Honours Degree in Computing

Data Analytics Assessment:
Analyse a dataset.

Submitted by: Disi Pepiq, B00138946

Submission date

Declaration

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, except where otherwise stated. I/We have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references.

I/We understand that plagiarism, collusion, and copying are grave and serious offences and accept the penalties that would be imposed should I/we engage in plagiarism, collusion or copying. I acknowledge that copying someone else's assignment, or part of it, is wrong, and that submitting identical work to others constitutes a form of plagiarism. I/We have read and understood the colleges plagiarism policy 3AS08 (available here).

This material, or any part of it, has not been previously submitted for assessment for an academic purpose at this or any other academic institution.

I have not allowed anyone to copy my work with the intention of passing it off as their own work.

Name: _____Disi Pepiq_____

Dated: _____01/11/23____

(Printing your name here will be taken as a digital signature)

Table of Contents . . .

Business Understanding

Give a brief background to the dataset itself.

- This dataset is a student dataset.
- It contains the student's information and education performance.
    - Such information are, students sex, age course and etc
- The database itself consists of 21 attributes.
- The dataset contains a Dependent Variable called Label.
- The Label contains a Fail or Pass variable which shows if the student passed or failed a specific Course.

Business objective

State what you want to accomplish when mining this data from a business point of view.

- Improving students' performance, grade, and success
- Enhance the students study effort and self-Regulation.
- Changing the student's learning style, to make learning more engaging.
- Improving Student's Extrinsic motivation and Intrinsic motivation
- Creating a productive environment such as activities, that encourages students to collab as a team in groups.

Data Mining objectives

State your technical objectives for mining the data.

- Visualisation which includes, plotting graphs, scatterplots
    - For example, scatter plots to visualise the number of males and females.
- Classification: Making predictions which include, predicting students' performance, grade, success

- Associations: finding associations between students working in groups or alone
- Deviation/Anomaly Detection: Identifying students who show markedly different grade performance, behaviour, or learning style.
- Clustering: Putting Students in groups with similar disciplines such as Humanities, Computing, Business and Engineering.

Data Understanding

Describe the data and summary statistics.

- The dataset consists of 1240 rows and 1 column. The database itself consists of 21 attributes.
- I decided to use the Pass Label.

Columns with independent variables

- "ExtrinsicMotivation";"GroupWork";"IntrinsicMotivation";"SelfEfficacy";"Self Regulation";"StudyEffort";"StudyTime";"Openness";"Conscientiousness";"HighSchoolAverage";"HighSchoolEnglish";"HighSchoolMaths";"Sex";"Age";"Discipline";"Course";"Modality";"LearningStyle";"id";"Label"

- Using data.info to get the independent Variables.

Attrtibute	Description	Datatype	Mean	Min	max	Range	25%	50%	75%	St.dev
AcademicYear	The year the student enrolled in college	Numeric	2016	2015	2017	2	2015	2016	2017	0.79
ExtrinsicMotivation	Students that are motivated to achieve a good grade	Numeric	6.7	0.2	45	44.8	5.6	6.7	7.8	2
GroupWork	Students that prefer working in groups rather than alone	Numeric	6.5	0	10	10	4.3	7.2	8.9	2.9
IntrinsicMotivation	Students that are motivated by desire to learn new skills	Numeric	5.5	0	10	10	4.4	5.6	6.6	1.6
SelfEfficacy	The number of student's that hopes in completing goals or tasks	Numeric	5.8	1.1	45	43.9	4.7	5.8	6.6	1.9
SelfRegulation	Student's ability to control their behaviour	Numeric	4.7	0.2	8.5	8.3	3.8	4.8	5.5	1.4
StudyEffort	The amount effort that the student puts into studying	Numeric	5.4	0.5	10	9.5	4.4	5.4	6.6	1.6
StudyTime	Number of hours the student is studying	Numeric	6.1	0	10	10	4.7	6	7.6	2
Openness	Students that are willing to learn or engage in new things or ideas	Numeric	4.6	0	9.8	9.8	3.5	4.5	5.6	1.6
Conscientiousness	The student's level of discipline and organisation	Numeric	5.4	0	10	10	4.4	5.4	6.4	1.5
HighSchoolAverage	Student's average grade in high school	Numeric	60.8	40.0	79.8	39.8	55.8	61.8	67.1	8.7
HighSchoolEnglish	Student's grade in the English subject	Numeric	57.9	40.0	80	40	52	59	64	8.6
HighSchoolMaths	Student's grade in the Maths subject	Numeric	51.4	40.0	73	33	46	51	57	7.2
Age	The age of each student	Numeric	23.5	18	51	33	21	25	51	5.9

Numeric Variables Table

- Using data.describe() to get the Numeric variables.

Categorical Variables table

Attribute	Description	datatype	count	Unique/ number of values	Unique Values	Mode	Freq
Sex	The gender of each student	Categorical	1235	2	Male, Female	Male	658
Discipline	Represents the field of study that each student is doing	Categorical	1235	4	Humanities, Computing, Business and Engineering	Humanities	449
Course	Represents the course the student is doing	Categorical	1235	9	Computing, AppliedSocialCare, Business, CommunityDev , CreativeDigitalMedia, EarlyChildCare, Engineering, InternationalBusiness and BusinessWithIT	Computing	219
Modality	The modes of learning used, such visual, auditory.	Categorical	1134	3	Kineastheic, Visual and Auditory	Visual	881
LearningStyle	The type of learning style of each student	Categorical	1240	3	Deep, Shallow and Strategic	Deep	669
Label	Represents label of study, which shows whether the student failed or passed a particular course	Categorical	1240	2	Pass, Fail	Pass	721

