Student Performance Predictor

Group Members:

Madhusudhan Tungamitta, Balaji Arunachalam, Shreyas Ghuge, Naman Mittal, Imran Shahid

Changes to the project statement: ⚠

After trying out multiple things on the previous problem statement of H&M Recommendation System, our team decided to work on a relatively easier topic so as to explore different ML algorithms in more depth. The reason for the change in topic was basically due to a very low accuracy score. This was communicated to the prof and we have submitted the new project proposal. For our current problem statement, we have taken into account different ML approaches and further used the inbuilt ML libraries.

Preprocessing:

Generally, ML models are trained and tested on a humongous dataset. However, we have a minimalistic dataset from Kaggle, that is in a csv format. The dataset has columns such as school, sex, age, address, famsize, … , health, absences, passed. Having multiple attributes with a variety of records we had to scale down and standardize the values in them. For achieving this, we used the MinMaxScaler that Scikit learns provides. Furthermore, to sanitize the dataset having textual data, we converted it to numerical values by using the Factorize method from pandas.

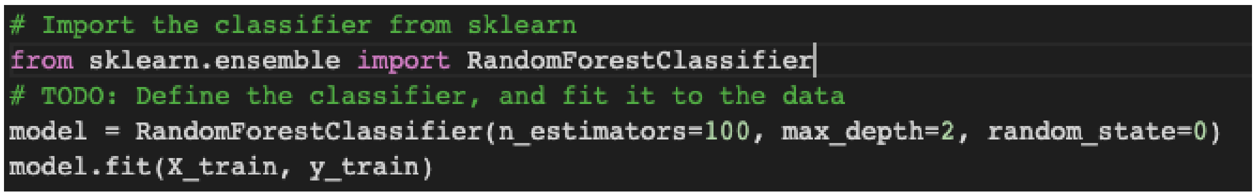
MinMax Scaling - "MinMax scaling transforms attributes by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one" [1]

Factorize - encode the object as an enumerated type or categorical variable.
This method is useful for obtaining a numeric representation of an array when all that matters is identifying distinct values.

Basic data cleaning - removing na and blank values

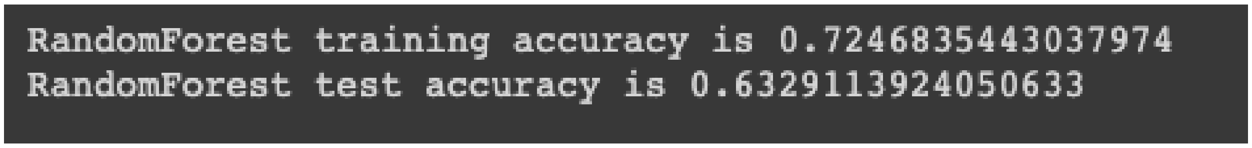
Algorithm 1: Random Forest (Naman Mittal, Shreyas Ghuge):

A random forest classifier consists of a large number of individual decision trees that operate as a unit. Each individual tree gives a prediction and the class with the highest votes is our model's prediction.



```
# Import the classifier from sklearn
from sklearn.ensemble import RandomForestClassifier
# TODO: Define the classifier, and fit it to the data
model = RandomForestClassifier(n_estimators=100, max_depth=2, random_state=0)
model.fit(X_train, y_train)
```

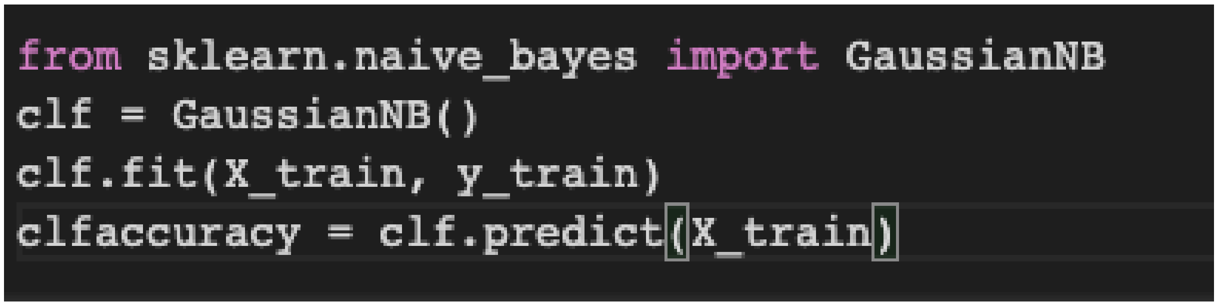
Why Random Forest For our Data set : The reason for choosing random forest is because this model works very well since a large number of correlated trees working as a group will perform much better than any of the individual models. We use the Random Forest Classifier from sklearn. ensemble, defining the parameters n_estimators, max_depth, and random_state to get the model.We have also got the accuracy of the training and testing data



```
RandomForest training accuracy is 0.7246835443037974
RandomForest test accuracy is 0.6329113924050633
```

