

# “Optimizing Student Performance Prediction with Supervised Learning”

Manjula Kalmath<sup>Asst.Prof</sup>

JSS Shri Manjunatheshwara MCA Institute, Vidyagiri, Dharwad 04  
Affiliated with Karnataka University Dharwad  
City: Dharwad, State: Karnataka, Country: India

\*\*\*

**Abstract** - Accurately predicting student performance is crucial for effective educational interventions and resource allocation. This study compares prominent supervised machine learning algorithms for forecasting student achievement, addressing the lack of comprehensive evaluations in existing research. Analyzing a dataset of 60 students from JSS SMI UG and PG Studies, Dharwad, India, the research incorporates student demographic information, academic performance indicators, engagement metrics, and final performance results to identify factors influencing academic success. The study evaluates several models, including logistic regression, decision trees, random forest, support vector machine (SVM), k-nearest neighbors (KNN), and Naive Bayes algorithms. A 5-fold cross-validation technique assesses model performance more reliably than a simple train\_test\_split, enhancing the evaluation of the model's generalizability and mitigating overfitting risks. Results indicate that the decision tree classifier achieves a mean accuracy of approximately 0.967, with most folds demonstrating perfect accuracy, although it may be susceptible to overfitting. The random forest classifier exhibits a mean accuracy of around 0.883, showing slightly more variability but greater robustness. KNN attains a mean accuracy of approximately 0.7667, while Gaussian Naive Bayes (NB) demonstrates consistent accuracy with a mean of around 0.817. Logistic regression displays lower and less consistent accuracy at approximately 0.583. SVC exhibits the lowest accuracy at 0.45, suggesting that it might not be well-suited for the dataset. This study provides valuable insights into selecting and applying appropriate machine learning techniques for predicting student performance, potentially benefiting educators and researchers in the field.

**Key Words:** Supervised machine learning algorithms, Academic performance, 5-Fold cross-validation, Confusion matrix, Accuracy, Prediction, Classification Report.

## 1.INTRODUCTION :

In education, predicting student academic performance is essential for identifying at-risk students and enhancing educational outcomes. This study

investigates the relationship between various factors, such as internal assessments, seminar engagement, assignment completion, attendance, and extracurricular activities, and their influence on students' final academic results. By employing historical data and machine learning techniques, this research aims to develop a predictive model that accurately estimates final student grades based on these internal academic indicators. The ability to precisely forecast student performance enables educators and institutions to implement timely interventions and tailor learning strategies to individual needs.

The primary objective of this research is to assess the efficacy of various supervised learning algorithms in predicting student performance. A real-world dataset will be utilized, incorporating features such as gender, assignment scores, seminar participation, internal marks, attendance, and overall average, with the student's grade as the target variable. While numerous supervised learning methods offer potential solutions, there is a lack of comprehensive understanding regarding the comparative advantages and disadvantages of different approaches in this field. Selecting the most appropriate algorithm for a specific educational context remains challenging. This study addresses this gap by conducting a comprehensive comparative analysis of several algorithms, including logistic regression, decision trees, random forest, SVM, KNN, and Naive Bayes. To evaluate the performance of these algorithms, 5-fold cross-validation will be utilized, ensuring robust assessment and minimizing the risk of overfitting. The algorithms will be compared using performance metrics such as accuracy, precision, recall, and F1 score, which will be calculated using a confusion\_matrix.

Additionally, the study will evaluate the models by predicting grades for new, unseen data, visualizing the predictions with various output plots such as confusion\_matrix and heatmap. These visualizations will aid in interpreting the effectiveness of each algorithm in predicting student performance. By elucidating the strengths and limitations of different machine learning models in educational settings, this research aims to

provide insights into which algorithms are most effective for student performance prediction and the practical implications of their use in real-world scenarios. Furthermore, the study seeks to address the challenges faced by educational institutions in managing diverse and complex data sources, ultimately contributing to more informed decision-making processes.

