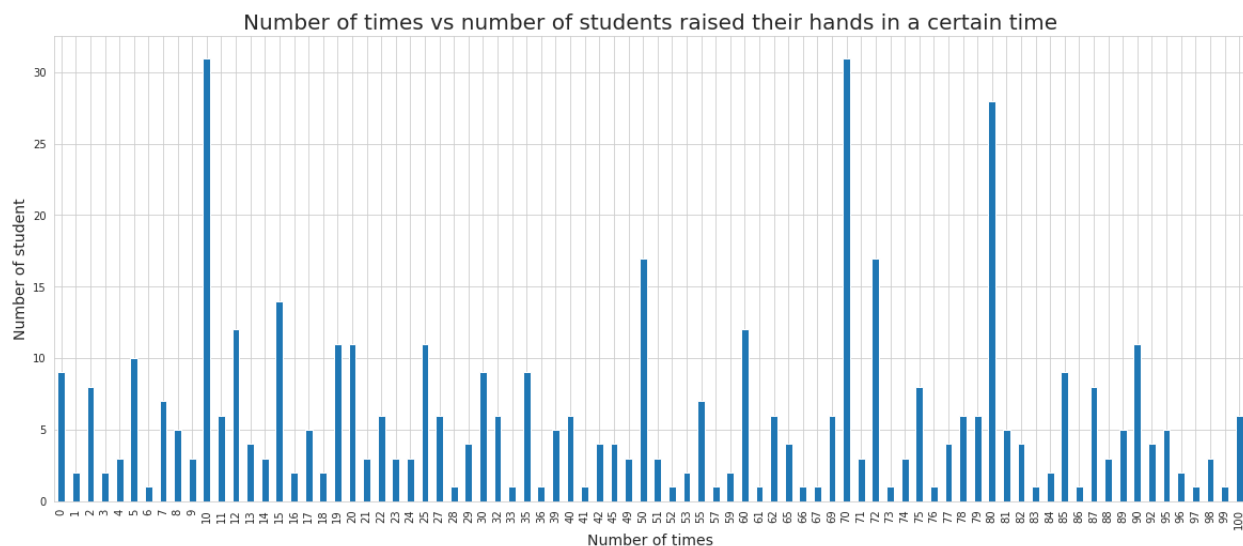


Figure 3: Student Raised Hands in a Certain Time

We see as more students are in a learning environment , there is more participation in the class.

2. Student Visited Resources in a Certain time

By this we want to visualize how many times student visited Resource in a certain time

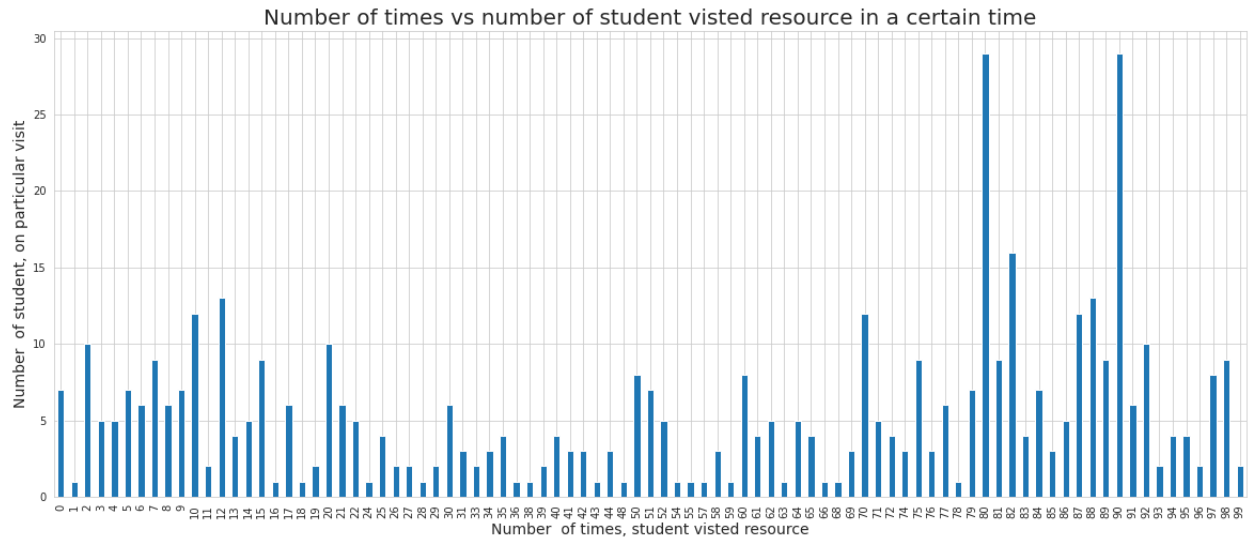


Figure 4: Number of Times Student Visited Resource in Certain Time

We see that there is a direct correlation between the number of students in class and number of visits to a resource.

