Predicting Student Performance

1. Introduction and Context

The data was obtained in a survey of secondary school students undertaking a Portuguese course at Gabriel Pereira and Mousinho da Silvera secondary schools. The survey contained 649 students, 33 attributes covering social, study and gender information and appeared unbiased after review.

The aim was to build a model using this data in order to predict whether a student was likely to have an above or below average performance. This would help teachers find students who may benefit from additional support as well as indicating which factors have a positive or negative impact on performance.

2. Data Preparation

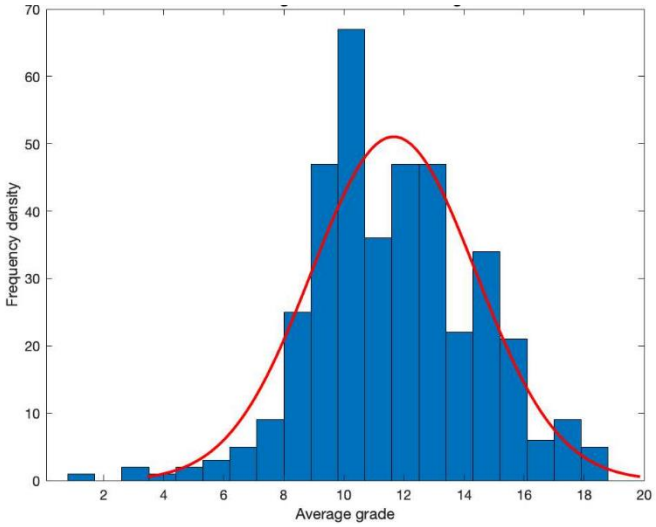

Figure 1: Grade average distribution of training data (see Appendix A for code)

In order to avoid overfitting the model and it learning irrelevant details, the data was split into three parts; 80% training data, 10% validation data and 10% test data (see Appendix B). The model was first developed on the training set before iteratively assessing its accuracy on the validation set. The test data was only used at the end of the development process to evaluate how successful the final model was on unseen data.

Categoric variables, such as gender, were converted to binary values as the model could only interpret the data in this format.

A column was added to calculate the students' mean scores across the three termly tests. An average of these results was used as the boundary for whether a student was classed as an above or below average achiever.

A histogram (Figure 1) was used to assess the distribution of the training data. The normal distribution showed that undersampling the data was not necessary. Furthermore, the boundary for whether a student was achieving at an above or below average level was placed at the mean, splitting the students equally.

3. Creating the model

A decision tree was used to select the features to be used in the final model. It was a greedy model which made locally optimum choices, anticipating that this would result in close to the global optima. Figure 2 assesses the impact of depth on accuracy.

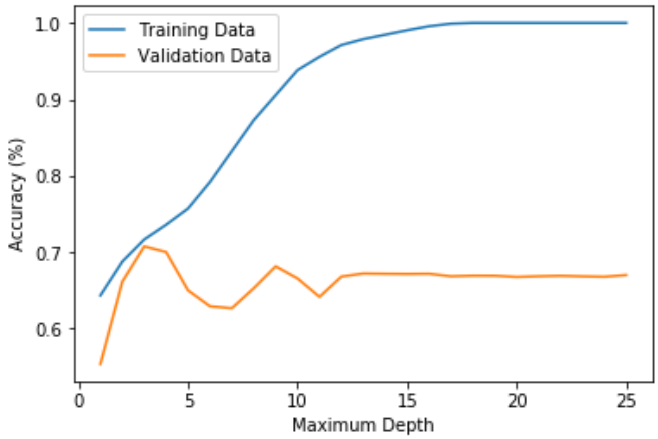

Figure 2: Impact of depth on the accuracy of the model (an average of 1000 repeated runs). See Appendix C

The training data accuracy continued to increase as more features were added. However, the validation data peaked between 3 and 4 features and then plateaued. This was due to the model overfitting to the training set. A depth of 3 was therefore selected for the predictive model.

