Prediction of Student Performance in Mathematics in Secondary Education

SENG 474 Project Report

December 5th, 2018

Han-wei Lin V00803987
Kibae Kim V00878886
Andrew Yang V00878595

Abstract

This project intends to explore and investigate the prediction of student performance in Mathematics in secondary education by building a decision tree classifier learning model. The dataset used to train and evaluate the decision tree classifier is the *Student Performance Data Set* from the UCI Machine Learning Repository [2]. Data preprocessing is required due to the nature of the dataset and the machine learning library chosen. *Python* and *Scikit-learn* are the main tools used for data mining and data evaluation.

Table of Contents

1. Introduction

Student performance in an academic subject is commonly measured in grades. The grade achieved by a student in an academic subject usually indicates how well the student understands the materials and how solid the work demonstrated by the student in the subject. A student's grades can potentially determine the trajectory of his or her life, since academic grades can be a deciding factor of what university or college the student is accepted to, what scholarships the student can receive, or what career the student might be in. Student performance is determined by many factors. Academic performance has been linked to intelligence and personality. Students with higher IQs tend to achieve better in academic settings [7]. However, many other factors can possibly determine the academic performance of a student, such as demographic, social, and school related factors. In this project, we explore and investigate in predicting student performance in Mathematics in secondary education by building a decision tree classifier learning model.

2. Related Work

Al-Radaideh, Al-Shawakfa and Al-Najjar (2006) [1] used the CRISP framework to create a Decision Tree model of students' performance in their courses. They prepared the data using the WEKA toolkit and selected the most significant attributes using the feature selection approach.

First-year failure rates of Universities are high and for many years have baffled education administrators. Superby, Vandamme and Meskens (2006) [5] separated students into three different groups, low-risk, medium-risk, and high-risk, in order to determine which students were in most need of assistance. They used many psychological models coupled with data from student questionnaires to identify the most at risk factors. To process all this information and create a framework for student classification, Superby, Vandamme and Meskens used Decision Trees.

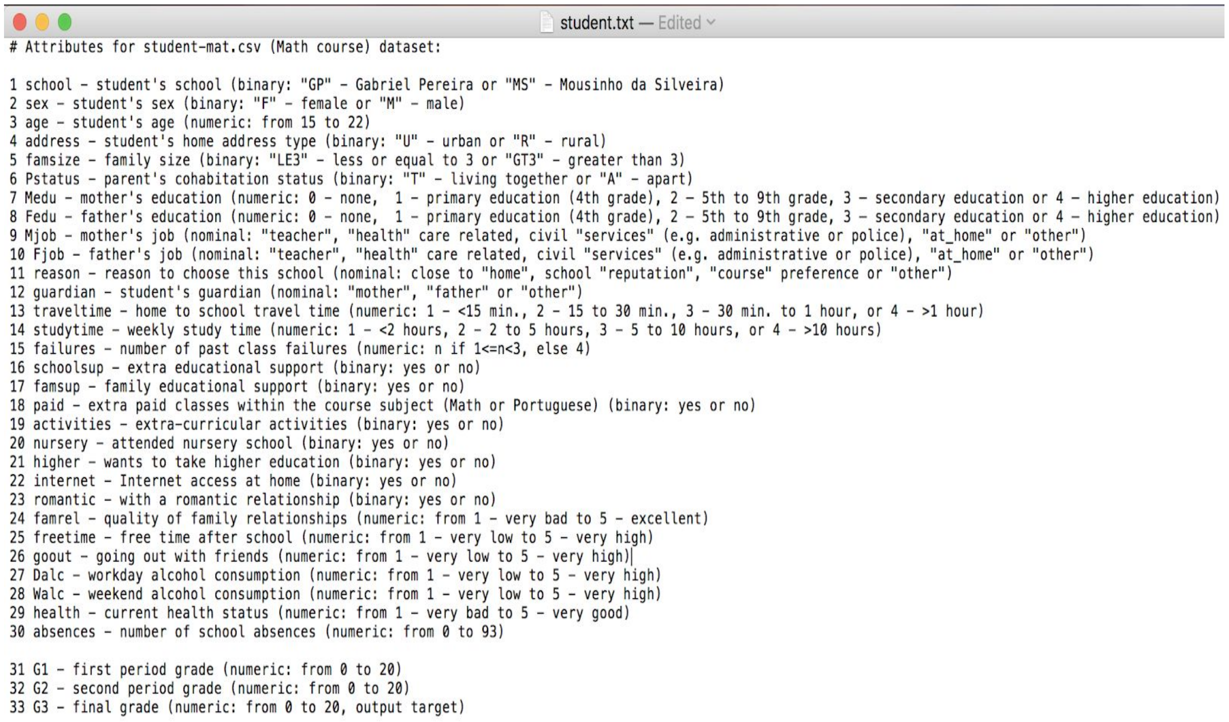
Using data collected from school reports and questionnaires Cortez and Silva (2008) [2] used data mining techniques to analyze the lack of success of Portuguese students in mathematics and Portuguese language classes. Data mining techniques employed were Decision Trees, Random Forest, Neural Networks and Support Vector Machines.