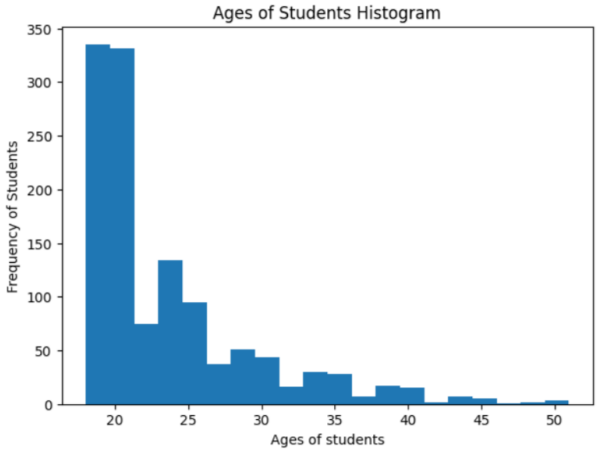
- Using data.describe(include='object') to get categorical variables
- There are no ordinal variables in this dataset.

- The dataset seems to be reasonably distributed.
- The count for modality is a bit little and not too close to 1240.
- The data seems to have very a small or no presence of outliners.

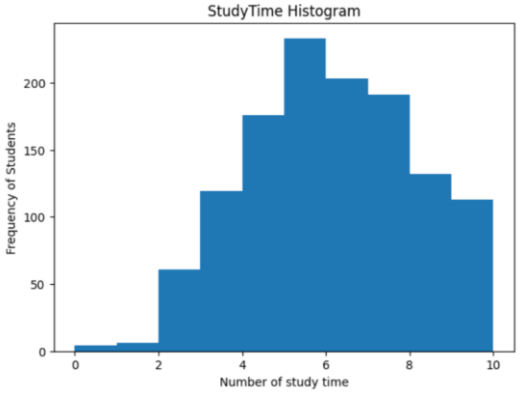
Explore the data.

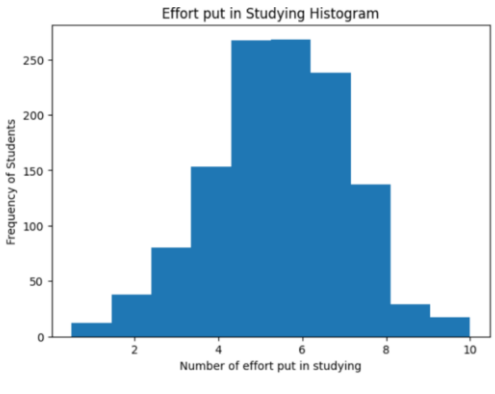
```
plt.figure(figsize= (7, 5))
plt.hist(x= data.Age, bins=20)
plt.xlabel('Ages of students')
plt.ylabel('Frequency of Students')
plt.title('Ages of Students Histogram')
```

```
Text(0.5, 1.0, 'Ages of Students Histogram')
```



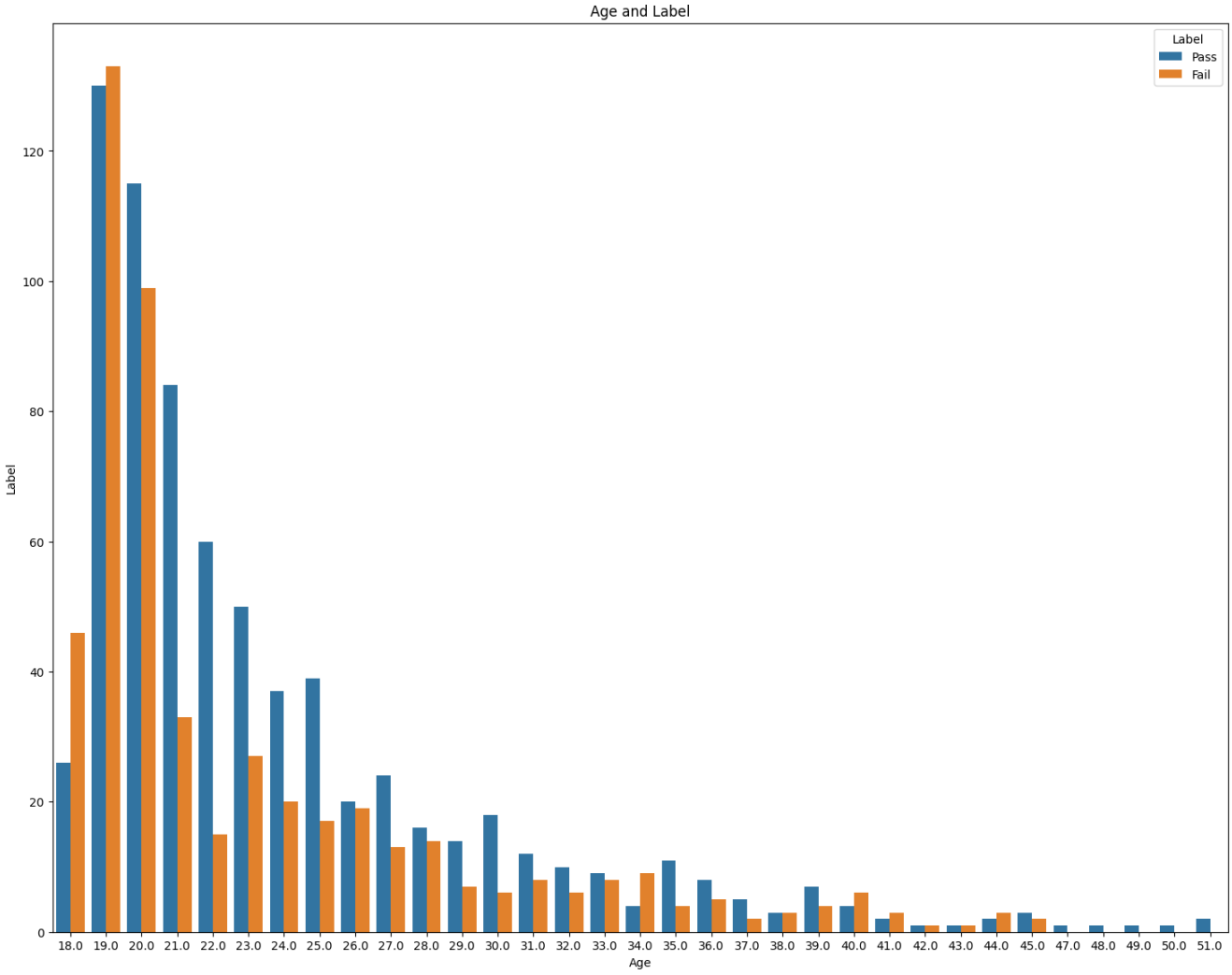
- This is an age histogram.
- The distribution is right skewed with most of values being low values, the most occurring ages are between 18 to 22.
- This distribution indicates little variability with most students having a narrow age range.





