Machine learning assignment 3 report

Muhd hilman bin rozaini (P121535)

Data pre-processing

Firstly, the data consist of strings and we convert all string values into binary or nominal values so that we can perform classification using decision tree method. The converting values are as figure below, where some variables are assigned to 1 for “yes”, and 0 for “no”, and some unique strings are classified into binary values respectively to represent the dataset in integer data type.

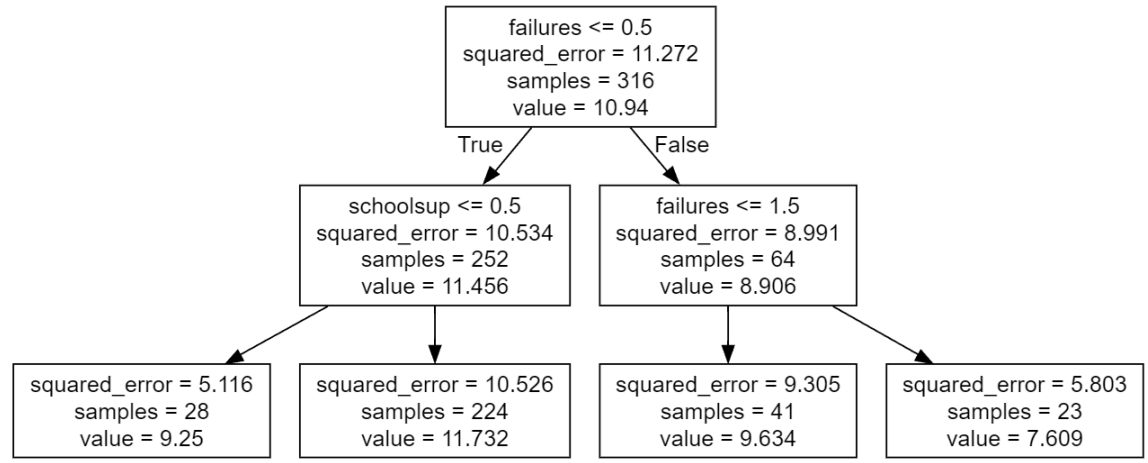
```
#converting all string into binary or nominal value

df_mat['school'] = df_mat.school.map({'GP':0, 'MS':1})
df_mat['sex'] = df_mat.sex.map({'F':0, 'M':1})
df_mat['address'] = df_mat.address.map({'U':0, 'R':1})
df_mat['famsize'] = df_mat.famsize.map({'LE3':0, 'GT3':1})
df_mat['Pstatus'] = df_mat.Pstatus.map({'T':0, 'A':1})
df_mat.Mjob = pd.factorize(df_mat.Mjob)[0]   # assigning numeric value to each categorical value in the column
df_mat.Fjob = pd.factorize(df_mat.Fjob)[0]  # assigning numeric value to each categorical value in the column
df_mat.reason = pd.factorize(df_mat.reason)[0]
df_mat.guardian = pd.factorize(df_mat.guardian)[0]
df_mat['schoolsup'] = df_mat.schoolsup.map({'no':1, 'yes':0})
df_mat['famsup'] = df_mat.famsup.map({'no':1, 'yes':0})
df_mat['paid'] = df_mat.paid.map({'no':1, 'yes':0})
df_mat['activities'] = df_mat.activities.map({'no':1, 'yes':0})
df_mat['nursery'] = df_mat.nursery.map({'no':1, 'yes':0})
df_mat['higher'] = df_mat.higher.map({'no':1, 'yes':0})
df_mat['internet'] = df_mat.internet.map({'no':1, 'yes':0})
df_mat['romantic'] = df_mat.romantic.map({'no':1, 'yes':0})


df_mat.head()
```

Next, the dataset is checked for any NULL values, and any NULL values present will be filled with mode in the variable. The reason for using mode is because mode represents the most common value in a column, and we can preserve the existing distribution of the data by using mode. This can further help to maintain the overall patterns and characteristic of the dataset as it minimises the distortion of the original data.

Furthermore, the mode is appropriate to fill NA values in categorical values because it is able to help maintain the categorical nature of the variable.

Overall, using the mode to fill missing values can be a reasonable approach, especially for categorical variables, as it helps maintain the integrity of the data distribution and is easy to implement.

