UNIVERSITI KEBANGSAAN MALAYSIA

The National University of Malaysia

ASSIGNMENT 3

STQD6024: MACHINE LEARNING

SEMESTER 2 SESSION 2022/2023

MATRIX NO.: P119717

NAME: NUR MARDHIAH BT. ZULKHARI

LECTURER: DR MOHD AFTAR ABU BAKAR

By using a decision tree:

1. **Build a regression tree where the aim is to predict variable G1.**

This data approach student achievement in secondary education of two Portuguese schools the attributes include student grades, demographic, social and school related features and it was collected by using school reports and questionnaires. This analysis contain only to predict the grade of the students (G1, G2 and G3) for the Mathematics sucject. The data contain of 395 with 33 columns in total. The data have showed no missing values for all the variables. As can be seen in the pictures below, there are 33 variables with the mixtures of object and int64 data types.

```
student_math = pd.read_csv("assignment3/student-mat.csv",sep=";")
student_math
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	6	5	6	6
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	4	5	5	6
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7	8	10
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	2	15	14	15
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	4	6	10	10
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
390	MS	M	20	U	LE3	A	2	2	services	services	...	5	5	4	4	5	4	11	9	9	9
391	MS	M	17	U	LE3	T	3	1	services	services	...	2	4	5	3	4	2	3	14	16	16
392	MS	M	21	R	GT3	T	1	1	other	other	...	5	5	3	3	3	3	3	10	8	7
393	MS	M	18	R	LE3	T	3	2	services	other	...	4	4	1	3	4	5	0	11	12	10
394	MS	M	19	U	LE3	T	1	1	other	at_home	...	3	2	3	3	3	5	5	8	9	9

395 rows × 33 columns

```
student_math.info() #before
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      395 non-null    object
 1   sex         395 non-null    object
 2   age         395 non-null    int64
 3   address     395 non-null    object
 4   famsize     395 non-null    object
 5   Pstatus     395 non-null    object
 6   Medu        395 non-null    int64
 7   Fedu        395 non-null    int64
 8   Mjob        395 non-null    object
 9   Fjob        395 non-null    object
 10  reason      395 non-null    object
 11  guardian    395 non-null    object
 12  traveltime  395 non-null    int64
 13  studytime   395 non-null    int64
 14  failures    395 non-null    int64
```

```
 15  schoolsup   395 non-null    object
 16  famsup      395 non-null    object
 17  paid        395 non-null    object
 18  activities  395 non-null    object
 19  nursery     395 non-null    object
 20  higher      395 non-null    object
 21  internet    395 non-null    object
 22  romantic    395 non-null    object
 23  famrel      395 non-null    int64
 24  freetime    395 non-null    int64
 25  goout       395 non-null    int64
 26  Dalc        395 non-null    int64
 27  Walc        395 non-null    int64
 28  health      395 non-null    int64
 29  absences    395 non-null    int64
 30  G1          395 non-null    int64
 31  G2          395 non-null    int64
 32  G3          395 non-null    int64
```

In addition, below are the list of attributes for both student-mat.csv (Math course) datasets:

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 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education)

8 Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ 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)

Below varibles are the grades that related with the course subject Math :

31 G1 - first period grade (numeric: from 0 to 20)

31 G2 - second period grade (numeric: from 0 to 20)

32 G3 - final grade (numeric: from 0 to 20, output target)

In question 1, the question ask to specify which variables that important to predict the grades for G1. There are a few pre-processing being done before doing the analysis including checking the unique value in G1 with some others independent variables. In pandas, a categorical variable is a type of data that represents a finite set of possible values, often referred to as categories or levels which the object data types can use in categorical variables. Step 1 is to convert the data types *int64* into *object* column to a categorical variable as per below. The variables that included are *Medu, Fedu, traveltime, studytime, famrel, freetime, goout, Dalc, Walc and health.*

```
g1_studentmath['Medu'] = g1_studentmath['Medu'].astype('object')
g1_studentmath['Fedu'] = g1_studentmath['Fedu'].astype('object')
g1_studentmath['traveltime'] = g1_studentmath['traveltime'].astype('object')
g1_studentmath['studytime'] = g1_studentmath['studytime'].astype('object')
g1_studentmath['famrel'] = g1_studentmath['famrel'].astype('object')
g1_studentmath['freetime'] = g1_studentmath['freetime'].astype('object')
g1_studentmath['goout'] = g1_studentmath['goout'].astype('object')
g1_studentmath['Dalc'] = g1_studentmath['Dalc'].astype('object')
g1_studentmath['Walc'] = g1_studentmath['Walc'].astype('object')
g1_studentmath['health'] = g1_studentmath['health'].astype('object')
```

In step 2, to define the independent variable with object data types columns to encode. Step 3 is to create dummy variables for the independent variable columns as per below. The purpose of dummies for independents variable columns is to handle the categorical data with qualitative attributed to transform into binary (0 or 1) numerical variables that allowed to be incorporated into the analysis.