## 2. LITERATURE SURVEY:

Recent studies have expanded beyond final exams to consider **continuous assessments** and **extracurricular participation** as key factors influencing student outcomes. The success of predictive models heavily depends on the choice of **algorithm** and **dataset**. For graduate-level predictions, factors such as **GRE scores**, **GPA**, **motivation**, and **supervisor performance** are commonly used [1]; [2]; [3]. Incorporating **demographic**, **academic**, and **behavioural factors** enhances prediction accuracy, emphasizing the importance of selecting relevant features and robust algorithms [4]. Predicting postgraduate performance involves considering multiple aspects, such as **past academic records**, **demographics**, and **classroom behaviour** [5]; [6]. **Educational data mining (EDM)** and **machine learning** techniques are widely applied to analyze student data, uncover patterns, and generate insights for decision-making [7]; [8]; [9]. **Supervised machine learning** is particularly effective for **classification** and **regression problems** in predicting academic outcomes [10]. This study evaluated various supervised machine learning algorithms, with **logistic regression** showing the best performance in predicting student success.

Machine learning algorithms have demonstrated strong capabilities in predicting student academic performance based on various factors. random forest is a top performer, with studies reporting accuracies of 78% [11] and 77.3% [12]. logistic regression and ensemble methods like bagging and voting also perform well [13]; [14]; [15]. Key predictive factors include prior academic performance, study duration, gender, and engagement with online learning systems [12]. Other algorithms, such as Naïve Bayes and SMNaive Bayes, have shown significant predictive accuracy [16]. In one study, artificial neural networks (ANN) achieved the highest performance, explaining 89% of GPA variation [17].

Machine learning approaches, particularly supervised learning, have become integral to data analysis, with algorithms learning to categorize examples in classification tasks or predict numeric values in regression tasks [18]. Researchers are increasingly utilizing supervised, unsupervised, and semi-supervised learning techniques to handle the growing data volumes in education and other fields learning [19]; [20]. Supervised learning involves labelled data, focusing on

classification and regression, while unsupervised learning seeks patterns in unlabelled data using methods like clustering and dimensionality reduction [21]. Semi-supervised learning combines both labelled and unlabelled data for enhanced predictions. Popular supervised techniques such as decision tree, Naïve Bayes, logistic regression, support vector machine, and Neural Networks yield accurate results in various fields [22].

## 3. METHODOLOGY:

Figure 1 illustrates the overall working process of this study, evaluating six machine learning models for predicting student performance.

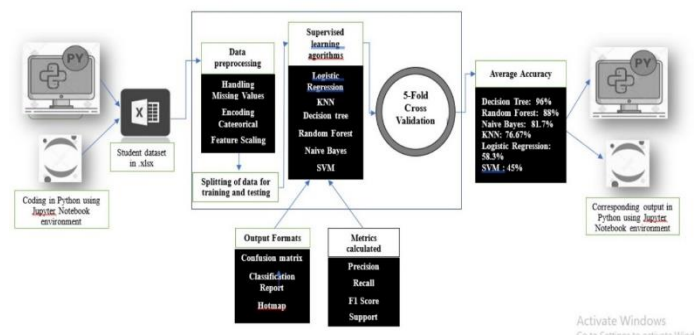


Figure 1 : Machine Learning-Based Prediction of Student Academic Performance.

### 3.1 Dataset Description:

This study employs a comparative analysis of supervised learning algorithms to predict student academic performance. We utilize a dataset maintained in an excel file, comprising records of student performance. The dataset includes the following features:

- Roll\_no: Unique identifier for each student.
- Gender: Student's gender (Male/Female).
- Assignment: Marks obtained in assignments.
- Seminar: Marks obtained in seminar presentations.
- Internal: Marks obtained in internal assessments.
- Attendance: Percentage attendance of the student.
- Average: Final average marks.
- Grade: Final grade obtained by the student (A, B, C, etc.).

The dataset consists of records for 60 students, with the grade received by each student treated as the target variable for the prediction task as shown in Table 2.