Kumar and Vijayalakshmi (2011) [3] extracted data from educational data using decision trees, rule mining and Bayesian networks to predict student performance for final exams. Their work could also help professors identify students who may need extra help in order to pass their final exams.

Ma, Liu, Wong and Lee (2000) [4] tackle the selection of students who may need remedial classes in Singapore’s Gifted Education Programme (GEP). A scoring function, named SBA, based on associated rules was created and employed.

3. Data Collection

The dataset collected to train and evaluate the decision tree classifier is the *student-mat.csv* file provided in the *Student Performance Data Set* from the UCI Machine Learning Repository [2]. The dataset contains 395 instances and each instance contains 33 attributes with the attribute *G3* - the final grade as the target attribute. This dataset approaches secondary school student performance in the subject of Mathematics of two Portuguese schools. The data attributes include student grades and demographic, social, and school related features and it was collected by using school reports and questionnaires [6]. The attributes information of *student-mat.csv* dataset is shown in Figure 1 below.



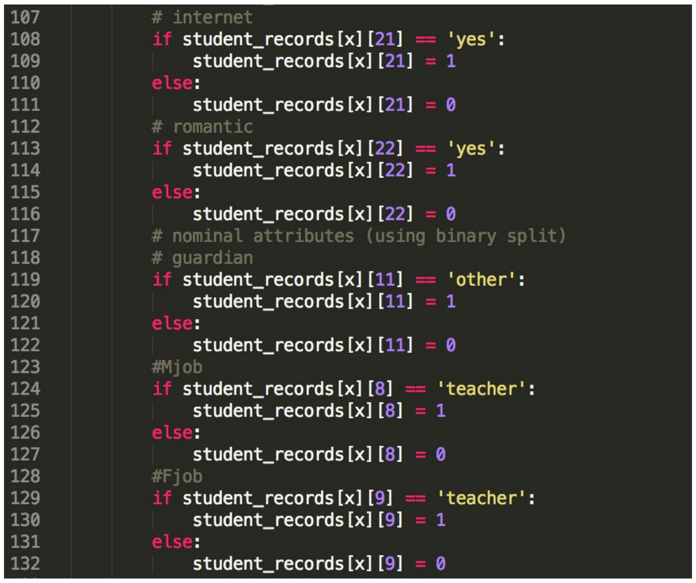
```
# Attributes for student-mat.csv (Math course) dataset:

1 school - student's school (binary: "GP" - Gabriel Pereira or "MS" - Mousinho da Silveira)
2 sex - student's sex (binary: "F" - female or "M" - male)
3 age - student's age (numeric: from 15 to 22)
4 address - student's home address type (binary: "U" - urban or "R" - rural)
5 famsize - family size (binary: "LE3" - less or equal to 3 or "GT3" - greater than 3)
6 Pstatus - parent's cohabitation status (binary: "T" - living together or "A" - apart)
7 Medu - mother's education (numeric: 0 - none,  1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
8 Fedu - father's education (numeric: 0 - none,  1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
9 Mjob - mother's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
10 Fjob - father's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
11 reason - reason to choose this school (nominal: close to "home", school "reputation", "course" preference or "other")
12 guardian - student's guardian (nominal: "mother", "father" or "other")
13 traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
14 studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
15 failures - number of past class failures (numeric: n if 1<=n<3, else 4)
16 schoolsup - extra educational support (binary: yes or no)
17 famsup - family educational support (binary: yes or no)
18 paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
19 activities - extra-curricular activities (binary: yes or no)
20 nursery - attended nursery school (binary: yes or no)
21 higher - wants to take higher education (binary: yes or no)
22 internet - Internet access at home (binary: yes or no)
23 romantic - with a romantic relationship (binary: yes or no)
24 famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
25 freetime - free time after school (numeric: from 1 - very low to 5 - very high)
26 goout - going out with friends (numeric: from 1 - very low to 5 - very high)
27 Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
28 Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
29 health - current health status (numeric: from 1 - very bad to 5 - very good)
30 absences - number of school absences (numeric: from 0 to 93)

31 G1 - first period grade (numeric: from 0 to 20)
32 G2 - second period grade (numeric: from 0 to 20)
33 G3 - final grade (numeric: from 0 to 20, output target)
```