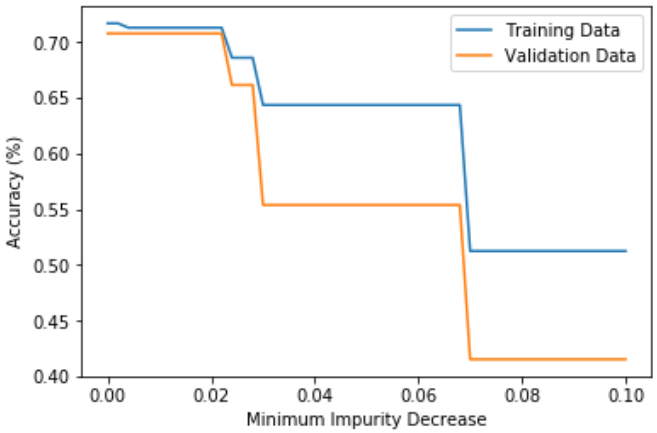

Figure 3: Impact of MID with a fixed maximum depth of 3 (an average of 1000 repeat runs). See Appendix C

Figure 4: Impact of depth on the accuracy of the model (an average of 1000 repeat runs). See Appendix C

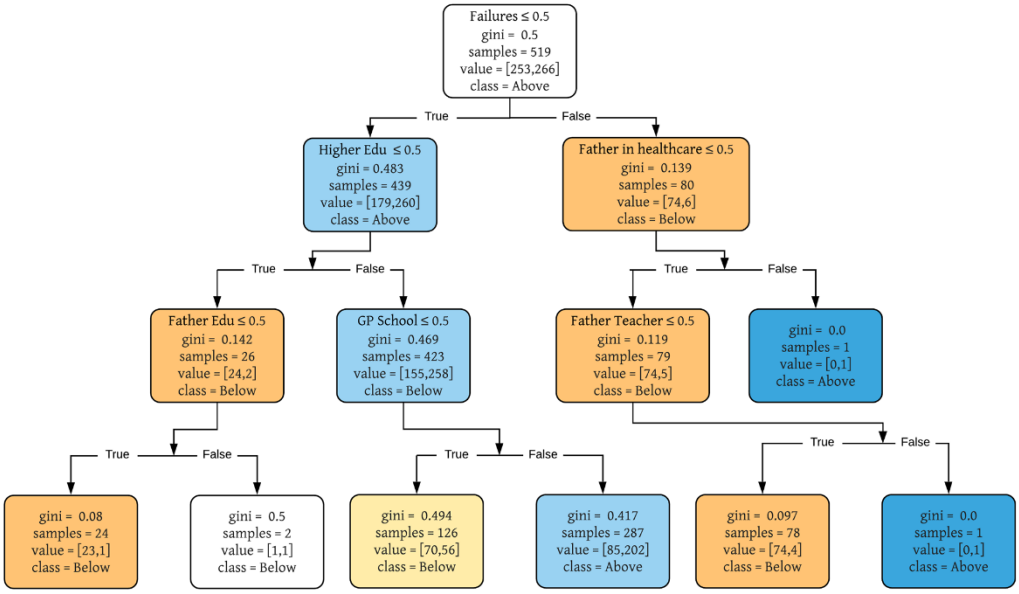

Figure 3 shows the effect of increasing the Minimum Impurity Decrease with a constant maximum depth of 3. The approach was to move down the tree, selecting the lowest gini number each time. However, if the reduction in the gini number was less than the MID, the split was not made. The highest accuracy was achieved on both data sets when the MID was 0 and this was therefore the selected value for the model.

4. Results and Analysis

Based on their gini values, the decision tree (see Figure 4) included the following features: whether a student had failed in the past, their desire to continue into higher education, whether their father is a teacher, whether their father completed secondary school, which of the two schools the pupil attends, and whether their father's job is in healthcare.

The training data was first split according to whether a student had failed previously or not. If a student had not failed previously, the second split was based on their desire to go into higher education. If this was false the group was split again, based on if their father completed secondary education. If not, the student was almost certainly a below average performer, with a gini of 0.08. These classifications continued down each branch of the tree, with the final ginis varying between 0 and 0.5, showing two leaf nodes to be pure and the rest impure.

The accuracy of the model on the test data was only 1% lower than on the validation data (see Appendix D). This showed that the model was not overfitted to the training data. The most significant feature was whether a pupil had failed in the past. Arguably this was an obvious result and would not provide new insight to teachers. However, the second most influential feature, the desire to continue to higher education, was more insightful. If this model were to be useful to other schools, the feature containing which school a pupil attends would have to be omitted.