By this , we can confirm that **participation in a class increases if there are more number of students.**

3. Relation Between Gender and Performance

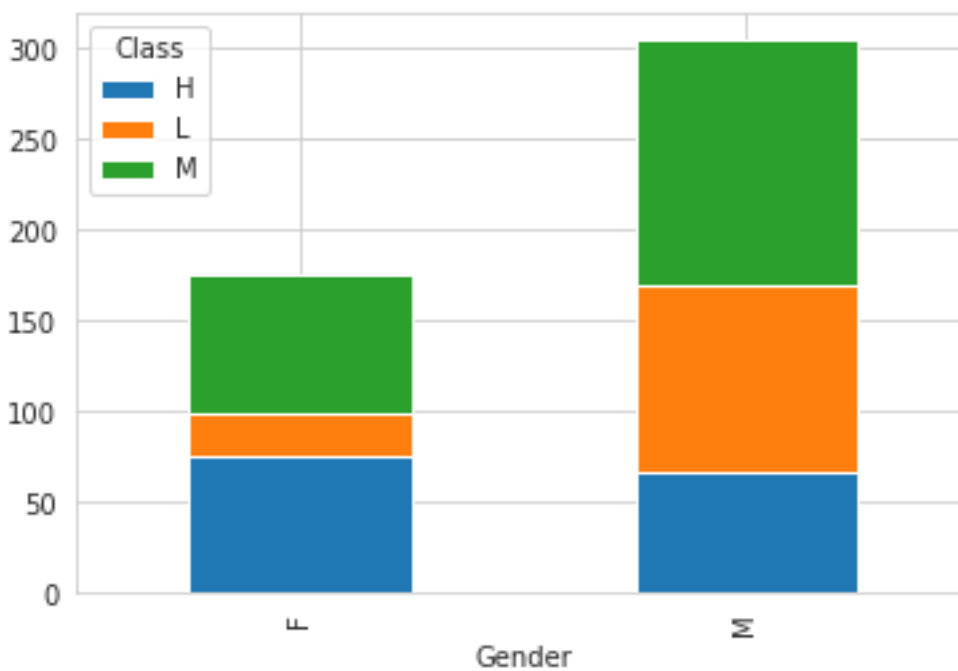


Figure 5: Relation between gender and performance

We see from our data that very less females score low than Male. From this bar chart we visualize that male students are more on "medium" and "lower" category while girls show better performance in the system. Since the data is from the middle-east schools, the factors contributing to this trend are unknown to us but this is interesting find.

Lets visualize it by subject and then country. If we look at gender wise performance by each subject , we see that the performance for male students is good in IT , Arabic and Spanish while in Science Subjects such as Chemistry, Biology performance is nearly equal.

Our visualization also shows us there is a very high gender disparity among students in middle east countries. That might be one factor affecting the performance of students.