Figure 1 Attributes information

4. Data Preprocessing

In this project, data preprocessing is required due to the nature of the dataset and the machine learning library chosen. In *student-mat.csv*, each instance consists of binary attributes, discrete numeric attributes, and nominal attributes, and most binary and nominal attributes have non-numeric values in the csv file. *Scikit-learn* is the machine learning library chosen for building the decision tree model. Moreover, *DecisionTreeClassifier* in *Scikit-Learn* only supports numeric data [9]. Hence, some data preprocessing is needed.

The *csv* library is used to load *student-mat.csv* into *StudentPerformance.py.* A sequence of *if-else* statements are used to convert the values of the binary and nominal attributes into numeric attributes, and a snippet of the code containing the conditional statements is shown in Figure 2. The conversion of binary attribute values into numeric values is simple. For example, the domain of the *sex* binary attribute is { “F”, “M”}; “F” is converted to 0, and “M” is converted to 1. Other binary attributes are converted in a similar manner. For nominal attributes, binary splits are used as the encoding method for conversion.
For example, the domain of the nominal attribute *Mjob* - mother’s education is { “teacher”, “health”, “services”, “at_home”, “other” }. *StudentPerformance.py* splits the domain into { { “teacher” }, { “health”, “services”, “at_home”, “other” } } and encodes attribute values in { “teacher” } as 1’s and attribute values in { “health”, “services”, “at_home”, “other” } as 0’s. The reason why binary splits are chosen as the encoding method for the conversion of nominal attributes is to reduce the programming complexity. One alternative approach is to use one-hot-encoding as the encoding scheme. However, this would add 14 additional attributes to each student record. Since there are only 33 attributes in each instance, we decided not to use one-hot-encoding as the encoding scheme.



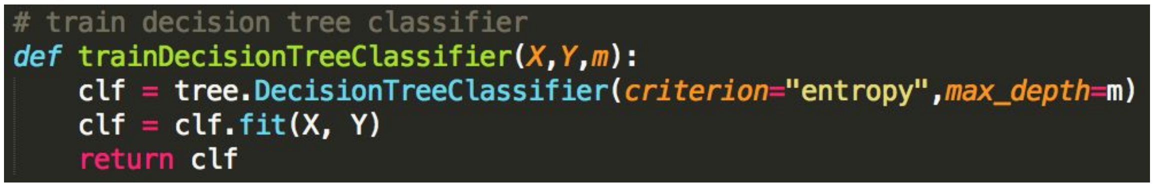
```
        # internet
        if student_records[x][21] == 'yes':
            student_records[x][21] = 1
        else:
            student_records[x][21] = 0
        # romantic
        if student_records[x][22] == 'yes':
            student_records[x][22] = 1
        else:
            student_records[x][22] = 0
        # nominal attributes (using binary split)
        # guardian
        if student_records[x][11] == 'other':
            student_records[x][11] = 1
        else:
            student_records[x][11] = 0
        #Mjob
        if student_records[x][8] == 'teacher':
            student_records[x][8] = 1
        else:
            student_records[x][8] = 0
        #Fjob
        if student_records[x][9] == 'teacher':
            student_records[x][9] = 1
        else:
            student_records[x][9] = 0
```