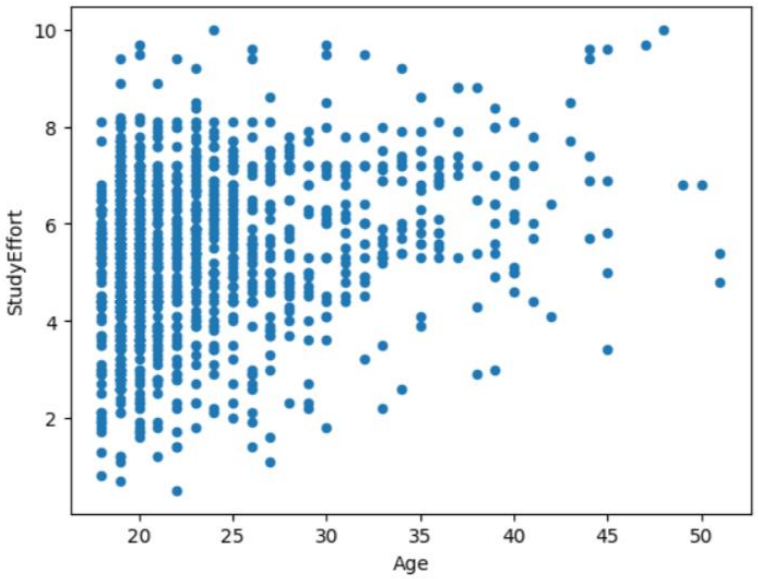
- These are histogram StudyTime and StudyEffort Histograms.
- The distribution is left skewed with most of values being high values, the most occurring for studyTime are roughly between 5 to 6, and the same for StudyEffort.
- Left skewed also indicates limited variability and that the students are not putting much time and effort into studying.

Attributes of interest to be explored visually and discussed in relation to the class label.



- Older students with ages from 46 to 51 only got pass rates and no fails.
- Students aged 19 got more fail rates than pass rates.
- Overall, most Students got a pass grade than a fail.

Visualising Correlations with age and studyEffort attributes



This shows that age and studyEffort have no Correlations.

Verify data quality.

Explore and discuss the following:

Does the dataset have missing values? Which attributes are affected? What's the likely impact?

- This dataset does have missing values with some attributes containing very low to no missing values.
- The attributes, openness, Conscientiousness and LearningStyle have no missing values.

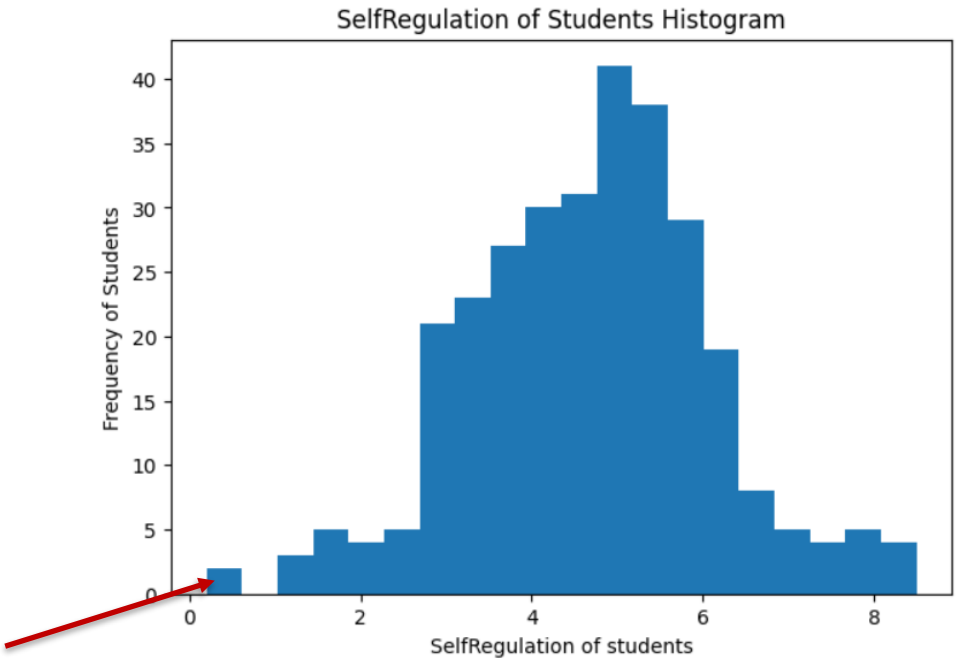
- Diagram below shows that SelfRegulation and Modality has the most missing data:
  - 'SelfRegulation' has 304 values with 936 missing values.
  - Modality has 1134 values with 106 missing values.
- The likely impact of this is that it can affect the student grade.
- The dataset appears to have no errors, no bias, and no inconsistencies.