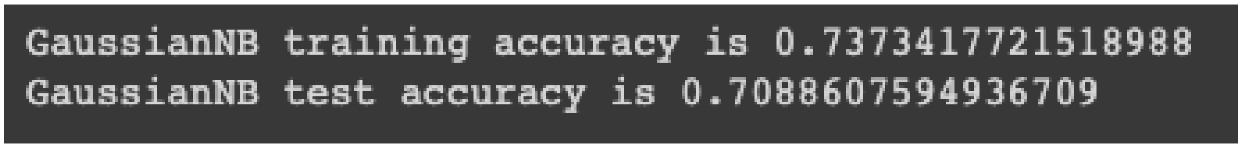
Algorithm 2: Gaussian NB(Madhusudhan Tungamitta, Balaji Arunachalam):

By using the libraries we implemented a working model that is able to get ().
Navies Bayes by default makes an assumption that features are independent. We are passing the x train and y train in the fit function to fit the model.



```
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
clf.fit(X_train, y_train)
clfaccuracy = clf.predict(X_train)
```

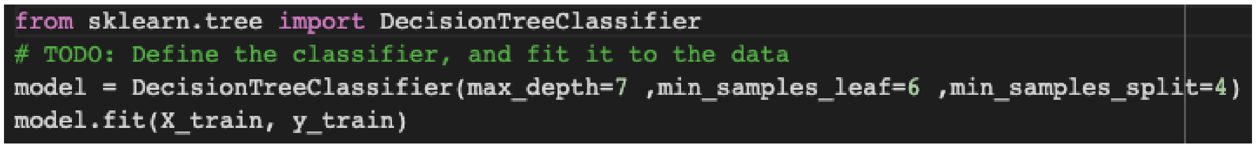
Why Gaussian NB For our Data set: Further, we will set both the probabilities as 0.5 and by using a beta distribution to normalize the dataset. We will compute the probabilities and update the prior values with each step. Since a Gaussian Naive Bayes uses given p(a/b) to identify the classes enough features that allow the entropy of the choice to be reduced greatly thus allowing us to make highly precise and accurate predictions.



```
GaussianNB training accuracy is 0.7373417721518988
GaussianNB test accuracy is 0.7088607594936709
```

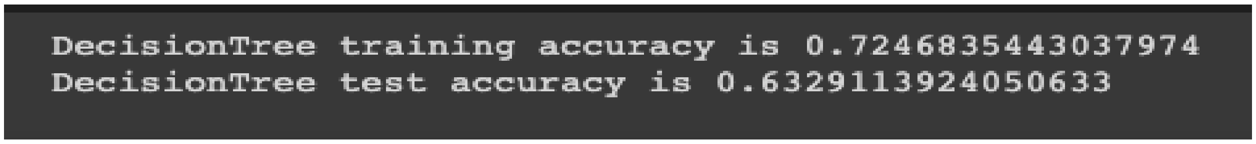
Algorithm 3: Decision Trees(Imran Shahid, Madhusudhan Tungamitta):

We are using a decision tree classifier to predict the student's pass and fail based on the many features listed. In a decision tree, an internal node represents an attribute, a branch has the decision rule and the leaf would be an outcome.



```
from sklearn.tree import DecisionTreeClassifier
# TODO: Define the classifier, and fit it to the data
model = DecisionTreeClassifier(max_depth=7 ,min_samples_leaf=6 ,min_samples_split=4)
model.fit(X_train, y_train)
```

