Machine Learning Engineer Nanodegree

CAPSTONE PROJECT REPORT

predict the student performance of there secondary education.

Akhila Chitluri

June 27Th 2018

I.Definition

Project Overview :

I today's world everyone are curious to know about future. Especially about our career and performance in further studies. For that purpose we need to evaluate the our past performance for that supervised learning is very useful for prediction by regression and classification. This techniques are used in many streams that will to predict our future Eg : whether prediction, wholesale customer or retail customer, medical fields.

In our project we need to find the student performance based on the primary education and then we will predict their performance in their secondary education.

Dataset Link : https://archive.ics.uci.edu/ml/datasets/student+performance

Problem Statement:

Our Aim is to classify the student's performance for secondary education based on the primary education. Our data contains the details of the students of primary education and some data related to their performance. It is multi class classification.

Features and Description :

Given the details of 649 students related to their primary education and some features related to their family and there economical status based on that now we need to predict the how a student secondary education would be.

These are the features related to dataset:

→ School
→ Sex
→ Age
→ Family size
→ Father's Education

→ Mother’s Education
→ Mother’s job
→ Father’s job
→ Internet
→ travel time to school to home
→ Family educational support
→ wants to go to higher educational
→ Quality of family relationships
→failures
→ Absences
→ 1st time grade
→ 2nd term grade
By considering the above factors we will classify the data and then we will predict their future performance in their secondary education.

Performance Metrics:

In this we will use the Accuracy, F beta score and others like recall, precision .

We define them as :
accuracy =(TP+TN)/(TP+TN+FP+FN)
precision = TP/(TP+FP)
recall = TP(TP+FN)
f beta score = (1+beta^2)*(precision*recall)/(beta^2*precision)+recall
beta =0.5

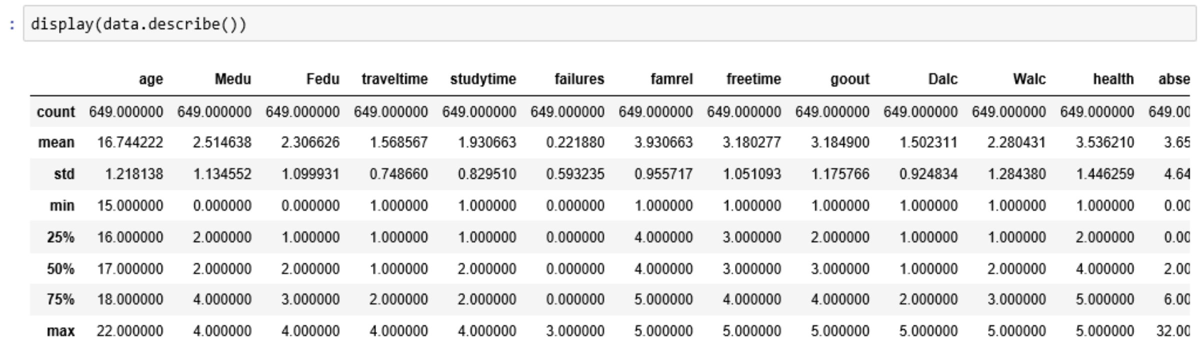
II. Analysis

Data Exploration:

In this Data Exploration, I explored who may Attributes and how may instances the data set contains . In this face I found that There are missing values in the data. A also explored that the data consists of mixed variants like categorical, numerical and some binary data .(values Yes/No) . I also found some attributes which are not necessary for our classification like address,romatic,guardian.

Data Visulaization :

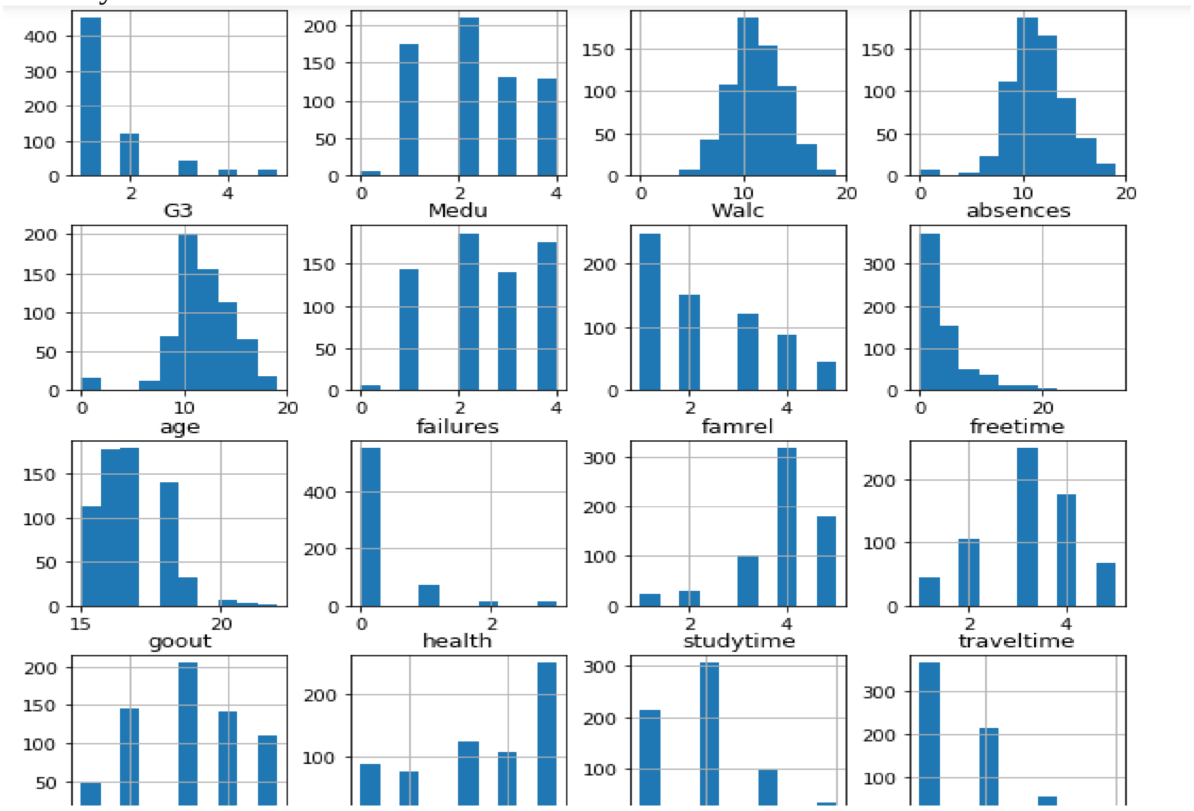
→ In this part, I found some data related to the data like its mean, standard deviation,min,max values.



```
: display(data.describe())
```