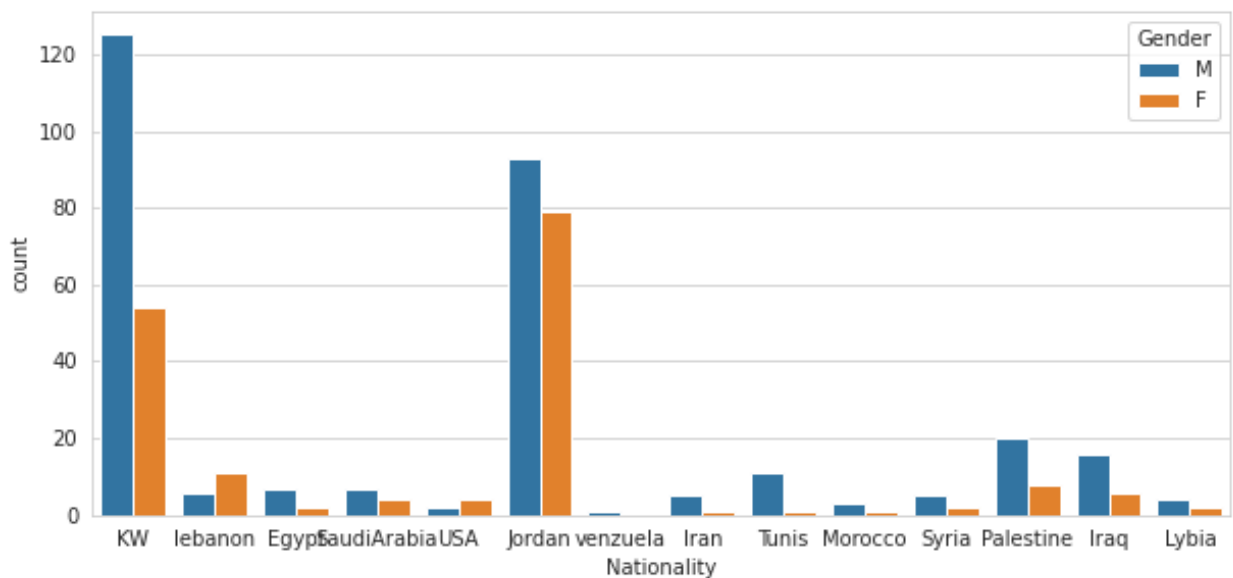
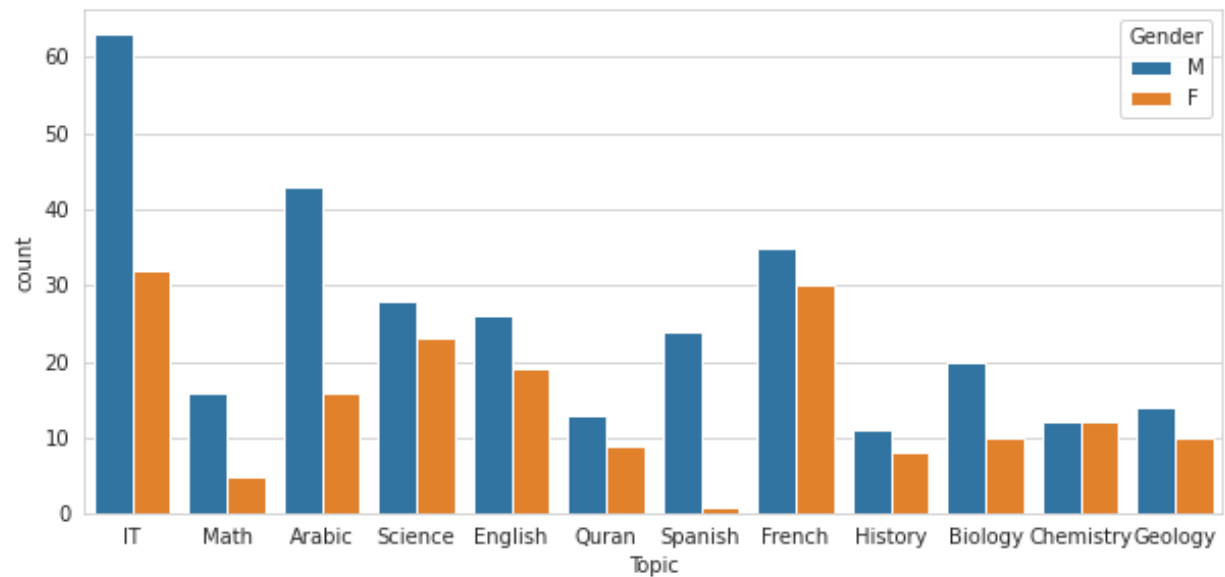


Figure 6 : Gender Wise Pariticipation Visualizations

4. Participation by Country

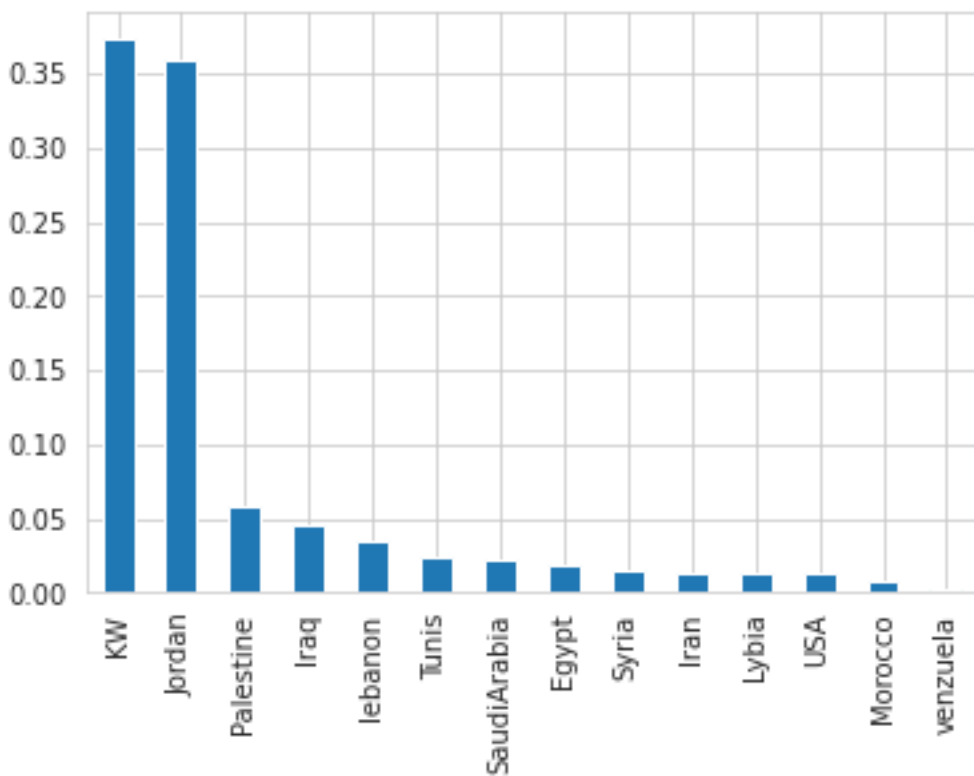


Figure 7: Participation Country Wise

We see that Kuwait and Jordan have the highest number of students registered in the system.