Q1: Regression tree to predict variable G1




```
In [115]: importance_scores = pd.Series(regr_tree_mat.feature_importances_, index=X.columns)
          importance_scores
```

Fedu	0.000000
Mjob	0.000000
Fjob	0.000000
reason	0.000000
guardian	0.000000
traveltime	0.000000
studytime	0.000000
failures	0.718995
schoolsup	0.281005
famsup	0.000000
paid	0.000000
activities	0.000000
nursery	0.000000
higher	0.000000
internet	0.000000
romantic	0.000000
famrel	0.000000
freetime	0.000000
goout	0.000000

The dept of the regression tree is 2. From the dataset, we can see that the most important indicator to predict G1 is “failures” and “schoolsup” where the importance score is 0.719 and 0.281 respectively.

We can see that output with failures <= 0.5 is classified as 0 and shift to the left, and any values that failures are more than 0.5 is classified as 1 and shift to the right. It means that non-failures of predicted G1 grade is on the left node from the parent node. Thus, for True statement, it showed that 252 students has not failed G1 due to school support, and 64 students has predicted to fail G1 grade.

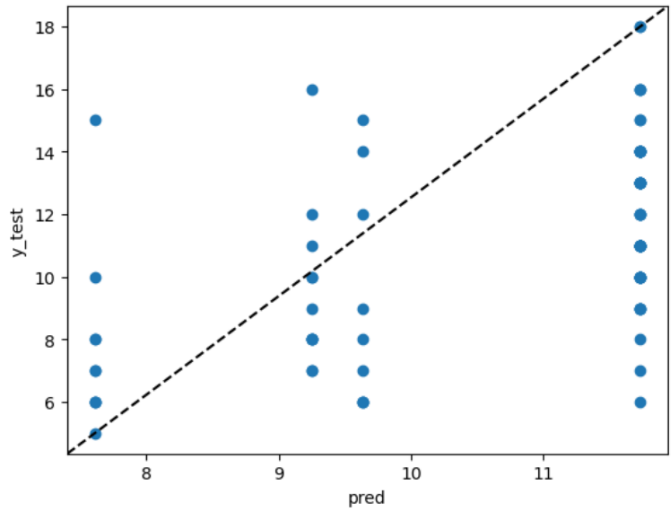
Next, the variable “failures” measures the number of past class failure. The tree indicates that the lower number of past class failures correspond to the extra school education support (schoolsup). The regression tree predicts that the value of G1 grade is 11.732, which can be rounded to approximately 12 when there is extra educational school support (Schoolsup != 0).

In contrast, the predicted grade G1 is predicted to achieve a value of 9.25, rounding to 9, when there is no extra school educational support (schoolsup = 0).

```
pred = regr_tree_mat.predict(X_test)

plt.scatter(pred,
            y_test,
            label = 'G1')

plt.plot([0, 1],
         [0, 1],
         '--k',
         transform = plt.gca().transAxes)

plt.xlabel('pred')
plt.ylabel('y_test')

mean_squared_error(y_test, pred)
```

```
7.338965243637666
```

Furthermore, the MSE for the model is 7.33 which is low, indicating that the predicted values and the actual values of the target variable does not have large difference. Thus, the lower MSE indicates that the model is performing relatively well.



The test set MSE associated with the regression tree is 7.34. The square root of the MSE is therefore around 2.71, indicating that this model leads to test predictions that are within around 2.71 of the true predicted G1 grade result.

Q2: Build a classification tree to classify the G2T variable

```
# Build a classification tree to classify the G2T variable
classifier = DecisionTreeClassifier(max_depth = 7)  #limiting depth = 7,
classifier.fit(X_train, y_train)
classifier.score(X_train, y_train)
```

```
0.7310126582278481
```

The classification tree has 7 layer depth because with 7th layer depth, it produces an accuracy of 73.1% which is generally a good model performance, with gini value of 0.698.

According to the classification tree image attached, it appears that the most important indicator of G2T grade prediction is number of past class failures.

From the classification tree, we can observe that number of absences are the main indicator that there is a predicted failure grade when predicting G2T grade.

In contrast, when statement is true that there is no failure (failure = 0), the most significant reason is due to reason to choose the school, which can consist of school reputation, student course preferences, and other factors from the reason.