### 3.2 Evaluation metrics:

The performance of six supervised machine learning algorithms logistic regression, decision trees, random

forest, SVM, KNN, and Naive Bayes was evaluated using several metrics:

- Accuracy: Percentage of correctly classified instances out of the total instances.
- Precision: Proportion of true positives among all positive predictions.
- Recall: Proportion of true positives among actual positives.
- F1 Score: Harmonic mean of precision and recall.
- Confusion Matrix: Visual representation of the model's performance, displaying true positives, false positives, true negatives, and false negatives.

### 3.3 Software and tools:

The analysis and model implementation were performed using python (version 3.11.5) in jupyter notebook (version 7.2.2) environment. The following libraries were utilized:

- pandas: For data manipulation and pre-processing.
- scikit learn: For implementing machine learning models, cross validation, and evaluation metrics.
- matplotlib and seaborn: For visualizing data and model performance.
- numpy: For efficient numerical computations.
- openpyxl: To read Excel files in python.

### 3.4 Methods:

#### Mathematical background:

To understand the metrics used in the classification report, let's break down the precision, recall, F1 score, and support using mathematical formulas, and explain them in the context of our dataset. These metrics are based on the model's confusion\_matrix for each class.

#### Confusion Matrix for a Single Class:

The **confusion\_matrix** for a binary classifier can be generalized to multiple classes.

**Table 1 : General form of confusion\_matrix**

Actual	Predicted Positive (class 1)	Predicted Negative (not class 1)
Actual Positive (class1)	True Positive (TP)	False Negative (FN)
Actual Negative (not class1)	False Positive (FP)	True Negative (TN)

Table 1 shows confusion\_matrix for the generalisation of a given class (e.g., "Grade = 1") :

True Positive (TP): The number of correctly predicted instances of class 1.

False Positive (FP): The number of instances incorrectly predicted as class 1 (but they are not class 1).

False Negative (FN): The number of instances of class 1 that were not predicted correctly (i.e., predicted as another class).

True Negative (TN): The number of correctly predicted instances that do not belong to class 1.

#### Calculation of precision, recall and F1 score:

Precision: measures the proportion of instances predicted as a certain class (e.g., class 1) that are actually that class, and is defined as:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

In this context, if class 1 represents a certain Grade (e.g., "Grade = A"), precision indicates the proportion of students predicted to have this grade who actually achieved it. A high precision suggests that when the model predicts "Grade = A", it is typically accurate.

Recall (Sensitivity or True Positive Rate): measures the proportion of actual instances of a class (e.g., class 1) the model correctly predicted, and is defined as:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

In this dataset, recall for "Grade = A" indicates the proportion of students who actually have an A and were correctly predicted as such by the model. A high recall suggests that the model effectively identifies most of the true instances of "Grade = A."

F1 Score: The F1 score is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall. It is particularly useful when balancing the two is necessary, especially in cases of imbalanced datasets (i.e., when some grades occur more frequently than others).

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

In this context, the F1 score provides a combined measure of the model's performance in predicting a specific grade, balancing both precision (accuracy when predicting a grade) and recall (identifying all instances of that grade). A high F1 score indicates an effective balance between precision and recall.

Support: Support represents the quantity of actual instances for each class within the dataset. It is not a measure of model performance; rather, it indicates the number of examples present in the test set for each class (grade).

For instance, in the classification report:

- Class 1: Support = 6
- Class 2: Support = 6
- Class 4: Support = 0

This indicates:

- There are 6 instances of class 1 (Grade 1).
- There are 6 instances of class 2 (Grade 2).
- There are no instances of class 4 in the test set.

**Practical implementation:**

In order to comprehend the entire operational procedure of each algorithm, the following steps need to be executed. The initial step involves pre-processing, which includes addressing missing values, encoding categorical variables, performing feature scaling, and dividing the dataset into testing and training sets. This particular phase is universal for all algorithms since each algorithm's effectiveness is assessed using the same dataset. Subsequently, each model is formulated, trained, tested, used for predicting test values, utilized for creating a confusion\_matrix, generating a classification report, and producing visualisations. Finally, 5-fold cross-validation is performed it offers a reliable performance estimate by reducing variance, making efficient use of data, providing balanced evaluation. It trains and tests on different data portions, resulting in a more generalized and robust estimate of a model's real-world performance.