Figure 2

Moreover, the attributes *G1* - first period grade and *G2* - second period grade are strongly correlated with the target attribute *G3* - the final grade [2]. Hence, the attributes *G1* and *G2* are discarded during the data preprocessing, so the predictions of the learning model are more useful and more likely to reveal interesting patterns.

5. Data Mining

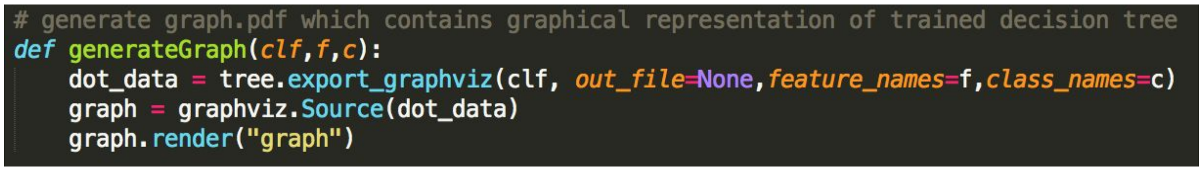
In the data mining phase of this project, the learning model is built based on a decision tree classifier. The reason why a decision tree classifier is chosen is because it is simple and suitable for this type of classification problem. Python and the *Scikit-learn* library are the main tools for building the classification model. Note that the target attribute *G3* is multiclass with 21 class labels from 0 to 20. The class *DecisionTreeClassifier* in *Scikit-Learn* [9] is used in *StudentPerformance.py* to build the decision tree classification model as shown in Figure 3. The *criterion* in the function call is set to “entropy”, so that entropy for the information gain is used to measure the quality of a split.



```python
# train decision tree classifier
def trainDecisionTreeClassifier(X,Y,m):
    clf = tree.DecisionTreeClassifier(criterion="entropy",max_depth=m)
    clf = clf.fit(X, Y)
    return clf
```

Figure 3

During the training phase of the classification model, 300 instances are chosen randomly with replacements from the dataset *student-mat.csv.* Hence, the size of the training set is approximately 75% of the size of the original dataset. The function *generateGraph* utilizes the library *graphviz* to output the graphical representation of a built decision tree in a *pdf* file named *graph.pdf* as shown in Figure 4.



```python
# generate graph.pdf which contains graphical representation of trained decision tree
def generateGraph(clf,f,c):
    dot_data = tree.export_graphviz(clf, out_file=None,feature_names=f,class_names=c)
    graph = graphviz.Source(dot_data)
    graph.render("graph")
```

Figure 4

StudentPerformance.py uses a *while* loop to build 10 decision tree classifiers for the purpose of evaluation. Since training instances are selected at random to build each decision tree classifier, the internal structure of each built classification model varies, which is a known disadvantage of decision tree classifiers. However, the predicating accuracy of each built model is quite consistent as shown in the Evaluation section.