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc	health	abse
count	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.00
mean	16.744222	2.514638	2.306626	1.568567	1.930663	0.221880	3.930663	3.180277	3.184900	1.502311	2.280431	3.536210	3.65
std	1.218138	1.134552	1.099931	0.748660	0.829510	0.593235	0.955717	1.051093	1.175766	0.924834	1.284380	1.446259	4.64
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.00
25%	16.000000	2.000000	1.000000	1.000000	1.000000	0.000000	4.000000	3.000000	2.000000	1.000000	1.000000	2.000000	0.00
50%	17.000000	2.000000	2.000000	1.000000	2.000000	0.000000	4.000000	3.000000	3.000000	1.000000	2.000000	4.000000	2.00
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000000	4.000000	4.000000	2.000000	3.000000	5.000000	6.00
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	32.00

- ➔ I also visualized the attributes the by using histograms whether it is skewed or normally distributed.



By this I noticed that some of the data is skewed but does not vary to large values .so I concluded that the data can be normalized without using log transform.

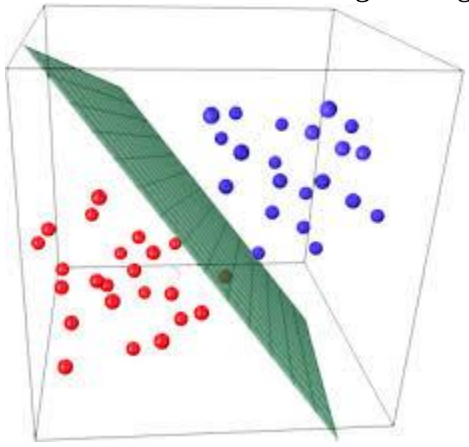
Algorithms and Techniques:

Algorithms:

Our Aim is to classify the data as we have the data which is labeled we use supervised classification and I used following Techniques and Algorithms.

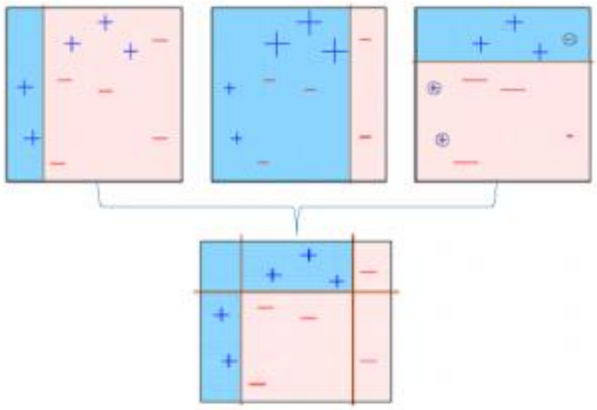
1. *Logistic Regression :*

I used Logistic Regression because it is simple to fast .But there are some disadvantages they are it may sometimes leads to over fitting .I the case of more features this Logistic Regression may not work well in all cases. Logistic regression classifies in this way.