**Pre-processing:**

Before feeding the data into machine learning algorithms pre-processing steps are applied:

**Handling missing values:**

Missing data entries, if any, were handled using imputation techniques such as filling with mean/median for numerical columns and mode for categorical columns. The data is collected and pre-processed in a spreadsheet format (Excel) for ease of analysis. It consists of both categorical (Gender, Grade) and numerical variables (Assignment, Seminar, Internal, Attendance, Average).

**Implementation:**

Code to import and display the data from excel file

```
df = pd.read_excel("studentdata.xlsx")
df
```

**Table 2 : shows the dataset used for studentdata**

Roll_no	Gender	Assignment	Seminar	Internal	Attendance	Average	Grade
1	Male	7	6	73	72	73	C
.....							
60	Female	8	7	72	98	85	B

Note : Complete dataset is given in the endnote<sup>i</sup>

Filtering and assigning fields to dependent and independent array variables:

Extracting two arrays X and y from a DataFrame df

```
X = df.iloc[:, 1: -1].values
```

1 means it starts from the second column (python uses zero based indexing, so 1 means the second column), -1 means it stops just before the last column. In our dataset Roll\_no and Grade columns are not included in X

```
y = df.iloc[:, -1].values
```

This selects all rows and only the last column (since -1 refers to the last index in python). In our dataset only Grade column is assigned to y

**Encoding categorical variables:**

The categorical variable Gender was encoded utilizing label encoding (0 for Male, 1 for Female), and the Grade target variable was converted to numerical labels (e.g., A: 0, B: 1, C: 2, etc.). Python employs LabelEncoder from the sklearn.pre-processing module to convert categorical data into numerical form.

```
from sklearn.pre-processing import LabelEncoder
le_gender=LabelEncoder()
le_grade=LabelEncoder()
```

This code creates instances of the LabelEncoder class, designated as le\_gender and le\_grade, which are utilized to transform the values in the 'Gender' and 'Grade' columns of the dataframe, respectively.

```
df['Gender'] = le_gender.fit_transform(df['Gender'])
df['Grade'] = le_grade.fit_transform(df['Grade'])
```

The fit\_transform method performs two functions:

fit: It identifies the unique values in the 'Gender' and 'Grade' columns.

transform: It subsequently converts these categories into their corresponding integer values.

Categorical variables such as 'Gender' and 'Grade' are transformed into numerical values because the models typically cannot interpret non-numeric data directly. By employing LabelEncoder, categorical values are converted to integers, enabling the machine learning model to utilize them in its calculations.

```
X = df[['Gender', 'Assignment', 'Seminar', 'Internal', 'Attendance', 'Average']]
y = df['Grade']
```

X comprises the features (independent variables) that will be utilized to make predictions. In this instance, the features are 'Gender', 'Assignment', 'Seminar', 'Internal', 'Attendance', and 'Average'.

y contains the target (dependent variable) that we are attempting to predict, which is the 'Grade' column.

In the context of a machine learning task,  $X$  would represent the input data fed into the model, and  $y$  would represent the output data (the actual grades) that the model is attempting to learn how to predict.

#### Feature scaling:

Features with large ranges (such as Internal and Attendance) were standardized utilizing z-score normalization to ensure all features contribute equally during model training. To implement feature scaling to our dataset, we can employ either the StandardScaler (for standardization) or MinMaxScaler (for min-max scaling).

Applying Feature Scaling using Scikit-learn with StandardScaler and Initializing the StandardScaler

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
Fit the scaler to the data and transform it
X_scaled = scaler.fit_transform(X)
```

#### Splitting of data for training and testing:

The dataset was partitioned into training and testing sets, with 80% of the data utilized for training and 20% reserved for testing.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
```

This function is employed to divide the dataset into two distinct subsets: a training set and a testing set. This step is crucial in the machine learning workflow to evaluate the model's ability to generalize to unseen data.

The `train_test_split(X, y)` function partitions the feature set  $X$  and the target variable  $y$  into:

$X_{train}$ : The features for the training data (80% of the original dataset).

$X_{test}$ : The features for the testing data (20% of the original dataset).