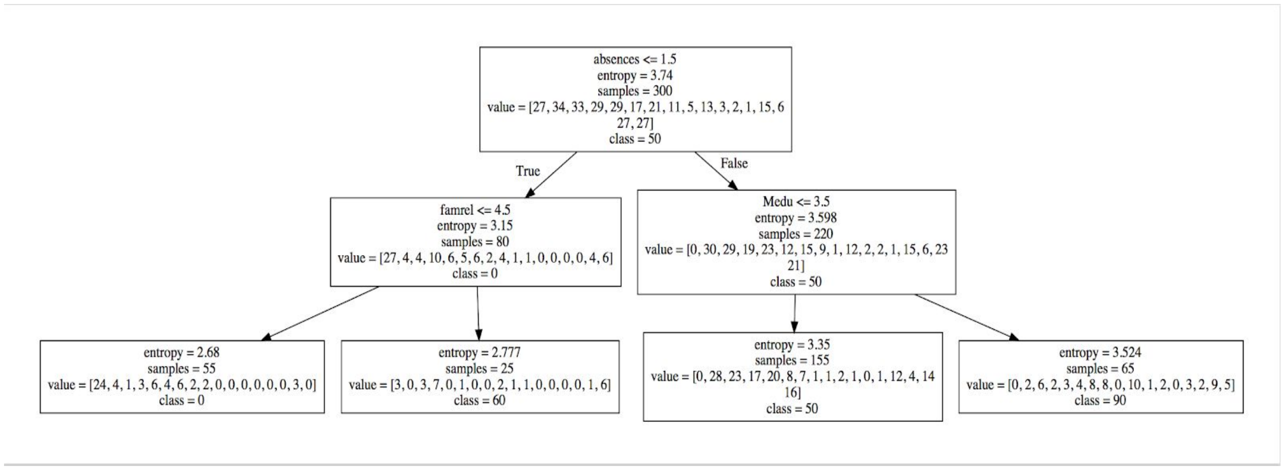

absences <= 1.5
entropy = 3.74
samples = 300
value = [27, 34, 33, 29, 29, 17, 21, 11, 5, 13, 3, 2, 1, 15, 6 27, 27]
class = 50
True
False
famrel <= 4.5
entropy = 3.15
samples = 80
value = [27, 4, 4, 10, 6, 5, 6, 2, 4, 1, 1, 0, 0, 0, 0, 4, 6]
class = 0
Medu <= 3.5
entropy = 3.598
samples = 220
value = [0, 30, 29, 19, 23, 12, 15, 9, 1, 12, 2, 2, 1, 15, 6, 23 21]
class = 50
entropy = 2.68
samples = 55
value = [24, 4, 1, 3, 6, 4, 6, 2, 2, 0, 0, 0, 0, 0, 0, 3, 0]
class = 0
entropy = 2.777
samples = 25
value = [3, 0, 3, 7, 0, 1, 0, 0, 2, 1, 1, 0, 0, 0, 0, 1, 6]
class = 60
entropy = 3.35
samples = 155
value = [0, 28, 23, 17, 20, 8, 7, 1, 1, 2, 1, 0, 1, 12, 4, 14 16]
class = 50
entropy = 3.524
samples = 65
value = [0, 2, 6, 2, 3, 4, 8, 8, 0, 10, 1, 2, 0, 3, 2, 9, 5]
class = 90

Figure 5

Figure 5 shows a partial built decision tree with depth 2 generated by *StudentPerformance.py.* P. Cortez and A. Silva [2] shows that the original dataset exhibits a normal distribution as shown in Figure 6. The sample training dataset in Figure 5 is quite consistent with the findings of P. Cortez and A. Silva [2], since the majority of students receive 50% in their final mathematics grades. Note that the class labels in Figure 5 are represented as percentages and the actual class labels of the target attribute is from 0 to 20. One interesting result from the particular tree in Figure 5 is that the first splitting attribute is the attribute *absences.* Moreover, after splitting, the node with *absences* less than or equal to 1.5 has the class label equals to 0% and the node with *absences* greater than 1.5 has the class label equals 50%. This result is counterintuitive, since the general belief is that students who have better attendance usually perform better in class. However, this result may due the nature of the *student-mat.csv* dataset. In the sample dataset in Figure 5, 27 students, which is nearly 10% of the training dataset, received 0% for their final grades and all of these 27 students have