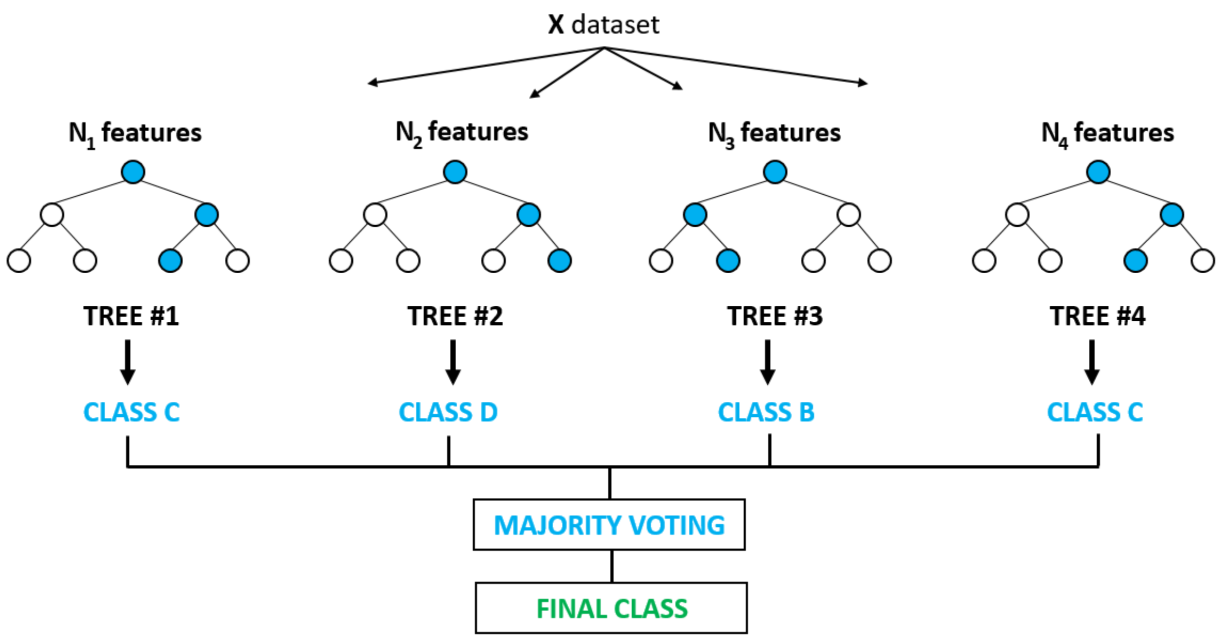
2. *Adaboost Classifier:*

This is one type of Boosting classifiers in the ensemble methods. In Adaboost classifier the data will be divided into subsets and then it will be evaluated according .This shows that it is not biased and the data may not over fit.
Adaboost classifies the data in this way.



3. *Random Forest Classifier:*

This Random forest Classifier uses decision tress by selecting some features and classifies the data this process will be repeated no of times and the best one among them will be selected. This shows that the data is will not be biased and does not over fits.



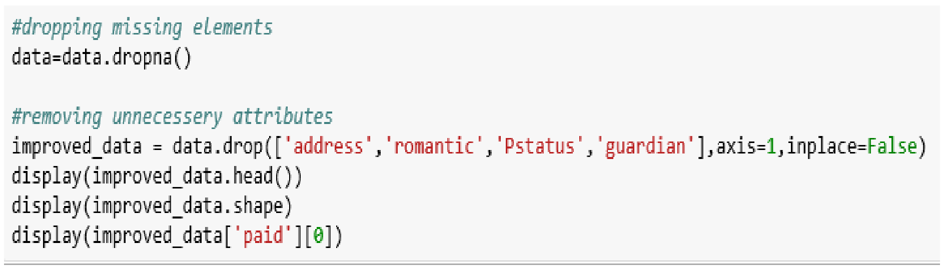
We use some Techniques to optimize the model and to get best results and in this problem. I use Grid search cv which helps to get the better results based on our parameters tuning .By tuning parameters the data may give more accurate results and its performance increases.

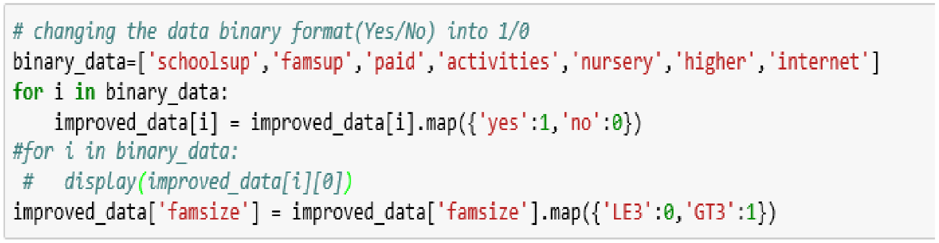
Benchmark Model:
I used Logistic regression as the benchmark model as it is fast to implement and simple classifer.In this I calculated the accuracy and fbeta score as performance matrices to improve the results the next ensemble methods like ada boost classifier and the random forest claasifers will be used and the best f them i.e: that will give the best fbeta and accuracy results will be selected.

III Methodology:

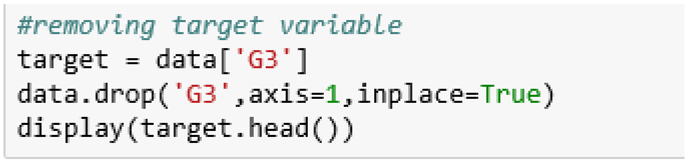
Data Preprocessing:

i. In this stage I removed all the missing values and converted all the binary values yes/no to 1/0 respectively. And also I changed one attribute which has its LE3(less than 3) and GT3 (greater than 3) .so I changed this data into 0/1 respectively as it is also a binary data.