AcademicYear	5
ExtrinsicMotivation	1
GroupWork	1
IntrinsicMotivation	2
SelfEfficacy	1
SelfRegulation	936
StudyEffort	1
StudyTime	2
Openness	0
Conscientiousness	0
HighSchoolAverage	5
HighSchoolEnglish	5
HighSchoolMaths	5
Sex	5
Age	5
Discipline	5
Course	5
Modality	106
LearningStyle	0
id	0
Label	0

Shows the number of missing data of each attribute.

The histogram for SelfRegulation below shows that there are outliners ,the red arow shows the outliner.

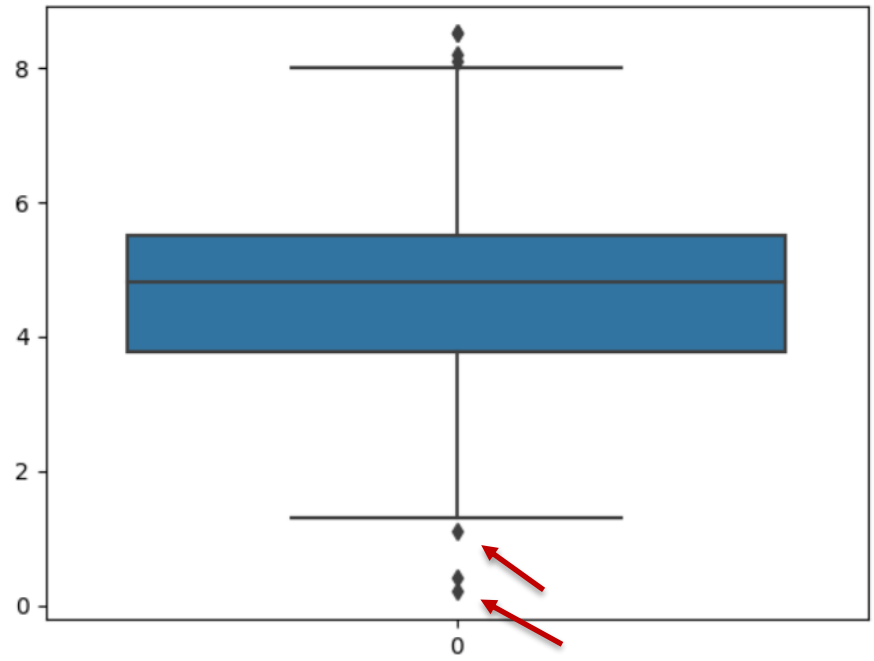
- The graph is negative skewed.
- There is a presence of low outliers.



The Boxplot for SelfRegulation below shows that there are outliners ,the red arrows show the outliners.

```
sns.boxplot(data['SelfRegulation'])
```

```
<Axes: >
```



Are there sufficient attributes and examples to achieve your mining objectives?

- The 21 attributes in the dataset may be enough for some data mining goals.
- There are many examples to achieve the mining objectives, when working with the label class like "Pass" or "Fail."

Algorithm, Test Design and Baseline

I chose the decision tree algorithm for modelling the baseline.

- I chose the decision tree algorithm classifier because it's a simple model algorithm to use and understand compared to other algorithms.
- I can visualize the tree which makes it easier to view information.
- The tree able to work with both categorical and numerical data.
- Even though the true model that the data were created slightly deviates from its assumptions, it still performs good.
- This algorithm is also able to deal with issues with multi-output.

(scikit, 2023)

I chose the classification metrics for measuring performance (Accuracy, Recall, Precision, F1 Measure)

Accuracy

- Accuracy can be described as the ratio of all positive (P) and negative (N) observations to the true positives (TP) and true negatives. (TN)
- This also indicates the proportion of the total number of predictions generated by the machine learning model that will result in a right prediction. (KumarI, 2023)

Recall

- Recall measures how well it can identify the positives from actual positives.
- For instance, the recall would be the percentage of positive reviews that your model correctly predicted to be positive, if it were attempting to identify good reviews.

Precision

- Precision measures the percentage of correctly predicted positive labels.
- It's also used to trade-off false positives and false negatives when combined with the recall score.

F1 Measure

- F1 Measure represents the harmonic mean of the recall and precision positive class.
- It gives an average between precision and recall.

I chose the Split-Validation for the validation technique.

How it works

- The initial data set is split into two halves at random, test dataset and training dataset.
- The original sample is split around 70% to 30%, 70% for training and 30% for testing.
- The testing dataset is for assessing the model's performance and the training dataset is for training the data mining model.

Why I chose it?

- It's a simple technique to split data.
- It's easy to understand and to implement.
- It can be used with the tree algorithm and with the metrics classification .

Document and discuss the performance.

```
              precision    recall  f1-score   support

        Fail       0.71      0.71      0.71       149
        Pass       0.81      0.80      0.80       223

    accuracy                           0.77       372
   macro avg       0.76      0.76      0.76       372
weighted avg       0.77      0.77      0.77       372

[[106  43]
 [ 44 179]]
```

- The performance of this dataset is reasonably good.
- For accuracy performance is high at 0.77 which is very good
- The pass rates for precision, recall and f1-score are roughly at 0.80 which is good pass rates.