```
# STEP 2: Define the independent variable(object dtypes) columns to encode
#

independent_variables_cate = ['school', 'sex','address','famsize','Pstatus','reason','guardian','Mjob','Fjob',
                              'schoolsup','famsup','paid','activities','nursery','higher','internet',
                              'romantic','Medu', 'Fedu', 'traveltime', 'studytime', 'famrel',
                              'freetime', 'goout', 'Dalc', 'Walc', 'health']

# STEP 3: Create dummy variables for the independent variable columns
g1_studentmath_new = pd.get_dummies(g1_studentmath,columns=independent_variables_cate)
```

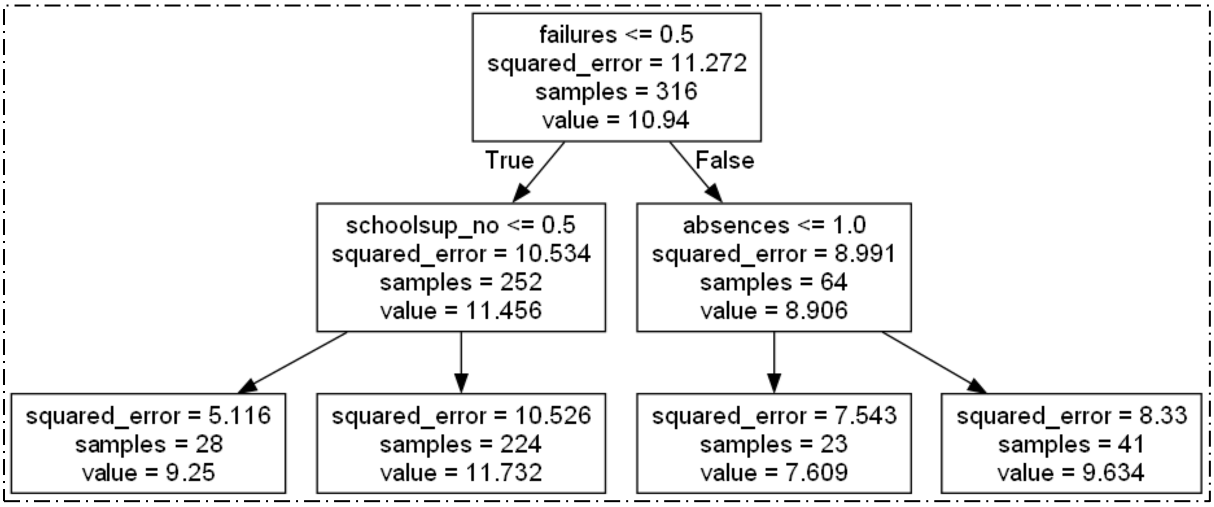
```
g1_studentmath_new.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 97 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   age              395 non-null    int64
 1   failures         395 non-null    int64
 2   absences         395 non-null    int64
 3   G1               395 non-null    int64
 4   G2               395 non-null    int64
 5   G3               395 non-null    int64
 6   school_GP        395 non-null    uint8
 7   school_MS        395 non-null    uint8
 8   sex_F            395 non-null    uint8
 9   sex_M            395 non-null    uint8
 10  address_R        395 non-null    uint8
 11  address_U        395 non-null    uint8
```

Next is step 4: which to define the dependent variable (y) and step 5 to define the independent variable (x) which the drop method that removes specified columns from the DataFrame which is ['G1', 'G2', 'G3'] with using of axis=1 represents columns. X now holds the modified DataFrame without the specified columns which useful to separate the features independent variables (x) from the target variable dependent variable (y). In stp 6 is to split data set into training and testing sets where the train size is 80:20 in ratio with random state equal to 1.