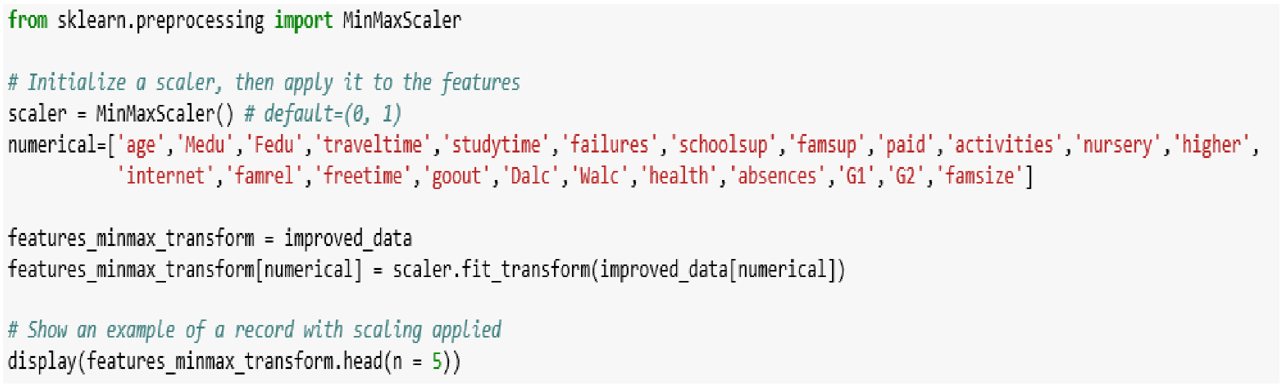

```
#dropping missing elements
data=data.dropna()

#removing unnecessery attributes
improved_data = data.drop(['address','romantic','Pstatus','guardian'],axis=1,inplace=False)
display(improved_data.head())
display(improved_data.shape)
display(improved_data['paid'][0])
```

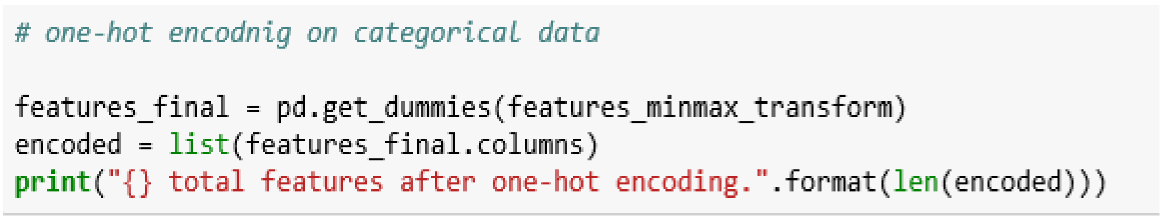

```
# changing the data binary format(Yes/No) into 1/0
binary_data=['schoolsup','famsup','paid','activities','nursery','higher','internet']
for i in binary_data:
    improved_data[i] = improved_data[i].map({'yes':1,'no':0})
#for i in binary_data:
 #   display(improved_data[i][0])
improved_data['famsize'] = improved_data['famsize'].map({'LE3':0,'GT3':1})
```

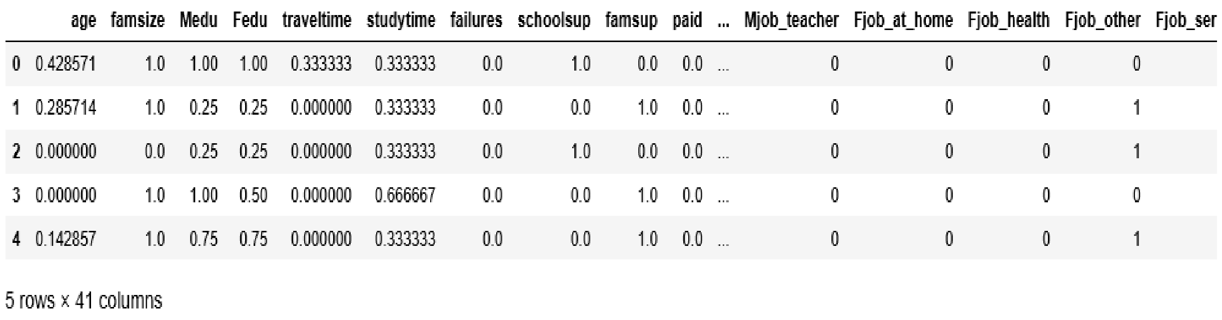
ii. Data split: in this stage the data is divided into target values and inputs differently.


```
#removing target variable
target = data['G3']
data.drop('G3',axis=1,inplace=True)
display(target.head())
```

iii. Data transformation: In this the will be transformed by using min max scalar and then I used one hot encoding to convert the categorical data into numerical


```
from sklearn.preprocessing import MinMaxScaler

# Initialize a scaler, then apply it to the features
scaler = MinMaxScaler() # default=(0, 1)
numerical=['age','Medu','Fedu','traveltime','studytime','failures','schoolsup','famsup','paid','activities','nursery','higher',
          'internet','famrel','freetime','goout','Dalc','Walc','health','absences','G1','G2','famsize']

features_minmax_transform = improved_data
features_minmax_transform[numerical] = scaler.fit_transform(improved_data[numerical])

# Show an example of a record with scaling applied
display(features_minmax_transform.head(n = 5))
```