5. Visualizing Categorical Features with Numerical Features

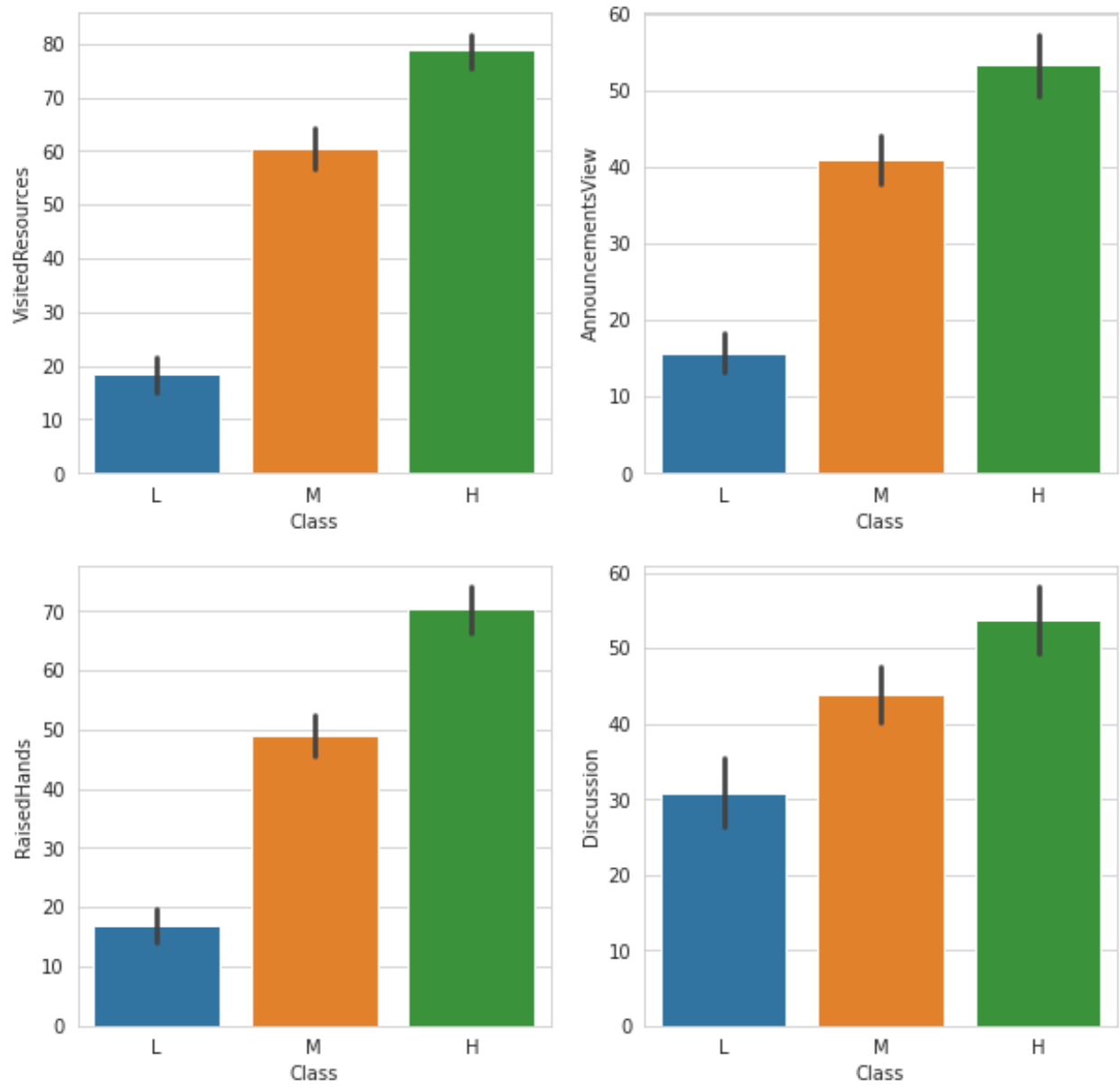


Figure 8: Visualizing Categorical Features with numerical features

As expected, those that participated more (higher counts in Discussion, raisedhands, AnnouncementsViews, RaisedHands), performed better in the class. We will see later that these findings will be confirmed by our model or not.