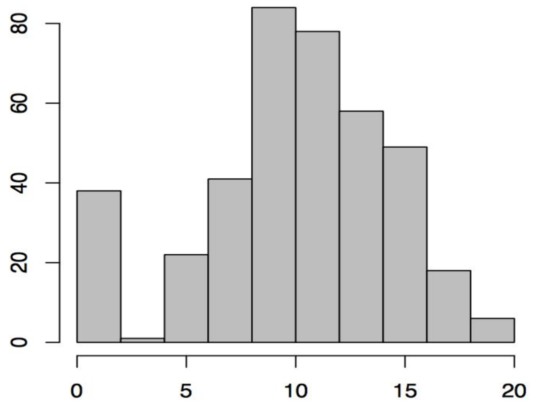
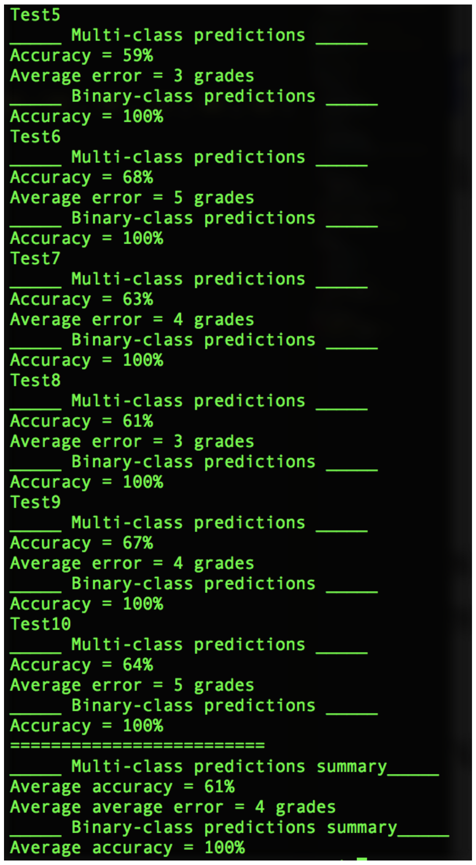


Figure 6 - referenced from P. Cortez and A. Silva [2]

absences less than or equal to 1.5 causing the node with *absences* less than or equal to 1.5 to have the class label 0%. Hence, this abnormality is very likely due the nature of the samples in *student-mat.csv* dataset.

6. Evaluation

To evaluate the built decision tree classification model, 100 student instances are chosen randomly with replacements from the dataset *student-mat.csv* to produce the test dataset. The size of the testing set is approximately 25% of the original dataset.
StudentPerformance.py uses a *while* loop to build 10 decision tree classification models for the purpose of evaluation. Since training instances are selected at random to build each decision tree classifier, the internal structure of each built classification model varies. However, the predictability of each built model is quite consistent. Given a test instance, *StudentPerformance.py* is designed to output a multi-class prediction which is a grade from 0 to 20. In addition, *StudentPerformance.py* is modified, so that it can also output a binary prediction, that is, 0 (failing Math) or 1 (passing Math), given a test student instance. The results of 10 tests of *StudentPerformance.py* is shown in Figure 7.



```
Test5
_____ Multi-class predictions _____
Accuracy = 59%
Average error = 3 grades
_____ Binary-class predictions _____
Accuracy = 100%
Test6
_____ Multi-class predictions _____
Accuracy = 68%
Average error = 5 grades
_____ Binary-class predictions _____
Accuracy = 100%
Test7
_____ Multi-class predictions _____
Accuracy = 63%
Average error = 4 grades
_____ Binary-class predictions _____
Accuracy = 100%
Test8
_____ Multi-class predictions _____
Accuracy = 61%
Average error = 3 grades
_____ Binary-class predictions _____
Accuracy = 100%
Test9
_____ Multi-class predictions _____
Accuracy = 67%
Average error = 4 grades
_____ Binary-class predictions _____
Accuracy = 100%
Test10
_____ Multi-class predictions _____
Accuracy = 64%
Average error = 5 grades
_____ Binary-class predictions _____
Accuracy = 100%
=========================
_____ Multi-class predictions summary_____
Average accuracy = 61%
Average average error = 4 grades
_____ Binary-class predictions summary_____
Average accuracy = 100%
```