Data Preparation

Selecting the data

Removal of column

More than 40% missing data

- I have decided for 'SelfRegulation' attribute to drop this column table altogether, due to it have more than 40% missing data.
- The attribute has over half of its values missing; therefore, it could be a useless column.
- By removing this column, the overall performance of the dataset will not be affected or changed.
- The accuracy is the same compared to previous one.

```
## remove the SelfRegulation column
data1 = data1.drop(['SelfRegulation'], axis=1);
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1235 entries, 0 to 1234
Data columns (total 20 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   AcademicYear         1235 non-null   float64
 1   ExtrinsicMotivation  1235 non-null   float64
 2   GroupWork            1235 non-null   float64
 3   IntrinsicMotivation  1235 non-null   float64
 4   SelfEfficacy         1235 non-null   float64
 5   StudyEffort          1235 non-null   float64
 6   StudyTime            1235 non-null   float64
 7   Openness             1235 non-null   float64
 8   Conscientiousness    1235 non-null   float64
 9   HighSchoolAverage    1235 non-null   float64
 10  HighSchoolEnglish    1235 non-null   float64
 11  HighSchoolMaths      1235 non-null   float64
 12  Sex                  1235 non-null   object 
 13  Age                  1235 non-null   float64
 14  Discipline           1235 non-null   object 
 15  Course               1235 non-null   object 
 16  Modality             1131 non-null   object 
 17  LearningStyle        1235 non-null   object 
 18  id                   1235 non-null   int64  
 19  Label                1235 non-null   object 
```

```
              precision    recall  f1-score   support

        Fail       0.70      0.72      0.71       149
        Pass       0.81      0.79      0.80       223

    accuracy                           0.77       372
   macro avg       0.76      0.76      0.76       372
weighted avg       0.77      0.77      0.77       372

[[108  41]
 [ 46 177]]
```

Removal of rows

Less than 5% missing data

- The attributes, AcademicYear, ExtrinsicMotivation, GroupWork, IntrinsicMotivation, SelfEfficacy, StudyEffort, StudyTime, Openness, Conscientiousness, HighSchoolAverage, HighSchoolEnglish, HighSchoolMaths, Sex, Age, Discipline, Course , LearningStyle, id, Label.
    - All have less than 5% of missing data, therefore, remove these rows.

- Removing these rows, the overall performance of the dataset will not be affected or changed.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1235 entries, 0 to 1234
Data columns (total 21 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   AcademicYear         1235 non-null   float64
 1   ExtrinsicMotivation  1235 non-null   float64
 2   GroupWork            1235 non-null   float64
 3   IntrinsicMotivation  1235 non-null   float64
 4   SelfEfficacy         1235 non-null   float64
 5   SelfRegulation       304 non-null    float64
 6   StudyEffort          1235 non-null   float64
 7   StudyTime            1235 non-null   float64
 8   Openness             1235 non-null   float64
 9   Conscientiousness    1235 non-null   float64
 10  HighSchoolAverage    1235 non-null   float64
 11  HighSchoolEnglish    1235 non-null   float64
 12  HighSchoolMaths      1235 non-null   float64
 13  Sex                  1235 non-null   object
 14  Age                  1235 non-null   float64
 15  Discipline           1235 non-null   object
 16  Course               1235 non-null   object
 17  Modality             1131 non-null   object
 18  LearningStyle        1235 non-null   object
 19  id                   1235 non-null   int64
 20  Label                1235 non-null   object
dtypes: float64(14), int64(1), object(6)
```

	precision	recall	f1-score	support
Fail	0.70	0.72	0.71	149
Pass	0.81	0.80	0.80	223
accuracy			0.77	372
macro avg	0.76	0.76	0.76	372
weighted avg	0.77	0.77	0.77	372

Univariate Imputation

Between 5% and 40% missing data

The attribute Modality has between 5% and 40% of missing data.

```
Int64Index: 1235 entries, 0 to 1234
Data columns (total 20 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   AcademicYear        1235 non-null   float64
 1   ExtrinsicMotivation  1235 non-null   float64
 2   GroupWork           1235 non-null   float64
 3   IntrinsicMotivation  1235 non-null   float64
 4   SelfEfficacy        1235 non-null   float64
 5   StudyEffort         1235 non-null   float64
 6   StudyTime           1235 non-null   float64
 7   Openness            1235 non-null   float64
 8   Conscientiousness   1235 non-null   float64
 9   HighSchoolAverage   1235 non-null   float64
 10  HighSchoolEnglish   1235 non-null   float64
 11  HighSchoolMaths     1235 non-null   float64
 12  Sex                 1235 non-null   object
 13  Age                 1235 non-null   float64
 14  Discipline          1235 non-null   object
 15  Course              1235 non-null   object
 16  Modality            1235 non-null   object
 17  LearningStyle       1235 non-null   object
 18  id                  1235 non-null   int64
 19  Label               1235 non-null   object
```

Performance of imputation

	precision	recall	f1-score	support
Fail	0.74	0.72	0.73	149
Pass	0.82	0.83	0.82	223
accuracy			0.79	372
macro avg	0.78	0.78	0.78	372
weighted avg	0.79	0.79	0.79	372