Further in the true statement, we can see that when there is weak reason (reason <= 0.5) for choosing the school, it is usually due to extra paid classes within the course subject such as math subject so that the student will not be predicted to fail G2T grade. However, when there is a strong reason (reason > 0.5), it is usually the student's guardian made the reason.

As there are 7 layer depth in the classification tree, any high gini value will result to more classification of variables as there still exist some impurity variables behind the specific node that might lead to another indicator that provides a better insight in predicting G2T grades.

In overall, we can see that there are 252 students that is not predicted to fail the G2T grade prediction due to multiple reason for choosing the school, but there are 64 students predicted to fail G2T grade prediction mainly due to absences in school.

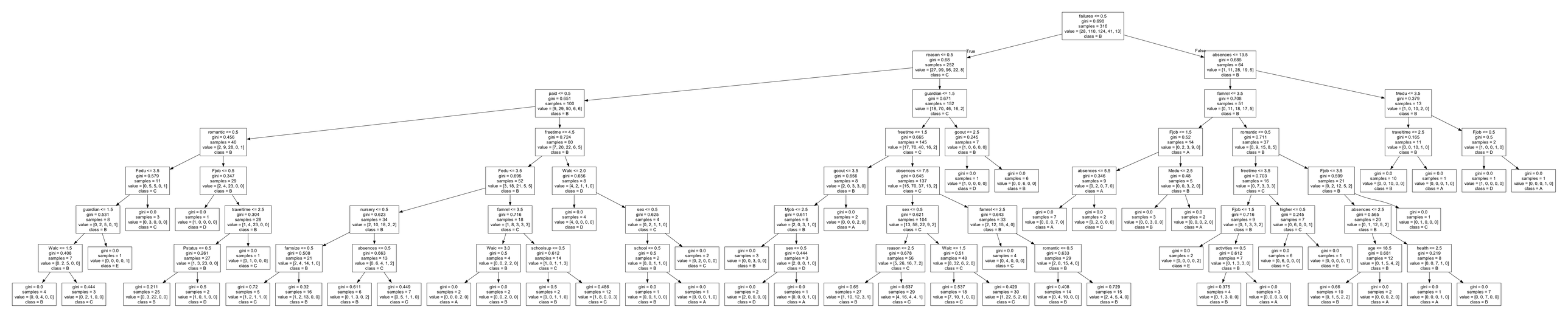
Next, we can evaluate the performance on the model by testing with test data as figure below:

```
In [149]: # to evaluate tree performance on test data
          pred = classifier.predict(X_test)
          pred
          cm = pd.DataFrame(confusion_matrix(y_test, pred).T,
                            index = ['A', 'B', 'C', 'D', 'E'],
                            columns = ['A', 'B', 'C', 'D', 'E'])
          print(cm)
          # (36+22)/80 = 0.745
```

```
   A   B   C  D  E
A  0   1   0  0  0
B  2  12  17  0  0
C  3  12  18  7  0
D  0   0   4  2  0
E  0   0   0  1  0
```

From here, we can observe that the model is making a 40.1% accuracy in predicting the G2T value. But it is the most optimum model as compared when the depth increases because the accuracy remains similar or that the differences is very insignificant.

In overall, the model provides a generally good performance but the accuracy is rather below average of around 40% which mainly due to high impurities in the nodes for 7^{th} depth tree. However, it is to take note that higher the depth does not assure better accuracy as there is also a tendency of overfitting the tree.