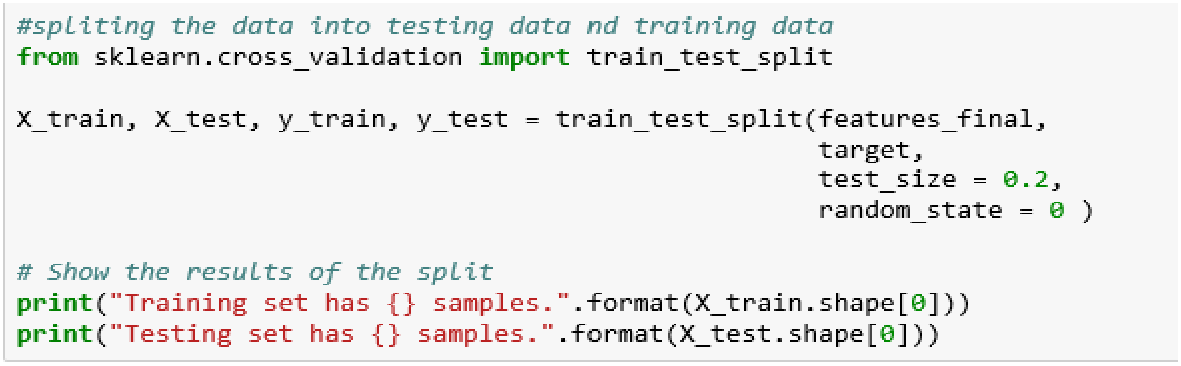

```
# one-hot encodnig on categorical data

features_final = pd.get_dummies(features_minmax_transform)
encoded = list(features_final.columns)
print("{} total features after one-hot encoding.".format(len(encoded)))
```



	age	famsize	Medu	Fedu	traveltime	studytime	failures	schoolsup	famsup	paid	...	Mjob_teacher	Fjob_at_home	Fjob_health	Fjob_other	Fjob_ser
0	0.428571	1.0	1.00	1.00	0.333333	0.333333	0.0	1.0	0.0	0.0	...	0	0	0	0	
1	0.285714	1.0	0.25	0.25	0.000000	0.333333	0.0	0.0	1.0	0.0	...	0	0	0	1	
2	0.000000	0.0	0.25	0.25	0.000000	0.333333	0.0	1.0	0.0	0.0	...	0	0	0	1	
3	0.000000	1.0	1.00	0.50	0.000000	0.666667	0.0	0.0	1.0	0.0	...	0	0	0	0	
4	0.142857	1.0	0.75	0.75	0.000000	0.333333	0.0	0.0	1.0	0.0	...	0	0	0	1	

5 rows × 41 columns

iv. Data splitting : we will split the data into training data and testing data in which we train the model using training data and test using testing data I kept 20% of data for testing



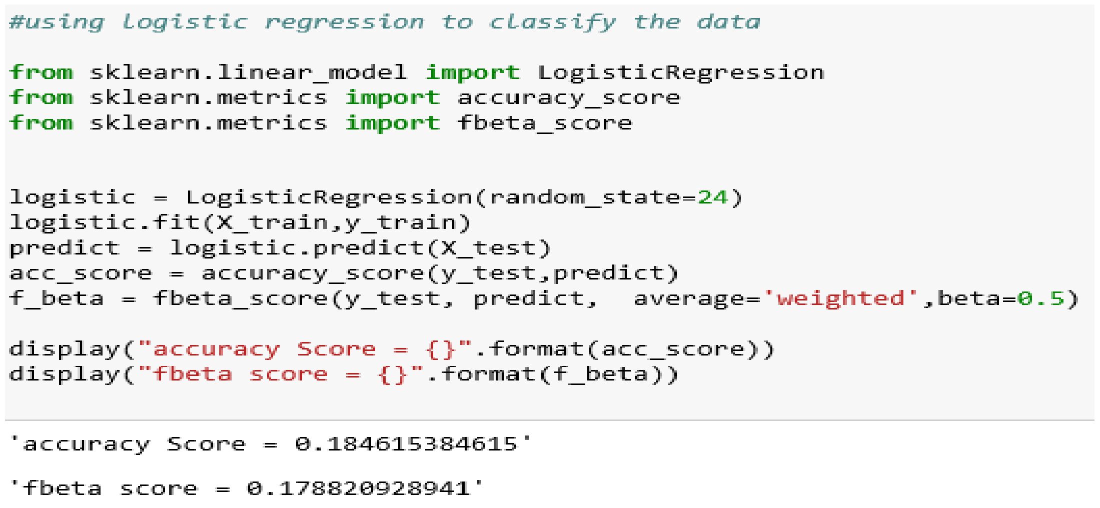
```python
#spliting the data into testing data nd training data
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(features_final,
                                                    target,
                                                    test_size = 0.2,
                                                    random_state = 0 )

# Show the results of the split
print("Training set has {} samples.".format(X_train.shape[0]))
print("Testing set has {} samples.".format(X_test.shape[0]))
```

v. Data classification and fitting and Predicting with different models.

a) For every model we need to import the classifier for the respective packages in the sklearn .

b) We will fit the data to the model with training data and then we will predict with the by using testing data after that we will find the accuracy and fbeta score by using the true values and predicted values.