There were less false positives (8) in the confusion matrix than false negatives (12). The aim was to minimise the number of false positives to ensure struggling pupils were not overlooked. This ratio confirmed the decision to not adjust the threshold value.

The model could be improved by using a larger data set from a wider range of schools and adding more potentially insightful questions to the survey. The model's greedy method may also not have resulted in a globally optimum model.

The Random Forest Model (RFM) was used to assess the performance of the decision tree (see Appendix E). It is a supervised classification algorithm that works by combining the results of randomized decision trees to determine the most accurate final prediction.

The 6 most influential features determined by the RFM were identical to those selected by the decision tree, verifying the trained model (see Appendix F).

The overall accuracy of the RFM on the test data was 6% higher than the decision tree. This result was expected as a decision tree tends to overfit.

Confusion Matrix	[[18 12] [8 27]]
Accuracy	0.6923
Precision	0.6923
Recall	0.7714

Table 1: Results of decision tree model on test data

APPENDIX A - CHECKING DATA DISTRIBUTION (MATLAB)

```
Data = csvread('trainnormalcheck.csv');
column1 = 'Average grades';

%boxplot for descriptive analysis
figure()
boxplot(Data, 'labels', {column1})
title('Average grade distribution of training data')
xlabel('average grade')
ylabel('frequency density')
hold on
plot(mean(Data),'dg')
hold off

%histogram to check for gaussian distribution
figure()
histfit(Data(:,1))
xlabel('Average grade')
ylabel('Frequency density')
title('Grade average distribution of training data')
```

APPENDIX B – DATA PREPARATION

```
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
import warnings

warnings.filterwarnings("ignore")

StudentRaw= pd.read_csv('student-por.csv')
Student = pd.get_dummies(StudentRaw,
columns=['school','sex','address','famsize','Pstatus','Medu','Fedu','Mjob','Fjob','reason
','guardian','traveltime','studytime','famrel','freetime','goout','Dalc','Walc',])

mean = Student['Average'].mean()
Student['Good'] = Student['Average'] > mean
Student['Good'] = Student['Good'].astype(int)
Student = Student.drop(columns=['Average','G1','G2','G3'])

train, other = train_test_split(Student, test_size=0.2, random_state=0)
validation, test = train_test_split(other, test_size=0.5, random_state=0)

y_train = train['Good']
x_train = train.drop(columns=['Good'])

y_val = validation['Good']
x_val = validation.drop(columns=['Good'])

y_test = test['Good']
x_test = test.drop(columns=['Good'])
```

APPENDIX C – CREATING THE MODEL

TESTING THE IDEAL MAX DEPTH

```
lista = list(range(1,26))
listtrain = []
listval = []
count = list(range(1,1001))
number = 0

for x in lista:
    temptrain = []
    temp = []
    for y in count:
        model = tree.DecisionTreeClassifier(max_depth=x, min_impurity_decrease = 0)
        model.fit(x_train,y_train)
        y_pred = model.predict(x_val)
        y_pred_train = model.predict(x_train)
        acc_train = accuracy_score(y_train,y_pred_train)
        acc = accuracy_score(y_val,y_pred)
        prec = precision_score(y_val,y_pred)
        rec = recall_score(y_val,y_pred)
        temptrain.append(acc_train)
        temp.append(acc)
        if y == 1000:
            meantrain = sum(temptrain)/len(temptrain)
            meanval = sum(temp)/len(temp)
            listtrain.append(meantrain)
            listval.append(meanval)
            number = number + 1
            print (number)

plot.plot(lista,listtrain)
plot.plot(lista,listval)
```

TESTING THE IDEAL MID

```
lista =  np.linspace(0,0.1,51)
listtrain = []
listval = []
count = list(range(1,101))
number = 0

for x in lista:
    temptrain = []
    temp = []
    for y in count:
        model = tree.DecisionTreeClassifier(max_depth=3, min_impurity_decrease = x)
        model.fit(x_train,y_train)
        y_pred = model.predict(x_val)
        y_pred_train = model.predict(x_train)
        acc_train = accuracy_score(y_train,y_pred_train)
        acc = accuracy_score(y_val,y_pred)
        prec = precision_score(y_val,y_pred)
        rec = recall_score(y_val,y_pred)
        temptrain.append(acc_train)
        temp.append(acc)
        if y == 100:
            meantrain = sum(temptrain)/len(temptrain)
            meanval = sum(temp)/len(temp)
            listtrain.append(meantrain)
```

```
            listval.append(meanval)
            number = number + 1
            print (number)

plot.plot(lista,listtrain)
plot.plot(lista,listval)
```