```
[[108  41]
 [ 38 185]]
```

- This will fill all the missing data of Modality its mode.
- All the classification metrics went up compared to the first original confusion matrix performance.
- Therefore, by filling the missing data of modality with its mode improved the performance.
- Also, the accuracy went up to 0.79.
- This could be used for the final model.

Multivariate outlier removal:

- I used this because want to know if removing any outliers will boost or make the overall performance better.
- By removing the outliers of the attributes, this made the performance slightly better.
- The first original performance showed an accuracy of 0.77 and this one with 0.78.
- This could be used for the final pipline.

	precision	recall	f1-score	support
Fail	0.71	0.71	0.71	149
Pass	0.81	0.80	0.80	223
accuracy			0.77	372
macro avg	0.76	0.76	0.76	372
weighted avg	0.77	0.77	0.77	372

Hyperparameter Tuning

Hyperparameter Tuning of the Decision Tree

1st parameter values I used was criterion.

- For this I computed entropy to measure the information gain
- At each node of the tree information gain selects the best attribute
- Computing entropy gets the easiness.
- The performance of this is not bad with the accuracy being 0.76.

	precision	recall	f1-score	support
Fail	0.69	0.73	0.71	149
Pass	0.81	0.78	0.80	223
accuracy			0.76	372
macro avg	0.75	0.76	0.76	372
weighted avg	0.77	0.76	0.76	372

2nd parameter values I used was changing the impurity threshold.

- This used in decreasing or changing threshold value and used to split the internal node.
- The performance is quite low with the accuracy being 0.73, worse than the previous parameter.

	precision	recall	f1-score	support
Fail	0.73	0.50	0.59	149
Pass	0.72	0.88	0.79	223
accuracy			0.73	372
macro avg	0.73	0.69	0.69	372
weighted avg	0.73	0.73	0.71	372

3rd parameter values I used was changing the depth of the tree.

- This is in controlling and changing the depth of the tree.
- The performance is quite low with the accuracy being 0.70, worse than the previous parameter.

	precision	recall	f1-score	support
Fail	0.60	0.74	0.66	149
Pass	0.80	0.66	0.72	223
accuracy			0.70	372
macro avg	0.70	0.70	0.69	372
weighted avg	0.72	0.70	0.70	372

4th parameter values I used was changing the number of rows or samples that is needed in splitting the internal node.

- This is used changing the number of samples that's needed to split the internal node.
- The performance is low with the accuracy being 0.75.

	precision	recall	f1-score	support
Fail	0.68	0.71	0.70	149
Pass	0.80	0.78	0.79	223
accuracy			0.75	372
macro avg	0.74	0.75	0.74	372
weighted avg	0.75	0.75	0.75	372

Last parameter values I used was Changing the number of rows or samples that is needed at a leaf.

- This is used changing the number of samples that's needed to split the terminal (leaf) node.
- The performance is not too bad with the accuracy being 0.76.
- This had a similar performance with but criterion has a higher recall and f1score.

	precision	recall	f1-score	support
Fail	0.68	0.74	0.71	149
Pass	0.81	0.77	0.79	223
accuracy			0.76	372
macro avg	0.75	0.75	0.75	372
weighted avg	0.76	0.76	0.76	372

I also did use the grid search to find the best set of parameters, but it does not work for me, and I got fits failed.

Evaluation

The technique that had the best performance for my dataset was the Univariate Imputation. This was used for Modality attribute because which had missing values from 5% to 40%. By filling the missing data off Modality with its mode, the overall performance was the best. This technique has a 0.78 accuracy compared first performance with accuracy of 0.77. Also removing the outlier improved the overall performance slightly but not as much as imputation. Univariate Imputation performed better because it does a small amount of harm to the distribution of the attributes. Also, Removing the selfRegulation column and removing the rows of attributes with less than 5% of missing data did not affect or change any overall performance. From all the parameters I used for Hyperparameter tuning, the optimal parameter of this dataset was the using the criterion. This parameter didn't have a better performance than first baseline model, but it was the best from all the other parameters. There are limitations of dropping the selfRegulation because half of data was missing. In future projects this can be addressed by exploring this more in depth and seeing if there was another technique in dealing with this attribute rather than just removing it. I learned much information about each attribute in this dataset. From this dataset I did learn that selfRegulation was the attribute with the missing data and had outliers. This attribute was not really needed in this dataset as it had a lot of missing data. I Found surprising that the older age group had good pass grades. I also learned that using imputation does boost the performance.

References

scikit (2023) *1.10. decision trees*, *scikit*. Available at: https://scikit-learn.org/stable/modules/tree.html (Accessed: 01 December 2023).

KumarI, A. (2023) *Accuracy, precision, Recall & F1-Score - Python examples*, *Analytics Yogi*. Available at: https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/#What_is_Accuracy_Score (Accessed: 01 December 2023).

Power, A. (2023) *COMP H4030 - Data Analytics.* Available at: https://vle-bn.tudublin.ie/course/view.php?id=258 (Accessed: 15 December 2023).

Appendix

```
import pandas as pd, numpy as np;
import matplotlib.pyplot as plt;
import seaborn as sns;
%matplotlib inline
import warnings;
warnings.filterwarnings("ignore");
import xgboost;
from sklearn.model_selection import train_test_split, cross_val_predict;
from sklearn.tree import DecisionTreeClassifier as DTC;
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score;
from sklearn.preprocessing import LabelEncoder;
from sklearn.preprocessing import MinMaxScaler;
from sklearn.neighbors import LocalOutlierFactor;
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import GridSearchCV;

data = pd.read_csv('studentData.csv', sep=';')
data.head()

data.shape

data.isnull().sum()

data.info()

data.describe()
```

```
data.describe(include='object')

data.sample(10)

data.SelfRegulation

plt.figure(figsize= (7, 5))
plt.hist(x= data.Age, bins=20)
plt.xlabel('Ages of students')
plt.ylabel('Frequency of Students')
plt.title('Ages of Students Histogram')

plt.figure(figsize= (7, 5))
plt.hist(x= data.StudyEffort, bins=20)
plt.xlabel('Number of effort put in studying')
plt.ylabel('Frequency of Students')
plt.title('Effort put in Studying Histogram')

plt.figure(figsize= (7, 5))
plt.hist(x= data.StudyTime)
plt.xlabel('Number of study time')
plt.ylabel('Frequency of Students')
plt.title('StudyTime Histogram')

numeric_data = data[['StudyTime','StudyEffort', 'Age']]
```

```
data.plot.scatter(x = 'StudyEffort', y='Age')
plt.show()

plt.figure(figsize=(18, 14))
sns.countplot(x='Age', hue='Label', data=data)
plt.title('Age and Label')
plt.xlabel('Age')
plt.ylabel('Label')
plt.show()

plt.figure(figsize= (7, 5))
plt.hist(x= data.SelfRegulation, bins=20)
plt.xlabel('SelfRegulation of students')
plt.ylabel('Frequency of Students')
plt.title('SelfRegulation of Students Histogram')

sns.boxplot(data['SelfRegulation'])

plt.figure(figsize=(7, 5))
data['Modality'].value_counts().plot(kind='bar', color='skyblue')
plt.xlabel('Modality')
plt.ylabel('Frequency of Modality')
plt.title('Modality Bar Chart')
plt.show()