Figure 7

For each test, a decision tree classification model is built and tested with 100 random test instances from the original dataset. For both multi-class predictions and binary predictions, accuracy is defined as the number of correct predictions divided by the total number of predictions. The average error for multi-class predictions is defined as the sum of the absolute values of the differences of the true student grade and the predicted student grade in an incorrect prediction divided by the number of total incorrect predictions. On average, the accuracy of multi-class predictions is approximately 60%, and each incorrect prediction has an error of 4 grades. This implies that if a student instance is selected at random, the decision tree classification model is able to predict the student's final grade in Mathematics correctly with the probability of 0.6, and for each incorrect prediction on average, the predicted grade could be 4 grades more or 4 grades less than the actual grade. The average error in an incorrect prediction is therefore significant, since 4 grades is equivalent to 20% of the final grade. However, the decision tree classification model has excellent performance in binary predictions with almost 100% accuracy. That is, given a test instance from the test set, the classification model is able to predict whether the student passes or fails Mathematics with almost certainty.

7. Conclusion

In order for *DecisionTreeClassifier* in *Scikit-Learn* to build classification models from *student-mat.csv*, a substantial amount of work needs to be dedicated in data preprocessing. The decision tree classification model built by *StudentPerformance.py* does not perform well in predicting the final grade from 0 - 20, given a test instance. This may due the lack of data, since *student-mat.csv* only contains 395 student records, or it may be the lack of relevant attributes, since most of the attributes in *student-mat.csv* are demographic, social, and school related features, which are non-cognitive factors, and it is shown that mathematics skills are closely linked to cognitive factors [7]. Nevertheless, the classification model performs extremely well in predicting whether a student passes or fails the Mathematics subject. This may implies that although building a classification model based on demographic, social, and school related attributes is not effective enough to predict the exact grade of a student in Mathematics, predicting whether the student passes the academic subject is quite sufficient.

8. Future Work

As stated in the Conclusion section, the decision tree classification model built by *StudentPerformance.py* does not perform well in predicting the final grade from 0 - 20, given a test instance. This may due to the lack of instances or the lack of relevant attributes in the *student-mat.csv* dataset. Therefore, collecting more student data instances with more potential relevant attributes is a possible option for the future work of this project. Another possible future work is to build the classification model based on different types of classifiers, such as the Naive Bayes classifier or the linear regression classifier.

9. Acknowledgements

This project report is part of the Winter 2018 SENG 474 Data Mining course project. We thank Dr. Alex Thomo for his guidance during the course.

10. References

[1] Al-Radaideh, Q., Al-Shawakfa, E. & Al-Najjar, M. (2006), Mining Student Data Using Decision Trees, International Arab Conference on Information Technology (ACIT'2006), Yarmouk University.

[2] P. Cortez and A. Silva. Using Data Mining to Predict Secondary School Student Performance. In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7.

[3] Kumar S. A. & Vijayalakshmi M. N. (2011), Efficiency of Decision Trees in Predicting Student's Academic Performance, First International Conference on Computer Science, Engineering and Applications, CS and IT 02, Dubai, pp. 335-343.

[4] Ma Y., Liu B.; Wong C., Yu P., and Lee S., (2000), Targeting the right students using data mining. In Proc. of 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Boston, USA, 457–464.

[5] Superby J.F., Vandamme J.P. & Meskens N. (2006), Determination of Factors Influencing the Achievement of the First year University Students using Data Mining Methods, Proceedings of the 8th international conference on intelligent tutoring systems, Educational Data Mining Workshop, (ITS'06), Jhongali, pp. 37-44.

[6] https://archive.ics.uci.edu/ml/datasets/student+performance# .

[7] von Stumm, S., Hell, B., & Chamorro-Premuzic, T. (2011). The hungry mind: Intellectual curiosity is the third pillar of academic performance. Perspectives on Psychological Science, 6(6), 574-588.

[8] U.S. DEPARTMENT OF HEALTH AND HUMAN SERVICES. *Alcohol Research & Health,* "Alcoholic Brain Damage" (Vol. 27, No. 2, 2003)

[9] https://scikit-learn.org/stable/modules/tree.html