failures <= 0.5
gini = 0.698
samples = 316
value = [28, 110, 124, 41, 13]
class = B
True
False
reason <= 0.5
gini = 0.68
samples = 252
value = [27, 99, 96, 22, 8]
class = C
absences <= 13.5
gini = 0.685
samples = 64
value = [1, 11, 28, 19, 5]
class = B
paid <= 0.5
gini = 0.651
samples = 100
value = [9, 29, 50, 6, 6]
class = B
guardian <= 1.5
gini = 0.671
samples = 152
value = [18, 70, 46, 16, 2]
class = C
famrel <= 3.5
gini = 0.708
samples = 51
value = [0, 11, 18, 17, 5]
class = B
Medu <= 3.5
gini = 0.379
samples = 13
value = [1, 0, 10, 2, 0]
class = B
romantic <= 0.5
gini = 0.456
samples = 40
value = [2, 9, 28, 0, 1]
class = B
freetime <= 4.5
gini = 0.724
samples = 60
value = [7, 20, 22, 6, 5]
class = B
freetime <= 1.5
gini = 0.665
samples = 145
value = [17, 70, 40, 16, 2]
class = C
goout <= 2.5
gini = 0.245
samples = 7
value = [1, 0, 6, 0, 0]
class = B
Fjob <= 1.5
gini = 0.52
samples = 14
value = [0, 2, 3, 9, 0]
class = A
romantic <= 0.5
gini = 0.711
samples = 37
value = [0, 9, 15, 8, 5]
class = B
traveltime <= 2.5
gini = 0.245
samples = 11
value = [1, 0, 0, 10, 0]
class = B
Fjob <= 0.5
gini = 0.5
samples = 2
value = [1, 0, 0, 1, 0]
class = D
Fedu <= 3.5
gini = 0.579
samples = 11
value = [0, 5, 5, 0, 1]
class = C
Fjob <= 0.5
gini = 0.347
samples = 29
value = [2, 4, 23, 0, 0]
class = C
Fedu <= 3.5
gini = 0.695
samples = 52
value = [3, 18, 21, 5, 5]
class = B
Walc <= 2.0
gini = 0.656
samples = 8
value = [4, 2, 1, 1, 0]
class = D
goout <= 3.5
gini = 0.656
samples = 8
value = [2, 0, 3, 3, 0]
class = C
absences <= 7.5
gini = 0.645
samples = 137
value = [15, 70, 37, 13, 2]
class = C
absences <= 5.5
gini = 0.346
samples = 9
value = [0, 2, 0, 7, 0]
class = A
Medu <= 2.5
gini = 0.48
samples = 5
value = [0, 0, 3, 2, 0]
class = B
freetime <= 3.5
gini = 0.703
samples = 16
value = [0, 7, 3, 3, 3]
class = C
Fjob <= 3.5
gini = 0.599
samples = 21
value = [0, 2, 12, 5, 2]
class = B
age <= 18.5
gini = 0.681
samples = 12
value = [0, 1, 5, 4, 2]
class = B
health <= 2.5
gini = 0.219
samples = 8
value = [0, 0, 7, 1, 0]
class = B
guardian <= 1.5
gini = 0.531
samples = 8
value = [0, 2, 5, 0, 1]
class = B
gini = 0.0
samples = 3
value = [0, 3, 0, 0, 0]
class = C
famrel <= 3.5
gini = 0.716
samples = 18
value = [1, 8, 3, 3, 3]
class = C
nursery <= 0.5
gini = 0.623
samples = 34
value = [2, 10, 18, 2, 2]
class = B
Mjob <= 2.5
gini = 0.611
samples = 6
value = [2, 0, 3, 1, 0]
class = B
gini = 0.0
samples = 2
value = [0, 0, 0, 2, 0]
class = A
sex <= 0.5
gini = 0.621
samples = 104
value = [13, 58, 22, 9, 2]
class = C
famrel <= 2.5
gini = 0.643
samples = 33
value = [2, 12, 15, 4, 0]
class = B
gini = 0.0
samples = 7
value = [0, 0, 0, 7, 0]
class = A
gini = 0.0
samples = 2
value = [0, 0, 0, 2, 0]
class = C
gini = 0.0
samples = 3
value = [0, 0, 3, 0, 0]
class = B
Medu <= 2.5
gini = 0.48
samples = 5
value = [0, 0, 3, 2, 0]
class = B
Fjob <= 1.5
gini = 0.716
samples = 9
value = [0, 1, 3, 3, 2]
class = B
higher <= 0.5
gini = 0.245
samples = 7
value = [0, 6, 0, 0, 1]
class = C
absences <= 2.5
gini = 0.565
samples = 20
value = [0, 1, 12, 5, 2]