CREATING THE FINAL MODEL

```
model = tree.DecisionTreeClassifier(max_depth=3, min_impurity_decrease = 0)
model.fit(x_train,y_train)

y_pred_val = model.predict(x_val)
acc_val = accuracy_score(y_val,y_pred_val)
prec_val = precision_score(y_val,y_pred_val)
rec_val = recall_score(y_val,y_pred_val)

y_pred_test = model.predict(x_test)
acc_test = accuracy_score(y_test,y_pred_test)
prec_test = precision_score(y_test,y_pred_test)
rec_test = recall_score(y_test,y_pred_test)
```

APPENDIX D – RESULTS

```
print ('--Validation--')
print ('Precision: {}'.format(prec_val))
print ('Accuracy: {}'.format(acc_val))
print ('Recall: {}'.format(rec_val))
conf_mat_val = confusion_matrix(y_val,y_pred_val)
print (conf_mat_val)

print ('--Test--')
print ('Precision: {}'.format(prec_test))
print ('Accuracy: {}'.format(acc_test))
print ('Recall: {}'.format(rec_test))
conf_mat_test = confusion_matrix(y_test,y_pred_test)
print (conf_mat_test)


with open("model.txt", "w") as f:
    f = tree.export_graphviz(model, out_file=f)
```

APPENDIX E – RANDOM FOREST MODEL

```
StudentRaw= pd.read_csv('student-por.csv')
Student = pd.get_dummies(StudentRaw,
columns=['school','sex','address','famsize','Pstatus','Medu','Fedu','Mjob','Fjob','reason
','guardian','traveltime','studytime','famrel','freetime','goout','Dalc','Walc',])

mean = Student['Average'].mean()
Student['Good'] = Student['Average'] > mean
Student['Good'] = Student['Good'].astype(int)
Student = Student.drop(columns=['Average','G1','G2','G3'])

train, other = train_test_split(Student, test_size=0.2, random_state=0)
validation, test = train_test_split(other, test_size=0.5, random_state=0)
```

```
y_train = train['Good']
x_train = train.drop(columns=['Good'])

y_val = validation['Good']
x_val = validation.drop(columns=['Good'])

y_test = test['Good']
x_test = test.drop(columns=['Good'])

model = RandomForestClassifier(max_depth = 8,min_samples_split = 10)
model = model.fit(x_train,y_train)

y_pred_val = model.predict(x_val)
acc_val = accuracy_score(y_val,y_pred_val)
prec_val = precision_score(y_val,y_pred_val)
rec_val = recall_score(y_val,y_pred_val)

y_pred_test = model.predict(x_test)
acc_test = accuracy_score(y_test,y_pred_test)
prec_test = precision_score(y_test,y_pred_test)
rec_test = recall_score(y_test,y_pred_test)

print ('--Validation--')
print ('Precision: {}'.format(prec_val))
print ('Accuracy: {}'.format(acc_val))
print ('Recall: {}'.format(rec_val))
conf_mat_val = confusion_matrix(y_val,y_pred_val)
print (conf_mat_val)

print ('--Test--')
print ('Precision: {}'.format(prec_test))
print ('Accuracy: {}'.format(acc_test))
print ('Recall: {}'.format(rec_test))
conf_mat_test = confusion_matrix(y_test,y_pred_test)
print (conf_mat_test)
```

APPENDIX F – RANDOM FOREST RESULTS

```
--Validation--
Precision: 0.6875
Accuracy: 0.7692307692307693
Recall: 0.8148148148148148

--Test--
Precision: 0.7567567567567568
Accuracy: 0.7538461538461538
Recall: 0.8
```

TOP SIX MOST INFLUENTIAL FEATURES

```
[(‘failures’, 0.5407927380284007),
('higher', 0.21922809848931032),
('school_MS', 0.175969451254606),
('Fjob_health', 0.026544579011408137),
('Fjob_teacher', 0.025880964536122996),
('Fedu_3', 0.011584168680152)...]
```