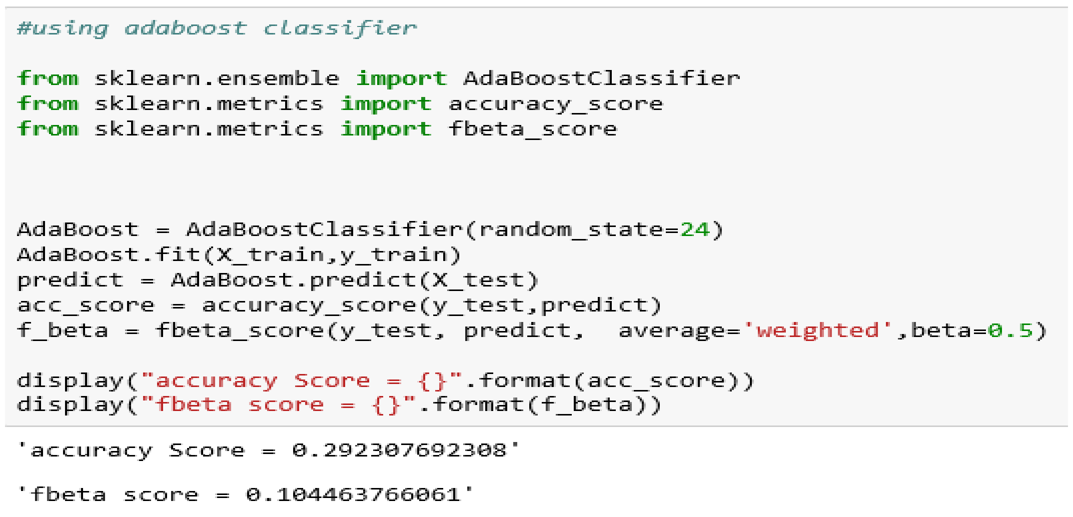
Logistic regression:



```python
#using logistic regression to classify the data

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import fbeta_score


logistic = LogisticRegression(random_state=24)
logistic.fit(X_train,y_train)
predict = logistic.predict(X_test)
acc_score = accuracy_score(y_test,predict)
f_beta = fbeta_score(y_test, predict,  average='weighted',beta=0.5)

display("accuracy Score = {}".format(acc_score))
display("fbeta score = {}".format(f_beta))
```

```
'accuracy Score = 0.184615384615'

'fbeta score = 0.178820928941'
```

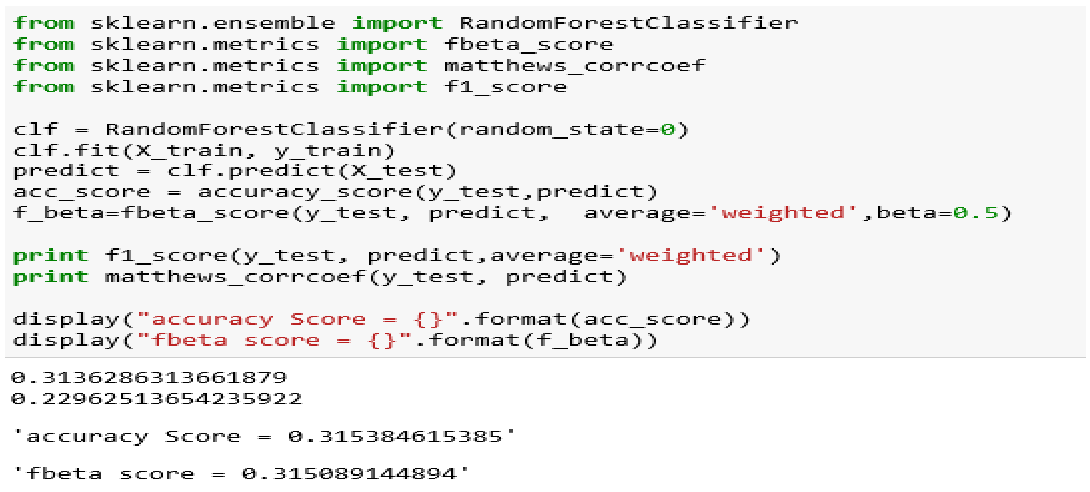
Adaboost classification:



```
#using adaboost classifier

from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import fbeta_score



AdaBoost = AdaBoostClassifier(random_state=24)
AdaBoost.fit(X_train,y_train)
predict = AdaBoost.predict(X_test)
acc_score = accuracy_score(y_test,predict)
f_beta = fbeta_score(y_test, predict,  average='weighted',beta=0.5)

display("accuracy Score = {}".format(acc_score))
display("fbeta score = {}".format(f_beta))
```

```
'accuracy Score = 0.292307692308'

'fbeta score = 0.104463766061'
```

Random Forest:



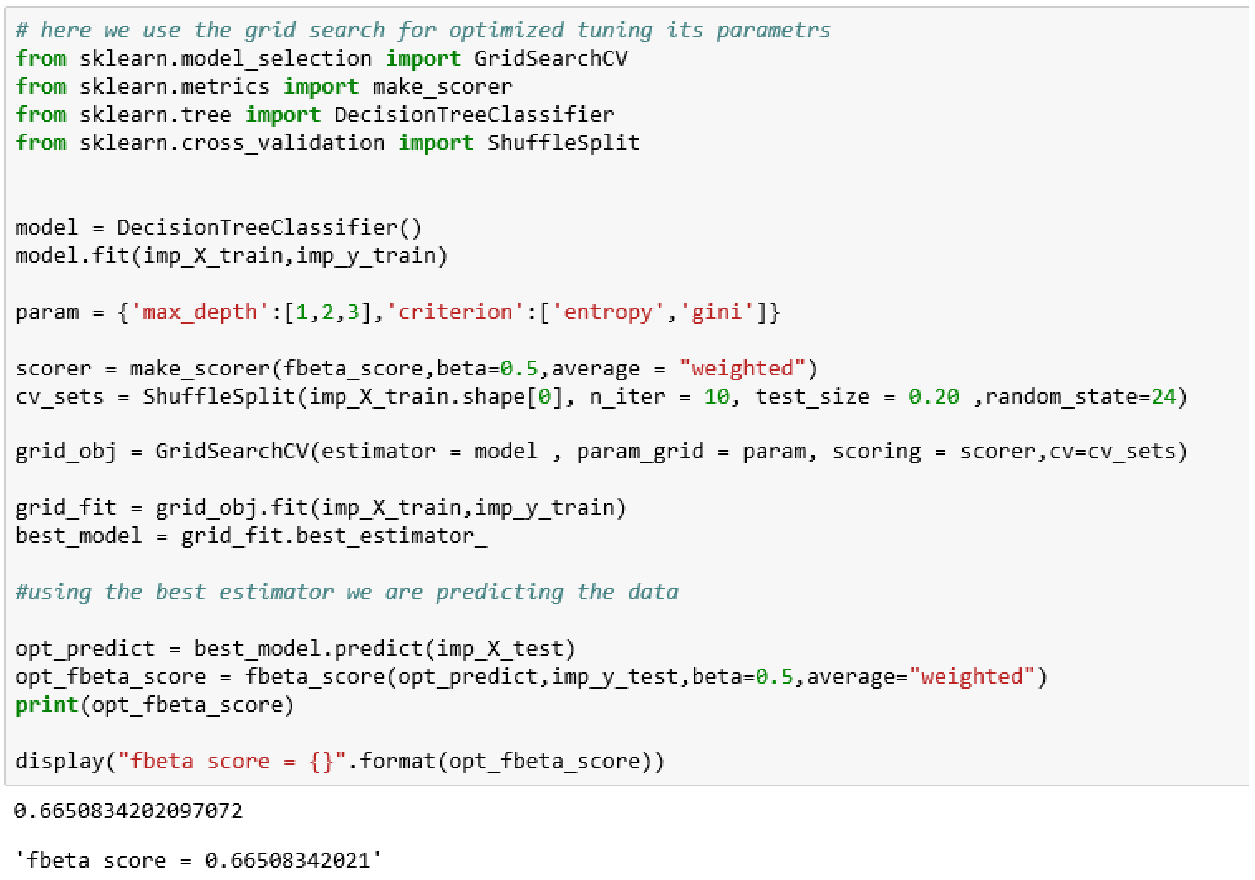
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import fbeta_score
from sklearn.metrics import matthews_corrcoef
from sklearn.metrics import f1_score

clf = RandomForestClassifier(random_state=0)
clf.fit(X_train, y_train)
predict = clf.predict(X_test)
acc_score = accuracy_score(y_test,predict)
f_beta=fbeta_score(y_test, predict,  average='weighted',beta=0.5)

print f1_score(y_test, predict,average='weighted')
print matthews_corrcoef(y_test, predict)

display("accuracy Score = {}".format(acc_score))
display("fbeta score = {}".format(f_beta))
```

```
0.3136286313661879
0.22962513654235922

'accuracy Score = 0.315384615385'

'fbeta score = 0.315089144894'
```

Among all above I used random forest classifier as it gave me the best when compared to the other two algorithms.

Reinforcement :

I used Grid serach cv for improving its performance and tuned some of the parameters in the random forest like max_depth , criterion and improved the result from 31 % to 66 %



```python
# here we use the grid search for optimized tuning its parametrs
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer
from sklearn.tree import DecisionTreeClassifier
from sklearn.cross_validation import ShuffleSplit


model = DecisionTreeClassifier()
model.fit(imp_X_train,imp_y_train)

param = {'max_depth':[1,2,3],'criterion':['entropy','gini']}

scorer = make_scorer(fbeta_score,beta=0.5,average = "weighted")
cv_sets = ShuffleSplit(imp_X_train.shape[0], n_iter = 10, test_size = 0.20 ,random_state=24)

grid_obj = GridSearchCV(estimator = model , param_grid = param, scoring = scorer,cv=cv_sets)

grid_fit = grid_obj.fit(imp_X_train,imp_y_train)
best_model = grid_fit.best_estimator_