## separate the class label from regular attributes
y = data[['Label']]
```

```
X = data[data.columns.difference(['Label'])]

X.info()



baseline_sklearn_data = X.copy()

num_cols = baseline_sklearn_data.select_dtypes(include=['int64', 'float64']).columns

baseline_sklearn_data[num_cols] = baseline_sklearn_data[num_cols].apply(lambda
col: col.fillna(0, axis=0))

obj_cols = baseline_sklearn_data.select_dtypes(include=['object',
'category']).columns

baseline_sklearn_data[obj_cols] = baseline_sklearn_data[obj_cols].apply(lambda col:
col.fillna('?', axis=0))

baseline_sklearn_data[obj_cols] =
baseline_sklearn_data[obj_cols].apply(LabelEncoder().fit_transform)

print(baseline_sklearn_data.info())

baseline_sklearn_data.sample(10)


## Use a ratio of 70% training - 30% testing

X_train, X_test, y_train, y_test = train_test_split(baseline_sklearn_data, y,
test_size=0.3, random_state=1)

## Create a decision tree classifier

sk_dt = DTC()

## Train the decision tree model

sk_dt.fit(X_train, y_train)

## Make predictions on the test portion

predictions = sk_dt.predict(X_test)

## Check the predicted labels against the actual labels stored in y_test and output
various scores
```

```
print(classification_report(y_test, predictions))

## Output the confusion matrix

cm = confusion_matrix(y_test, predictions)

print(cm)


Handling with missing

## drop the rows where with attributes missing less than 5%

data1 = data.copy();

data1 = data1.dropna(subset=['AcademicYear', 'ExtrinsicMotivation', 'GroupWork',
'IntrinsicMotivation',

                    'SelfEfficacy', 'StudyEffort', 'StudyTime', 'HighSchoolAverage',
'HighSchoolEnglish',

                    'HighSchoolMaths', 'Sex', 'Age', 'Discipline', 'Course']);

data1.info()

# Split validatiuon using ratio of 70% training - 30% testing

X_train, X_test, y_train, y_test = train_test_split(baseline_sklearn_data, y,
test_size=0.3, random_state=1)

sk_dt = DTC()

## first train it on the train portion of X and y

sk_dt.fit(X_train, y_train)

# Make predictions on the test portion

predictions = sk_dt.predict(X_test)

report = classification_report(y_test, predictions)

print(report)

# Output the confusion matrix

cm = confusion_matrix(y_test, predictions)

print(cm)
```

```
## remove the SelfRegulation column

data1 = data1.drop(['SelfRegulation'], axis=1);

data1.info()

# Split validatiuon using ratio of 70% training - 30% testing

X_train, X_test, y_train, y_test = train_test_split(baseline_sklearn_data, y,
test_size=0.3, random_state=1)

sk_dt = DTC()

## first train it on the train portion of X and y

sk_dt.fit(X_train, y_train)

# Make predictions on the test portion

predictions = sk_dt.predict(X_test)

report = classification_report(y_test, predictions)

print(report)

# Output the confusion matrix

cm = confusion_matrix(y_test, predictions)

print(cm)


# Try to replace Modality's missing data with mode

data10 = data1.copy()

mode_value = data10['Modality'].mode()[0]

data10['Modality'] = data10['Modality'].fillna(mode_value)


data10.Modality

# Split validatiuon using ratio of 70% training - 30% testing

X_train, X_test, y_train, y_test = train_test_split(baseline_sklearn_data, y,
test_size=0.3, random_state=1)

sk_dt = DTC()

## first train it on the train portion of X and y
```

```
sk_dt.fit(X_train, y_train)

# Make predictions on the test portion

predictions = sk_dt.predict(X_test)

report = classification_report(y_test, predictions)

print(report)

# Output the confusion matrix

cm = confusion_matrix(y_test, predictions)

print(cm)


Removing outliers

num_cols = X.select_dtypes(include=['int64', 'float64']).columns

cat_cols = X.select_dtypes(include=['object', 'category']).columns

baseline_sklearn_data = X.copy()

baseline_sklearn_data[num_cols] = baseline_sklearn_data[num_cols].apply(lambda
col: col.fillna(0, axis=0))

baseline_sklearn_data[cat_cols] = baseline_sklearn_data[cat_cols].apply(lambda col:
col.fillna('?', axis=0))

baseline_sklearn_data[cat_cols] =
baseline_sklearn_data[cat_cols].apply(LabelEncoder().fit_transform)


## Apply LOF for outlier detection

lof = LocalOutlierFactor()

outlier_scores = lof.fit_predict(baseline_sklearn_data)

outliers_mask = (outlier_scores == -1)

## Removing outliers

X_no_outliers = baseline_sklearn_data[~outliers_mask]

y_no_outliers = y[~outliers_mask]

# #Use a ratio of 70% training - 30% testing
```

```
X_train, X_test, y_train, y_test = train_test_split(X_no_outliers, y_no_outliers,
test_size=0.3, random_state=1)

sk_dt = DTC()

sk_dt.fit(X_train, y_train)

predictions = sk_dt.predict(X_test)

print(classification_report(y_test, predictions))

cm = confusion_matrix(y_test, predictions)

print(cm)
```

Hyperparameter Tuning of the Decision Tree

```
baseline_sklearn_data = X.copy()

num_cols = baseline_sklearn_data.select_dtypes(include=['int64', 'float64']).columns

baseline_sklearn_data[num_cols] = baseline_sklearn_data[num_cols].apply(lambda
col: col.fillna(0, axis=0))

obj_cols = baseline_sklearn_data.select_dtypes(include=['object',
'category']).columns

baseline_sklearn_data[obj_cols] = baseline_sklearn_data[obj_cols].apply(lambda col:
col.fillna('?', axis=0))

baseline_sklearn_data[obj_cols] =
baseline_sklearn_data[obj_cols].apply(LabelEncoder().fit_transform)

baseline_sklearn_data.sample(10)