$y_{train}$ : The target values corresponding to  $X_{train}$ .

$y_{test}$ : The target values corresponding to  $X_{test}$ .

Parameters:

`test_size=0.2`: this parameter specifies that 20% of the dataset will be allocated for testing, while 80% will be used for training.

`random_state=0`: this parameter ensures the reproducibility of the dataset split. The random seed (0 in this instance) controls the shuffling of data prior to partitioning.

### 3.5 Supervised machine learning algorithms:

#### 3.5.1. Logistic regression:

The logistic regression model is an uncomplicated and easy to understand model utilized for both binary and multiclass classification. By employing a logistic function, it predicts the likelihood of a class (such as student grades) based on the association between the input features and the result. It performs effectively in situations where the connection between features and the result is roughly linear and the dataset does not contain intricate nonlinear patterns. The model estimates the relationship between the independent variables and the dependent variable and is mostly used for analysing and explaining the relationship between a binary variable and a series of predicted variables. Logistic regression is developed together with Linear Regression, but they differ in binary variable and continuous variable response. In our dataset, in the context of student performance prediction our target variable ( $y_{train}$ ) is categorical, predicting specific Grade categories (A, B, C, etc.).

#### Model fitting and prediction:

```
from sklearn.linear_model import LogisticRegression
```

imports the logistic regression algorithm from Scikit-learn.

```
Model = LogisticRegression()
initializes the logistic regression model.
```

```
Model.fit(X_train, y_train)
```

trains the model on the training data ( $X_{train}$  for input features,  $y_{train}$  for target labels), establishing the relationship between features and the target outcome.

```
Y_pred = model.predict(X_Test)
```

The `predict()` method is employed to make predictions on the test data ( $X_{Test}$ ), which represents the unseen data, comprising 20% of the original dataset not used for training. The model applies the learned relationships to predict the target values ( $y_{pred}$ ) for the test data.  $Y_{pred}$  will be an array of predicted class labels corresponding to the input features in  $X_{Test}$ .

#### Evaluation of accuracy:

```
from sklearn.metrics import classification_report,
accuracy_score
```

This imports two important functions from Scikit-learn's metrics module to evaluate the performance of the model

```
print ("Accuracy:", accuracy_score (y_Test, y_pred))
```

`accuracy score`: computes the accuracy of the model, which is the ratio of correctly-predicted instances to the total number of instances.

**Classification report:**

```
print ("\nClassification Report:\n",
classification_report (y_Test, y_pred))
```

classification\_report: Provides a detailed breakdown of the model's performance, including precision, recall, F1 score, and support for each class.

Accuracy score (y\_Test, y\_pred) compares the true target values (y\_Test) with the predicted values (y\_pred) and calculates the proportion of correct predictions.

In this instance, the accuracy of the model is 0.5833 (or approximately 58.33%), indicating that the model correctly predicted about 58% of the test data.

```
Print ("\nClassification Report:\n",
classification_report (y_Test, y_pred))
```

**Table 3 : classification report of logistic regression**

Class	Precision	Recall	F1 score	Support
1	0.67	0.33	0.44	6
2	0.62	0.83	0.71	6
4	0.00	0.00	0.00	0
Accuracy	0.58			12
macro avg	0.43	0.39	0.39	12
weighted avg	0.65	0.58	0.58	12

Table 3 illustrates that 1,2,4 are encoded numbers for Grades 'B','C','Pass' each Grade considered as an individual class for which the metrics are calculated, support column illustrates that in the selected fold there are no records of class 4, 6 records of class 1 and 6 records of class 2.

For Class 1 (Grade 1):

Precision = 0.67: students predicted to have Grade 1, 67% actually had this grade.

Recall = 0.33: students who actually had Grade 1, the model correctly predicted 33%.

F1 score = 0.44: this metric balances the precision and recall into a single score.

Support = 6: there are 6 instances of students with Grade 1 in the test set.

Similarly

For Class 2 (Grade 2):

Precision 0.62, Recall 0.83, F1 score 0.71 and support 6

For Class 4 (Grade 4):

Precision, Recall, and F1 score = 0: There were no instances of Grade 4 in the test set; consequently, the model did not predict any.