6. Semester Wise Performance

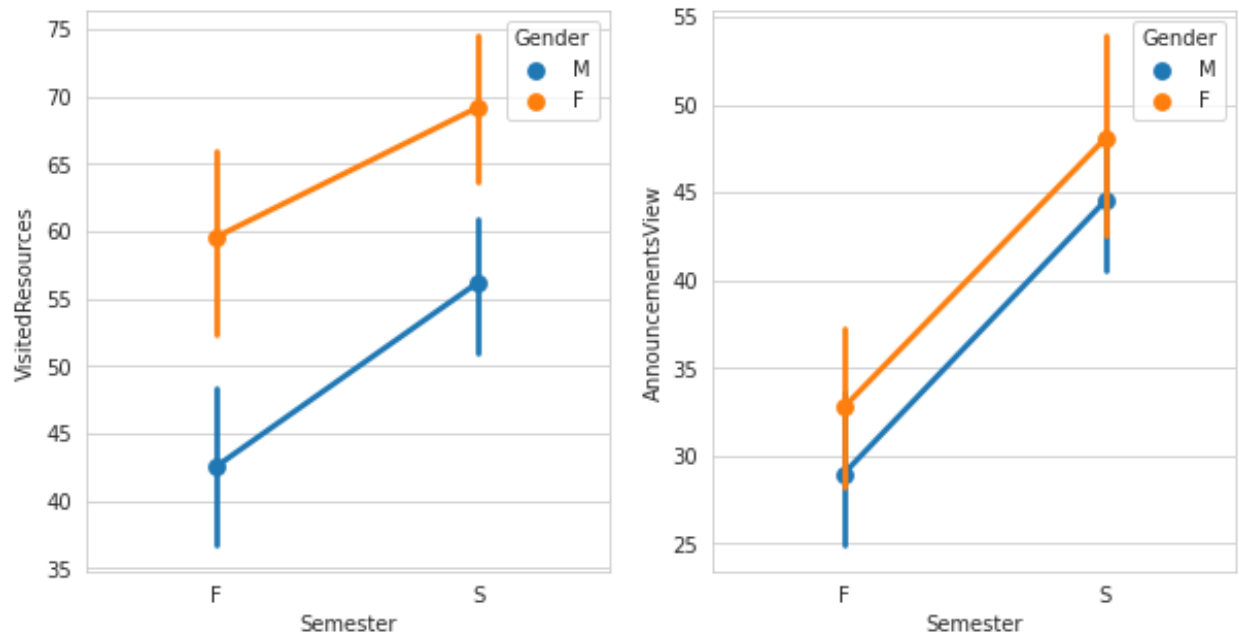


Figure 9: Semester Wise Performance

We see from above that students were more vigilant during the 2nd semester than in the first one (i.e. visited more resources and view more announcements).

Goal of the Project

Our Goal is to predict the Class label which is the level of the student as predicted by the model. The level of student is divided into three categories by their score/grade in the semester as following :

- Low-Level: interval includes values from 0 to 69,
- Middle-Level: interval includes values from 70 to 89,
- High-Level: interval includes values from 90-100.

We conclude it is a supervised classification task where we need to predict the categorical label. So we use the following three algorithms to test on the dataset and make 3 models.

1. Decision Trees
2. Naive Bayes
3. Random Forest Classifier

Finally , we will conclude with the model which has the most accuracy on the test data.

Decision Tree Algorithm

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.

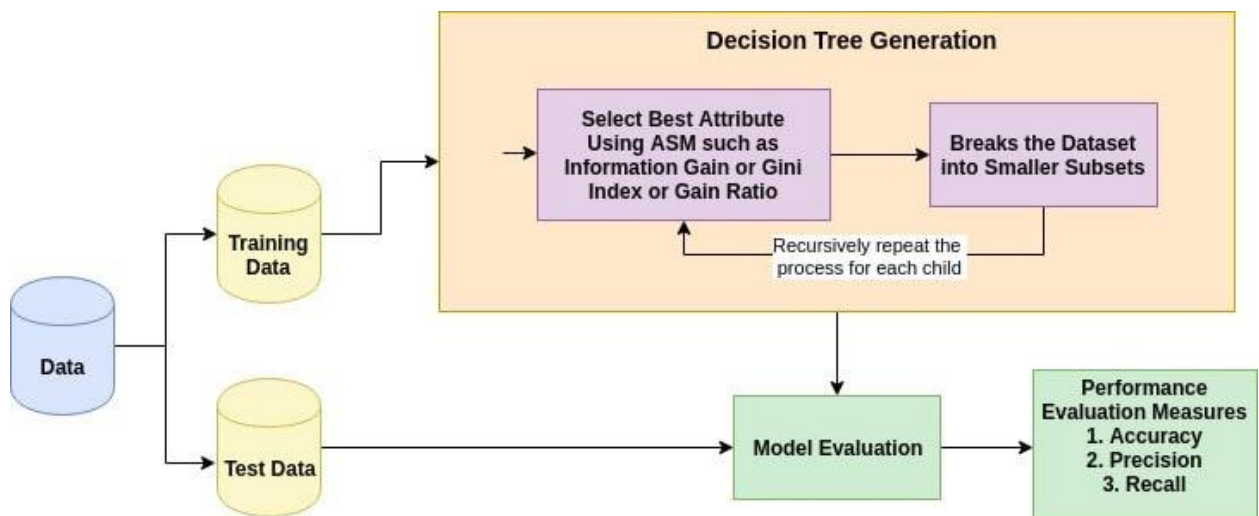
Advantages of Decision Tree

1. Decision Tree has a fast training time
2. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions.
3. Decision trees can handle high dimensional data with good accuracy