```
y = g1_studentmath_new['G1'] #step4
X = g1_studentmath_new.drop(['G1', 'G2', 'G3'], axis=1) #step5
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8, random_state = 1) #step6
```

For step 7, is where to get the best maximum depth and best mean square error that use the loop function that iterate from 1 till n-1 which set by n=10, create the Decision Tree Regressor to train model that used to make predictions on the testing set (X_test), evaluate the model on the testing set and check the current maximum depth gives a better perfomance. From the function, the **Best Maximum Depth is 2** and **best MSE is 7.459464.**

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

n= 10
best_max_depth = []
best_mse = float('inf') #initialized as positive infinity

# Iterate over maximum depth values
# the loop iterates from 1 to n-1, which in this case is 1 to 9
for max_depth in range (1,n):
    # Create the Decision Tree Regressor
    regressor = DecisionTreeRegressor(max_depth=max_depth,random_state=1)

    # Train the model
    #trained model is used to make predictions on the testing set (X_test)
    #calculated by comparing the predicted values (y_pred) with the actual target values (y_test).
    regressor.fit(X_train, y_train)

    # Evaluate the model on the testing set
    y_pred = regressor.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)

    # Check if current maximum depth gives better performance
    #if mse < best_mse,it means that the current maximum depth (max_depth) is giving better performance
    if mse < best_mse:
        best_mse = mse
        best_max_depth = max_depth

print("Best Maximum Depth:", best_max_depth)
print("Best Mean Squared Error:", best_mse)

Best Maximum Depth: 2
Best Mean Squared Error: 7.4594640215069905
```

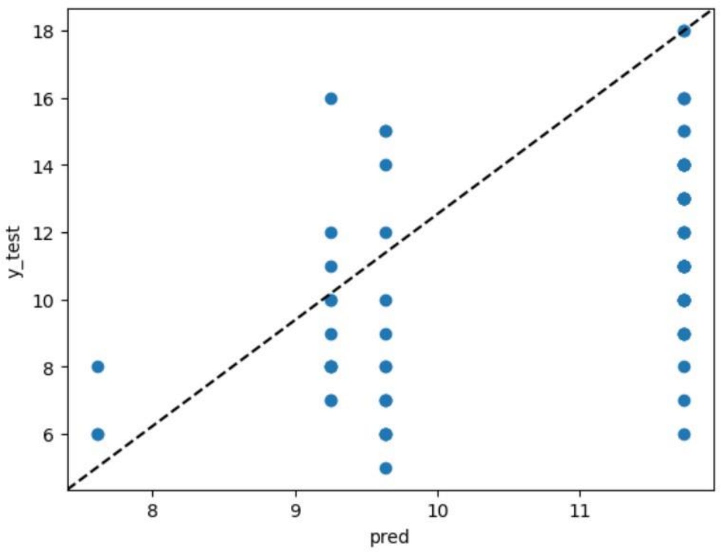


To display the graphical above, the export_graphviz() function is use to export the tree structure to a temporary .dot file, and the graphviz.Source() function to display the image. From the Decision Tree Regressor, everyting that places in the left will the true statement, as above, 252 students is failed the G1 due to school support and 64 students is failed G1 is due to not absence. If looking at the **true** statement, the grade is failures = 0 and the school support is 0 which it will give the squared error of 5.116 with samples of 28 out of 252 failures students with value of 9.25. But if, the school support = 1, the square errors is 10.526 with the sample of 224 students and value of 11.732. If looking at the **false** statement, the grade is failures = 1 and the absences is less than 1 which it will give the squared error of 8.33 with samples of 41 out of 64 absences students with value of 9.634. In order to predict **G1, the variables involve are failure and school support (schoosupp)**. Overall, 28 students out of 316 that fulfilled the failures =0 and school support =0 will not pass in G1.

Below graph, is to make the prediction value against the test data for the MSE. The graph show the MSE is 7.45946 with the straight line of points. This indicate the less points a lies on the line will not confirmed that the model is better. The fute study need to add more observations to get more better model.

```
mean_squared_error(y_test, pred)
```

```
7.4594640215069905
```



2. Transform the variable G2 into new variable, G2T and build a classification tree to classify the G2T variable.

For G2 variables, the tree will be build by using classification tree. From the question, it been asked to transform the variable G2 into new variable, G2T with five categories such as 0 to 4 into E, 5 to 8 into D, 9 to 12 into C, 13 to 16 into B and 17 to 20 into A. Step 1, the new variable G2T is create by defining the function to map the G2 values to categories based on the specified ranges and labels. The new column is create uunder G2T by mapping the G2 value by using function.

```
def map_G2_to_categories(value):

# value: The value of G2 to be mapped.
    if value >= 0 and value <= 4:
        return 'E'
    elif value >= 5 and value <= 8:
        return 'D'
    elif value >= 9 and value <= 12:
        return 'C'
    elif value >= 13 and value <= 16:
        return 'B'
    elif value >= 17 and value <= 20:
        return 'A'
    else:
        return None  # Return None for values outside the specified ranges

# Create a new column 'G2T' by mapping 'G2' values using the function
g2_studentmath_new['G2T'] = g2_studentmath_new['G2'].map(map_G2_to_categories)

g2_studentmath_new.head()  #need to be in dummies
```