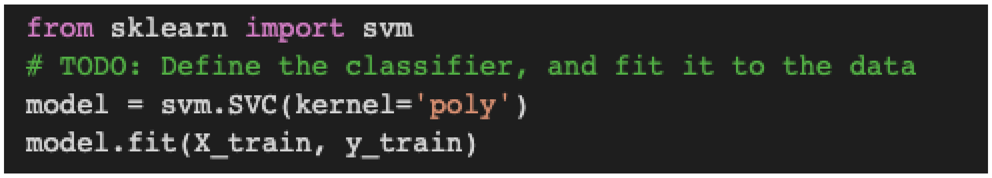
Why Decision Tree for this Data set: We use this classifier in our project since ours is a binary classification and we have a 50%-50% chance of students either passing or failing. We use the DecisionTreeClassifier from sklearn. tree, defining the max depth, min_samples_leaf, and min_samples_split parameters and using the model for prediction. The reason that the decision tree makes sense is that the default rate of classification in this problem is 50 percent and the decision tree will by its nature give us a greater result



```
DecisionTree training accuracy is 0.7246835443037974
DecisionTree test accuracy is 0.6329113924050633
```

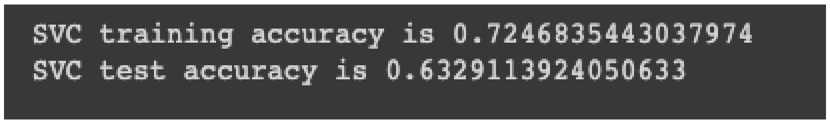
Algorithm 4: Support Vector Form(Shreyas Ghuge, Balaji Arunachalam):

Using SVM we are fitting the data and returning the best fit line that categorizes the data, Here SVM separates the classes in a good way by generating the hyperplanes and the class will choose the hyperplane that helps in the segregation of the classes separately.



```
from sklearn import svm
# TODO: Define the classifier, and fit it to the data
model = svm.SVC(kernel='poly')
model.fit(X_train, y_train)
```

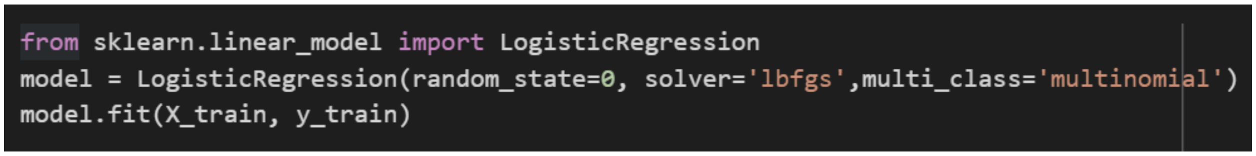
Why SVM: Since features may share semantic similarities and therefore be clustered according to perhaps only a few features out of the many and ultimately into two superclusters the classification labels pass or Fail. we may show that the data is separable using the hyperplane. This makes an SVM which is by itself an excellent tool for binary classification of clusters highly suited to this task.



```
SVC training accuracy is 0.7246835443037974
SVC test accuracy is 0.6329113924050633
```

<u>Algorithm 5:</u> Logistic Regression(Imran Shahid, Naman Mittal):

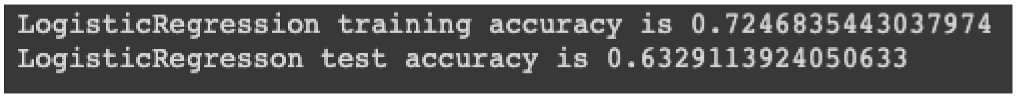
At the initial stage, we used Scikit Learn, an open source library in python which is used to predict data analytics in the most efficient and simple way.



```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(random_state=0, solver='lbfgs',multi_class='multinomial')
model.fit(X_train, y_train)
```

Why Logistic Regression for our dataset: When the dependent variable means the target is categorized then the Logistic Regression is one of the effective machine learning algorithms to predict the probability of results, in our project it means whether the student will pass or fail depends on the pre-processed data.

Among the various types of Logistic Regression, we are using Multinomial Logistic Regression which allows two or more categories of dependent or outcome variables without ordering. It does not assume linearity, normality, or homoscedasticity so it is one of the most attractive algorithms. The random state will ensure that the splits are reproducible to the data which we are generated. Scikit-Learn uses random permutation if the user doesn't define it here it will use this as a seed to a random number generator. The solver "lbfgs" (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) is used to approximate the updates in the second derivative matrix with the gradient evaluations. It is used to save memory by saving only the latest updates. In Scikit-Learn this will be the default solver for Multinomial Logistic Regression. The fit() uses the ordinary least squares method to describe the relationship between the set of predictors and continuous response.



```
LogisticRegression training accuracy is 0.7246835443037974
LogisticRegresson test accuracy is 0.6329113924050633
```

Summary:

//To do: add a histogram here comparing accuracies of all the above algorithms.

References:

1. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
2.