Process of Decision Tree

The basic idea behind any decision tree algorithm is as follows:

1. Select the best attribute using Attribute Selection Measures(ASM) to split the records.
2. Make that attribute a decision node and breaks the dataset into smaller subsets.
3. Starts tree building by repeating this process recursively for each child until one of the condition will match:
 - All the tuples belong to the same attribute value.
 - There are no more remaining attributes.
 - There are no more instances.



Making the Decision Tree Model

1. Importing Required Libraries

Let's first load the required libraries from sklearn.

```
[298]: # Great now lets start decision tree learning

from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

2. Feature Selection

Here, you need to divide given columns into two types of variables dependent(or target variable) which in our case is 'Class' and independent variable(or feature variables) such as Gender, Nationality e.t.c

```
[ ]: feature_columns=['Gender', 'Nationality', 'PlaceofBirth', 'StageID', 'SectionID', 'Topic', 'Semester', 'Relation', 'RaisedHands', 'VisitedReso']
X=df_dt[feature_columns]
y=df_dt.Class
```

3. Splitting the Data into Train and Test Set

To understand model performance, dividing the dataset into a training set and a test set is a good strategy. Let's split the dataset by using function `train_test_split()`. You need to pass 3 parameters features, target, and testset size.

```
[300]: X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.2,random_state=1)
indextoCheckDecisionTree=y_test.index
```

4. Building Decision Tree Model

Let's create a Decision Tree Model using Scikit-learn.

```
[301]: clf=DecisionTreeClassifier()
```

+ Code

+ Markdown

```
[302]: clf=clf.fit(X_train,y_train)
```

```
y_pred = clf.predict(X_test)
```

5. Evaluating Model

Accuracy can be computed by comparing actual test set values and predicted values.

```
[304]: # now lets test accuracy of our model
Accuracy_dt=metrics.accuracy_score(y_test,y_pred)
print("Accuracy : ",Accuracy_dt)
```

```
Accuracy : 0.6979166666666666
```

The resultant accuracy is 69.79 % .

6. Visualizing the Decision Tree

The resulting Decision Tree can be viewed at : <https://www.kaggle.com/zeeshankhand/machine-learning-course-project#Decision-Tree-Learning-on-Data>

We use the Graphviz library to generate the resultant tree visualization in the notebook.

Optimizing Decision Tree Performance (HyperParameter tuning)

We update the following hyper-parameters in our classifier and again check the accuracy of the data.

- **Criterion** : This parameter allows us to use the different-different attribute selection measure. We use “entropy” for information gain instead of the default “gini”
- **max_depth : int or None, optional (default=None) or Maximum Depth of a Tree**: The maximum depth of the tree. If None, then nodes are expanded until all the leaves contain less than min_samples_split samples. The higher value of maximum depth causes overfitting, and a lower value causes underfitting. We set our max_depth to 13.

Now lets check the accuracy of the model and see if its improved or not. The accuracy is now 75 %.

```
[311]: # The accuracy decreased by a certain margin. so entropy is not a good criterion. Now lets tune parameter of max_depth
clf=DecisionTreeClassifier(criterion="entropy",max_depth=13)
clf=clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
Accuracy_dt=metrics.accuracy_score(y_test,y_pred)
print("Accuracy : ",Accuracy_dt)
```

```
Accuracy : 0.75
```


The resultant decision tree can be viewed in the notebook at :

<https://www.kaggle.com/zeeshankhand/machine-learning-course-project#Decision-Tree-Learning-on-Data>

Naïve Bayes Algorithm Learning

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm.

Naive Bayes classifiers have high accuracy and speed on large datasets.

Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h)$: the probability of hypothesis h being true (regardless of the data). This is known as the prior probability of h .
- $P(D)$: the probability of the data (regardless of the hypothesis). This is known as the prior probability.
- $P(h|D)$: the probability of hypothesis h given the data D . This is known as posterior probability.
- $P(D|h)$: the probability of data d given that the hypothesis h was true. This is known as posterior probability.

Creating a Naïve Bayes Classifier

We will use the same train and test split (80-20). And also use the same encoded features.