Next step 2, which to define the dependent variable (y) and step 3 to define the independent variable (x) which the drop method that removes specified columns from the DataFrame which is ['G1', 'G2', 'G3', 'G2T'] with using of axis=1 represents columns. X now holds the modified DataFrame without the specified columns which useful to separate the features independent variables (x) from the target variable dependent variable (y). In step 4 is to split data set into training and testing sets where the train size is 80:20 in ratio with random state equal to 1 as per below.

```
y = g2_studentmath_new['G2T'] #step2
X = g2_studentmath_new.drop(['G1', 'G2', 'G3','G2T'], axis=1) #step3
X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, train_size = 0.8, random_state = 1)#step4
```

For step 5, is where to get the best maximum depth and best mean square error that use the loop function that iterate from 1 till n-1 which set by n=10, create the Decision Tree Classifier to train model that used to make predictions on the testing set (X_test2), evaluate the model on the testing set and check the current maximum depth gives a better perfomance. From the function, the **Best Maximum Depth is 1** and **best Accuracy Score is 0.56962** for the test data and predicted value**.** In classification, accuracy score is a commonly used metric to evaluate the performance of the model. Since classification involves predicted labels rather than continuous values, using MSE as an evaluation metric may not be appropriate.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

n= 10
best_max_depth = []
best_accuracy = 0

# Iterate over maximum depth values
# the loop iterates from 1 to n-1, which in this case is 1 to 9
for max_depth in range (1,n):
    # Create the Decision Tree Classifier
    classifier = DecisionTreeClassifier(max_depth=max_depth, random_state=1)

    # Train the model
    classifier.fit(X_train2, y_train2)

     # Make predictions on the testing set
    y_pred2= classifier.predict(X_test2)

    # Calculate the accuracy score - use for classification
    accuracy = metrics.accuracy_score(y_test2, y_pred2)

     # Check if current maximum depth gives better performance
    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_max_depth = max_depth


print("Best Maximum Depth:", best_max_depth)
print("Best Accuracy:", best_accuracy)
```

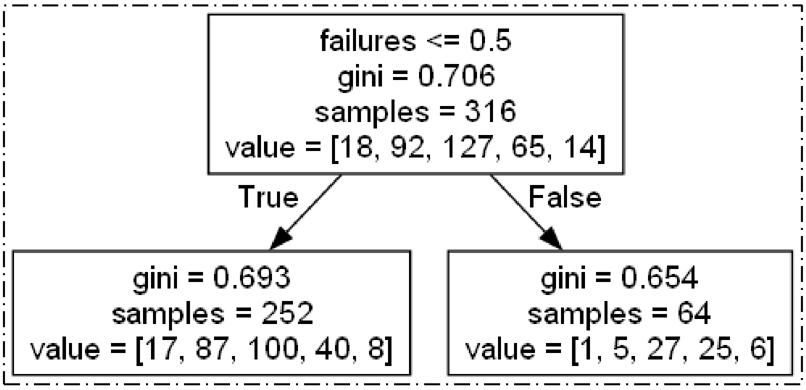
```
Best Maximum Depth: 1
Best Accuracy: 0.569620253164557
```

For the training data, it can be explained that the training accuracy is 40.189% with the maximum depth of 1.

```
classification_tree_g2 = DecisionTreeClassifier(max_depth = 1, random_state= 1)
classification_tree_g2.fit(X_train2, y_train2)
classification_tree_g2.score(X_train2, y_train2)
```

```
0.40189873417721517
```

From the Decision Tree Classifier, everyting that places in the left will the true statement, as below, since the maximum depth is only 1, the variables is only failure either True or False. Out of 316, 252 student is failed and 64 is not failed. So, if looking at the **true** statement, the grade is failures = 0, the mesure of impurity (gini) is 0.693 with the sample of 252 and value of sample 17, 87, 100, 40 and 8. If looking at the **false** statement, the grade is failures = 1, the mesure of impurity (gini) is 0.654 with the sample of 64 and value of sample 1, 5, 27, 25 and 6. In order to predict **G2T, the variables involve is only failure.**



Finally, the final step is to evaluate the tree's performance on the test data. The predict() function can be used for this purpose. This can then build a confusion matrix with the function Transpose to get the same row and column, which shows that we are making correct predictions for around 56.962% of the test data set based on the matrix calulation below.

```
pred = classification_tree_g2.predict(X_test2) #used to make predictions on the X_test2 dataset.
cm = confusion_matrix(y_test2, pred) #predicted labels are stored in the 'pred' variable.

cm_df = pd.DataFrame(cm.T, index=['A', 'B', 'C', 'D'],
                     columns=['A', 'B', 'C', 'D'])
print(cm_df)

   A   B   C   D
A  0   0   0   0
B  0   0   0   0
C  2  15  45  17
D  0   0   0   0

45/(2+15+17+45)  #to calculate the test data on confusion matrix

0.569620253164557
```