class = B
gini = 0.0
samples = 1
value = [0, 1, 0, 0, 0]
class = C
gini = 0.0
samples = 10
value = [0, 0, 10, 0, 0]
class = B
gini = 0.0
samples = 1
value = [0, 0, 0, 1, 0]
class = A
gini = 0.0
samples = 1
value = [1, 0, 0, 0, 0]
class = D
gini = 0.0
samples = 1
value = [0, 0, 0, 1, 0]
class = A
Walc <= 1.5
gini = 0.408
samples = 7
value = [0, 2, 5, 0, 0]
class = B
gini = 0.0
samples = 1
value = [0, 0, 0, 0, 1]
class = E
Pstatus <= 0.5
gini = 0.201
samples = 27
value = [1, 3, 23, 0, 0]
class = B
gini = 0.0
samples = 1
value = [0, 1, 0, 0, 0]
class = C
famsize <= 0.5
gini = 0.508
samples = 21
value = [2, 4, 14, 1, 0]
class = B
absences <= 0.5
gini = 0.663
samples = 13
value = [0, 6, 4, 1, 2]
class = C
Walc <= 3.0
gini = 0.5
samples = 4
value = [0, 0, 2, 2, 0]
class = B
schoolsup <= 0.5
gini = 0.612
samples = 14
value = [1, 8, 1, 1, 3]
class = C
school <= 0.5
gini = 0.5
samples = 2
value = [0, 0, 1, 1, 0]
class = B
gini = 0.0
samples = 2
value = [0, 0, 0, 0, 2]
class = C
sex <= 0.5
gini = 0.444
samples = 3
value = [2, 0, 0, 1, 0]
class = D
reason <= 2.5
gini = 0.678
samples = 56
value = [5, 26, 16, 7, 2]
class = C
Walc <= 1.5
gini = 0.51
samples = 48
value = [8, 32, 6, 2, 0]
class = C
gini = 0.0
samples = 4
value = [0, 4, 0, 0, 0]
class = B
romantic <= 0.5
gini = 0.633
samples = 29
value = [2, 8, 15, 4, 0]
class = B
Fjob <= 1.5
gini = 0.612
samples = 9
value = [0, 1, 3, 3, 2]
class = B
gini = 0.0
samples = 6
value = [0, 6, 0, 0, 0]
class = C
gini = 0.0
samples = 1
value = [0, 0, 0, 0, 1]
class = E
gini = 0.66
samples = 10
value = [0, 1, 5, 2, 2]
class = B
gini = 0.0
samples = 2
value = [0, 0, 0, 2, 0]
class = A
gini = 0.0
samples = 8
value = [0, 0, 7, 1, 0]
class = B
gini = 0.0
samples = 1
value = [0, 0, 0, 1, 0]
class = A
gini = 0.0
samples = 7
value = [0, 0, 7, 0, 0]
class = B
gini = 0.0
samples = 4
value = [0, 0, 4, 0, 0]
class = B
gini = 0.444
samples = 3
value = [0, 2, 1, 0, 0]
class = C
gini = 0.211
samples = 25
value = [0, 3, 22, 0, 0]
class = B
gini = 0.5
samples = 2
value = [1, 0, 1, 0, 0]
class = D
gini = 0.72
samples = 5
value = [1, 2, 1, 1, 0]
class = C
gini = 0.32
samples = 16
value = [1, 2, 13, 0, 0]
class = B
gini = 0.611
samples = 6
value = [0, 1, 3, 0, 2]
class = B
gini = 0.449
samples = 7
value = [0, 5, 1, 1, 0]
class = C
gini = 0.0
samples = 2
value = [0, 0, 0, 2, 0]
class = A
gini = 0.0
samples = 2
value = [0, 0, 0, 2, 0]
class = B
gini = 0.5
samples = 2
value = [0, 0, 1, 1, 0]
class = B
gini = 0.486
samples = 12
value = [1, 8, 0, 0, 3]
class = C
gini = 0.0
samples = 1
value = [0, 0, 0, 1, 0]
class = A
gini = 0.0
samples = 1
value = [0, 0, 0, 1, 0]
class = A
gini = 0.637
samples = 29
value = [4, 14, 4, 4, 1]
class = C
gini = 0.537
samples = 18
value = [7, 10, 1, 0, 0]
class = C
gini = 0.429
samples = 30
value = [1, 22, 5, 2, 0]
class = C
gini = 0.408
samples = 14
value = [2, 4, 10, 0, 0]
class = B
gini = 0.729
samples = 15
value = [2, 4, 5, 4, 0]
class = B
gini = 0.375
samples = 4
value = [0, 1, 3, 0, 0]
class = B
gini = 0.0
samples = 3
value = [0, 0, 0, 3, 0]
class = A