```
[317]: from sklearn.naive_bayes import GaussianNB
```

```
[321]: gnb = GaussianNB()
```

```
[322]: gnb.fit(X_train, y_train)
```

Out[322] GaussianNB()

```
y_pred = gnb.predict(X_test)
```

+ Code

+ Markdown

```
[328]: Accuracy_naive=metrics.accuracy_score(y_test,y_pred)
print("Accuracy : ",Accuracy_naive)
```

Accuracy : 0.7708333333333334

```
[329]: # To conclude on the same split , naive bayes has a better accuracy.
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
print("Confusion Metrix",metrics.confusion_matrix(y_test,y_pred))
cnf=metrics.confusion_matrix(y_test,y_pred)
print("Accuracy",metrics.accuracy_score(y_test,y_pred))
```

Confusion Metrix [[23 6 7]
[1 18 0]
[7 1 33]]
Accuracy 0.7708333333333334

We have 77.88 % accuracy on the test data for prediction of the class which is better and improved than decision trees.

Random Forest Classifier

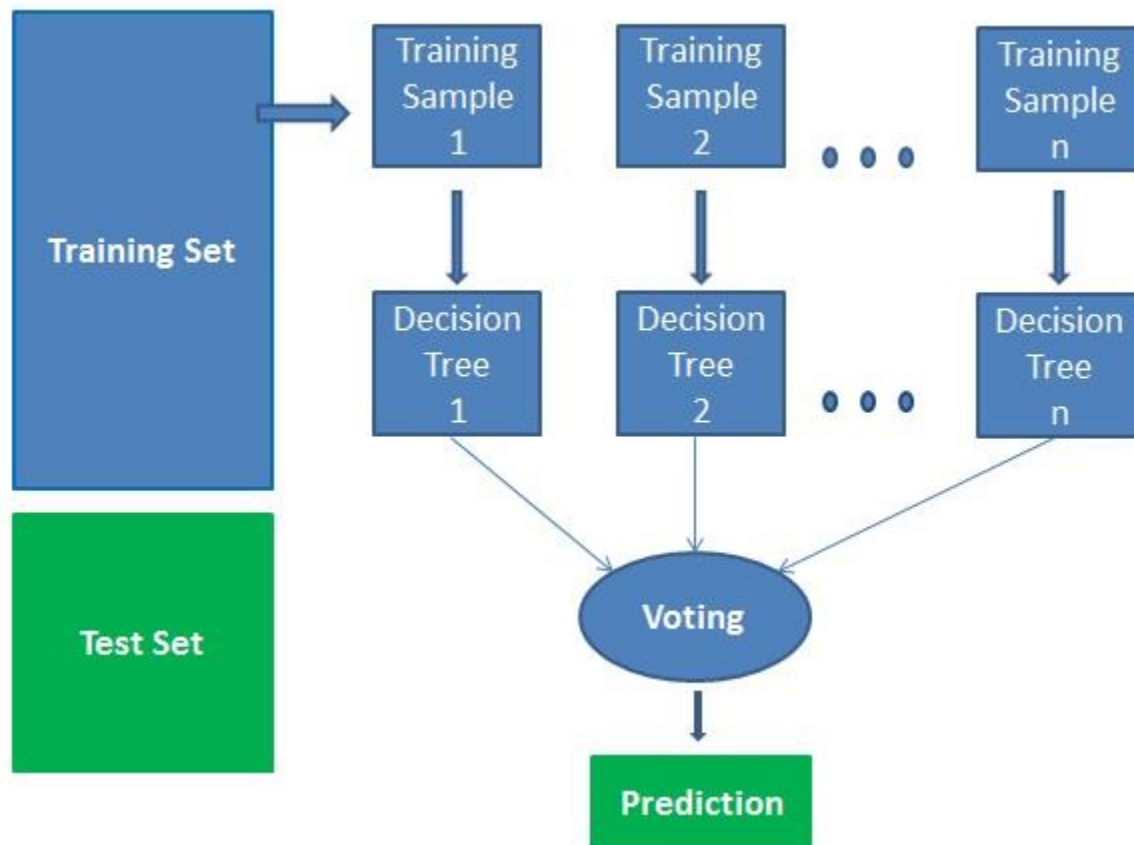
It technically is an ensemble method (based on the divide-and-conquer approach) of decision trees generated on a randomly split dataset. This collection of decision tree classifiers is also known as the forest. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio, and Gini index for each attribute. Each tree depends on an independent

random sample. In a classification problem, each tree votes and the most popular class is chosen as the final result. In the case of regression, the average of all the tree outputs is considered as the final result. It is simpler and more powerful compared to the other non-linear classification algorithms.

How the Algorithm Works ?

It works in four steps:

1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.



Generating a Random Forest Classifier Model

We will use the same split that we used on the decision tree and naïve bayes algorithm

```
In [74]: clf=RandomForestClassifier()
```

```
In [75]: clf.fit(X_train,y_train)
```

```
Out[75]: RandomForestClassifier()
```

```
In [76]: y_pred=clf.predict(X_test)
```

```
In [77]: Accuracy_rf = metrics.accuracy_score(y_test, y_pred)
print("Accuracy:", Accuracy_rf)
```

```
Accuracy: 0.7916666666666666
```

Feature Importance using Random Forest Classifier