X_train, X_test, y_train, y_test = train_test_split(baseline_sklearn_data, y,
test_size=0.3, random_state=1)

## Criterion

dtc = DTC(criterion='entropy', random_state=1)

dtc.fit(X_train, y_train)

predictions = dtc.predict(X_test)

print(classification_report(y_test, predictions))

cm = confusion_matrix(y_test, predictions)
```

```
print(cm)


baseline_sklearn_data = X.copy()

num_cols = baseline_sklearn_data.select_dtypes(include=['int64', 'float64']).columns

baseline_sklearn_data[num_cols] = baseline_sklearn_data[num_cols].apply(lambda
col: col.fillna(0, axis=0))

obj_cols = baseline_sklearn_data.select_dtypes(include=['object',
'category']).columns

baseline_sklearn_data[obj_cols] = baseline_sklearn_data[obj_cols].apply(lambda col:
col.fillna('?', axis=0))

baseline_sklearn_data[obj_cols] =
baseline_sklearn_data[obj_cols].apply(LabelEncoder().fit_transform)

baseline_sklearn_data.sample(10)

X_train, X_test, y_train, y_test = train_test_split(baseline_sklearn_data, y,
test_size=0.3, random_state=1)

## changin the impurity threshold

dtc = DTC(min_impurity_decrease=0.01, random_state=1);

dtc.fit(X_train, y_train)

predictions = dtc.predict(X_test)

print(classification_report(y_test, predictions))

cm = confusion_matrix(y_test, predictions)

print(cm)


baseline_sklearn_data = X.copy()

num_cols = baseline_sklearn_data.select_dtypes(include=['int64', 'float64']).columns

baseline_sklearn_data[num_cols] = baseline_sklearn_data[num_cols].apply(lambda
col: col.fillna(0, axis=0))

obj_cols = baseline_sklearn_data.select_dtypes(include=['object',
'category']).columns
```

```
baseline_sklearn_data[obj_cols] = baseline_sklearn_data[obj_cols].apply(lambda col:
col.fillna('?', axis=0))

baseline_sklearn_data[obj_cols] =
baseline_sklearn_data[obj_cols].apply(LabelEncoder().fit_transform)

baseline_sklearn_data.sample(10)

X_train, X_test, y_train, y_test = train_test_split(baseline_sklearn_data, y,
test_size=0.3, random_state=1)


## change the depth of the tree

dtc = DTC(max_depth=5, random_state=1);

dtc.fit(X_train, y_train)


predictions = dtc.predict(X_test)

print(classification_report(y_test, predictions))

cm = confusion_matrix(y_test, predictions)

print(cm)


baseline_sklearn_data = X.copy()

num_cols = baseline_sklearn_data.select_dtypes(include=['int64', 'float64']).columns

baseline_sklearn_data[num_cols] = baseline_sklearn_data[num_cols].apply(lambda
col: col.fillna(0, axis=0))

obj_cols = baseline_sklearn_data.select_dtypes(include=['object',
'category']).columns

baseline_sklearn_data[obj_cols] = baseline_sklearn_data[obj_cols].apply(lambda col:
col.fillna('?', axis=0))

baseline_sklearn_data[obj_cols] =
baseline_sklearn_data[obj_cols].apply(LabelEncoder().fit_transform)

baseline_sklearn_data.sample(10)

X_train, X_test, y_train, y_test = train_test_split(baseline_sklearn_data, y,
test_size=0.3, random_state=1)
```

```
## Changing the number of rows or samples that is needed in spliting the internal node

dtc = DTC(min_samples_split=10, random_state=1);

dtc.fit(X_train, y_train)


predictions = dtc.predict(X_test)

print(classification_report(y_test, predictions))

cm = confusion_matrix(y_test, predictions)

print(cm)


baseline_sklearn_data = X.copy()

num_cols = baseline_sklearn_data.select_dtypes(include=['int64', 'float64']).columns

baseline_sklearn_data[num_cols] = baseline_sklearn_data[num_cols].apply(lambda col: col.fillna(0, axis=0))

obj_cols = baseline_sklearn_data.select_dtypes(include=['object', 'category']).columns

baseline_sklearn_data[obj_cols] = baseline_sklearn_data[obj_cols].apply(lambda col: col.fillna('?', axis=0))

baseline_sklearn_data[obj_cols] = baseline_sklearn_data[obj_cols].apply(LabelEncoder().fit_transform)

baseline_sklearn_data.sample(10)

X_train, X_test, y_train, y_test = train_test_split(baseline_sklearn_data, y, test_size=0.3, random_state=1)


## Changing the number of rows or samples that is needed at a leaf

dtc = DTC(min_samples_leaf=5, random_state=1);

dtc.fit(X_train, y_train)
```

```
predictions = dtc.predict(X_test)
print(classification_report(y_test, predictions))
cm = confusion_matrix(y_test, predictions)
print(cm)

## Using the grid search to find the best set of paremeters
params = {
   'criterion': ['gini', 'entropy'],
   'max_depth': range(3, 11),
   'min_samples_split': range(5, 16),
   'min_samples_leaf': range(3, 10),
   'min_impurity_decrease': [0.01, 0.02, 0.03]
}

grid_search = GridSearchCV(DTC(random_state=1), param_grid=params,
verbose=2)

grid_search.fit(X, y)
print(grid_search.best_params_)
print(grid_search.best_estimator_)
print(grid_search.best_score_)
```