#using the best estimator we are predicting the data

opt_predict = best_model.predict(imp_X_test)
opt_fbeta_score = fbeta_score(opt_predict,imp_y_test,beta=0.5,average="weighted")
print(opt_fbeta_score)

display("fbeta score = {}".format(opt_fbeta_score))
```

```
0.6650834202097072

'fbeta score = 0.66508342021'
```

IV Results

Model Evaluation And Validation:

Finally my model selection is randomforest classifier .At first it gave the f_beta score result as 0.31 and accuracy score 0.31 and finally by using grid search cv and tuning parameters our fbeta score improved to 0.66 .

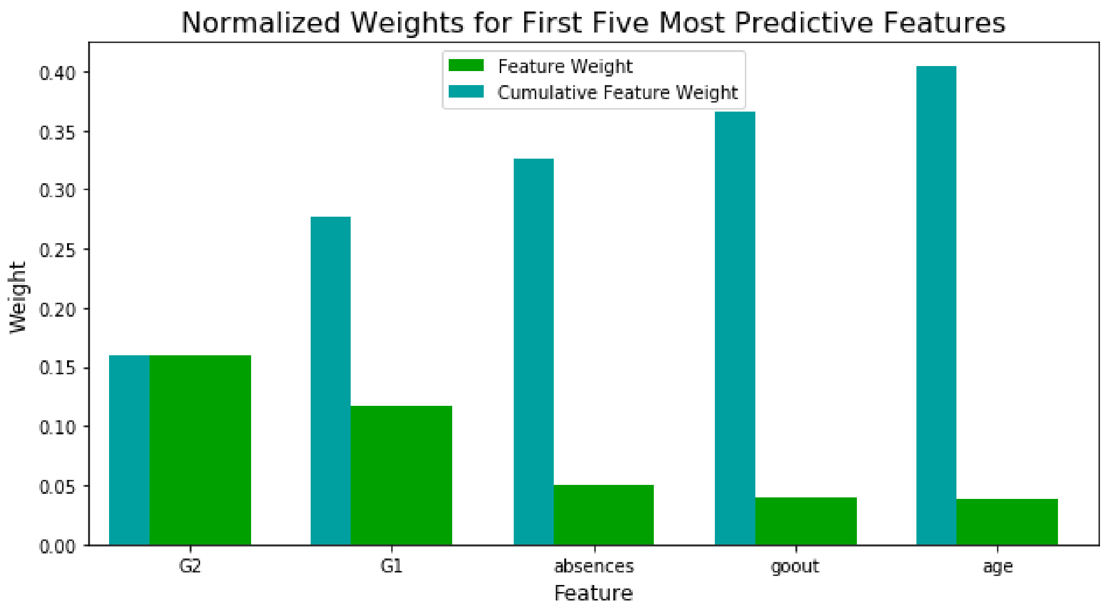
By this a can say that our model is more accurate and good when compared to other models I used before.

Justification :

Our benchmark model is logistic regression and final model is random forest classifier . In the case of logistic regression the fbeta score is very low its value is 0.17 and in our final model at initial stage our fbeta score is 0.31 which is far better than logistics’ score .By this analysis I selected the final model is random forest and after using our optimization techniques and tuning the parameters we achieved our fbeta score to 0.66 % .Hence I can justify that our model gives best solution.

V. conclusion:

Our final conclusion is the data is multi class classified and all the features does not define the output .out of 33 attributes present in my data set after removing unnecessary attributes we get 29 and after using feature importance that there are five features which mostly contribute to the output .They are



By this I can conclude that when we use many features it will be better to ensemble methods to classify and we can easily classify the data .In this project after finding the different models the best results that was provided by random forest classifier and I was satisfied as my fbeta score improved from initial to final from 0.31 to 0.66.

Reflection :

1. I collected my data from the UCI machine learning and my data set name is student performance.
2. After collecting the dataset I observed the data and its attributes which contains 649 instances and 33 attributes.
3. The data is inconsistent as my data is contains some of the missing values and my is multi variant data which contains the numerical, categorical and binary values also.
4. In the pre processing of the data I removed all the missing values and then separated target variables and input variables and plotted curves to visualize the data .
5. The I choice one bench mark model logistic regression and classify the data with that classification and then I also tried different types of the classifications and predicted the

values and fbeta score of all the classifications and picked the which has highest fbeta score.

6. Among all I used the best one is Random forest classifier and then I check for important features .by using that features I tried to fit the data for the same classifier and then checked the fbeta score which gave me approximately the same ans .
7. So by using those features of data I used a technique called grid search cv I optimized the data by tuning the parameters and I finally successes in getting the fbeta score as 66%

Improvements:

I think by using another ensemble methods like Xboost classifier we may get the better results .

We can also using boosting and bagging algorithms that may also gives good results.

I think my choice of algorithm is good and performed well .

References:

References:

→I used visuals.py from the previous project for feature selection.

Ref: https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/
Ref: https://www.analyticsvidhya.com/blog/2014/06/introduction-random-forest-simplified/
Techinques:
https://archive.ics.uci.edu/ml/datasets/student+performance