```
[341]: # Lets calculate feature importance using Random Forest Classifier
dn = {'features':feature_columns,'score':clf.feature_importances_}
df = pd.DataFrame.from_dict(data=dn).sort_values(by='score',ascending=False)
```

```
[342]: plot= sns.barplot(x='score',y='features',data=df,orient='h')
plot.set(xlabel="Score",ylabel="features",title='Feature Importance of Random Forest Classifier')
plt.setp(plot.get_xtickLabels(), rotation = 90)
plt.show()
```

The resultant graph is :

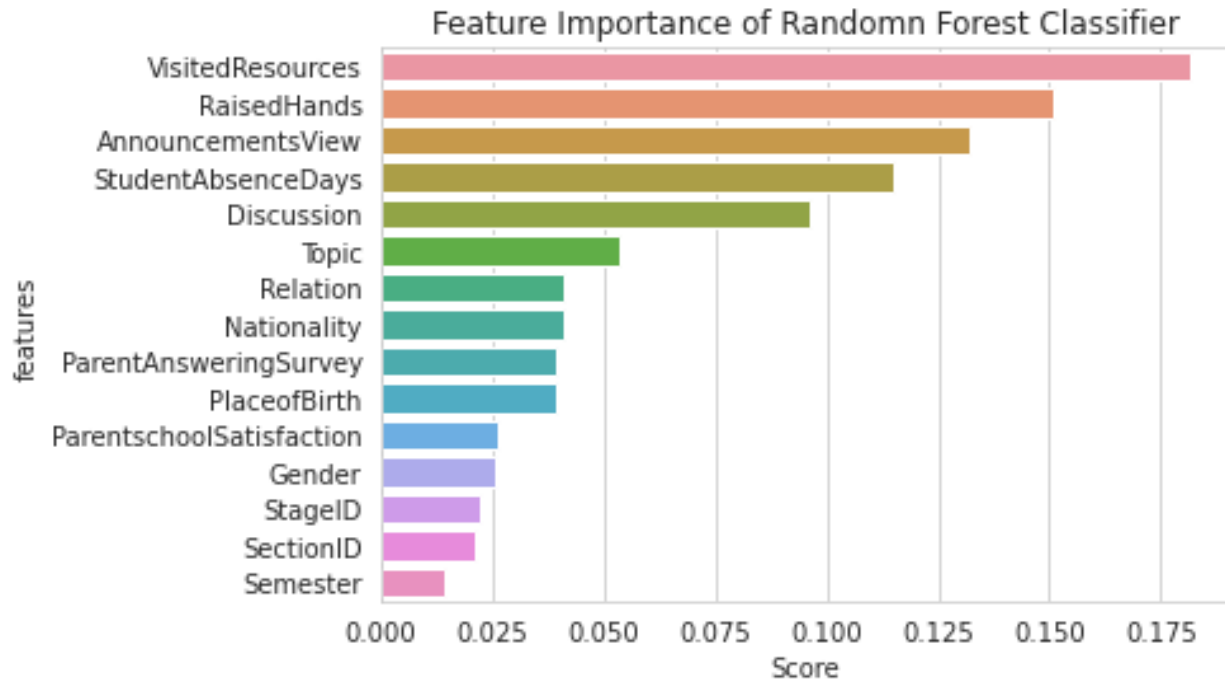


Figure 10: Features Importance

This also confirms our findings in the EDA Section. The Resource Visited , Raised Hands , Announcement View , the factors that contribute to participation in class , all result in determination of the grade and are relevant to the grade you obtain.

Conclusions and Future Work

With our work , we were able to find some very interesting and hidden insights from the data of an e-learning software that is currently in use. We then managed to create 3 models and compare their accuracies in order to select the best model for the prediction task based on the criterion of accuracy on the test set. From our analysis , we found that random Forest Classifier would be best to train the model upon and use in the production.

Through this whole process, we were able to create from scratch a model that is capable of performing/predicting the performance of a student in an adaptive e-learning environment by performing a real time classification task.

For the future work, it will be interesting to train and test the model with more large dataset to see if it maintains the same accuracy and there is no bias in the model's predictions. It will also be interesting to tweak the model more so as to reach a desired accuracy of 90 % or more.

For example , it will interesting to see how other supervised classification learning models give the results such as Support Vector Machines, k-Nearest Neighbour Algorithm fare on this dataset.

Results

The resultant code and python notebook is available at Kaggle Kernel on the following link :

<https://www.kaggle.com/zeeshankhand/machine-learning-course-project>

The suggestions , guidelines and improvements are always welcome. I hope our learning of the dataset and the results could be helpful to anyone working on this in the future.

References

1. <https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn>
2. <https://www.datacamp.com/community/tutorials/decision-tree-classification-python>
3. Scikit-learn Documentation : https://scikit-learn.org/stable/user_guide.html
4. Random Forest Classifier : <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>