**Overall Metrics:**

Accuracy: 0.58 The overall accuracy is 58%, indicating that the model correctly predicted the target for 7 out of 12 instances.

**Macro average:**

Precision: 0.43 The unweighted mean of precision across all classes.

Recall: 0.39 The unweighted mean of recall across all classes.

F1 score: 0.39 The unweighted mean of F1 score across all classes.

**Weighted average:**

Precision: 0.65 The weighted mean of precision, accounting for the support (number of instances) for each class.

Recall: 0.58 The weighted mean of recall.

F1 score: 0.58 The weighted mean of F1 score.

**Observations:**

The model demonstrates satisfactory precision and recall for classes 1 and 2; however, class 4 is not predicted, which may be attributed to a lack of examples in the test set or suboptimal model performance for this class. The overall accuracy of 58.33% is moderate but may require improvement, depending on the complexity of the problem. In conclusion, this evaluation step facilitates the measurement of the trained model's performance on unseen data, both in terms of overall accuracy and more detailed class-specific metrics.

**Prediction of new data:**

```
new_data = [[1, 8, 7, 75, 90, 80]]
```

for example: Male, 8 Assignment, 7 Seminar, 75 Internal, 90 Attendance, 80 Average

```
new_pred = model.predict(new_data)
```

```
print ("Predicted Grade (encoded):", new_pred)
```

```
print ("Predicted Grade (decoded):", le Grade.inverse
transform (new_pred))
```

output of the above code

```
Predicted Grade (encoded): [2]
```

```
Predicted Grade (decoded): ['C']
```

```
y_predicted=model.predict(X_Test)
```

this line uses the trained machine learning model (model) to predict the target values (y\_predicted) for the test data (X\_Test). The test data is a portion of the dataset that the model has not seen during training and is used to evaluate its performance.

**Confusion matrix:**

```
from sklearn.metrics import confusion_matrix
```

this imports the confusion\_matrix function from the sklearn.metrics module. A confusion\_matrix is a useful tool to evaluate the performance of classification algorithms, especially for multi class problems.

```
Cm=confusion_matrix(y_Test, y_predicted)
```

here, a confusion\_matrix (cm) is generated. It compares the true labels (y\_Test) with the predicted labels (y\_predicted). The resulting matrix shows how well the model is classifying data into different categories.

**Confusion matrix:**

```
array([[2, 3, 1],
       [1, 5, 0],
       [0, 0, 0]])
```

This is a 3x3 confusion\_matrix, indicating that our classification task has three classes.

The rows represent the actual classes and the columns represent the predicted classes.

Row 1: Class 0 :-The model correctly predicted 2 instances of class 0. It misclassified 3 instances as class 1 and 1 instance as class 2.

Row 2: Class 1:- The model correctly\_predicted 5 instances of class 1. It misclassified 1 instance as class 0.

Row 3: Class 2 :- The model did not correctly\_predict any instances of class 2. No instances were predicted as class 2.

The observation shows that the confusion\_matrix provides correct predictions along the diagonal (top left to bottom right). Misclassifications in the off diagonal cells. It can be particularly helpful for diagnosing issues in classification models.

**Visualisation (Heatmap):**

This code visualizes the confusion\_matrix using a heatmap from the seaborn library, making it easier to interpret the model's performance.

```
Import seaborn as sn
```

this imports the seaborn library, which is a python data visualization library based on matplotlib. It provides high level functions to create informative and attractive statistical graphics.

```
Plt.figure(figsize=(5,3))
```

this creates a new figure (a blank canvas for plotting) with the specified size, where figsize= (5, 3) means the figure will be 5 units wide and 3 units tall. This controls how large the resulting plot will appear.

```
Sn.heatmap(cm, annot=True):
```

this uses seaborn's heatmap function to create a heatmap for the confusion\_matrix (cm), the confusion\_matrix is passed as the data to be visualized.

Annot=True: it ensures that the actual values inside the confusion\_matrix cells are displayed on the heatmap, so we can see the number of correct and incorrect predictions directly.

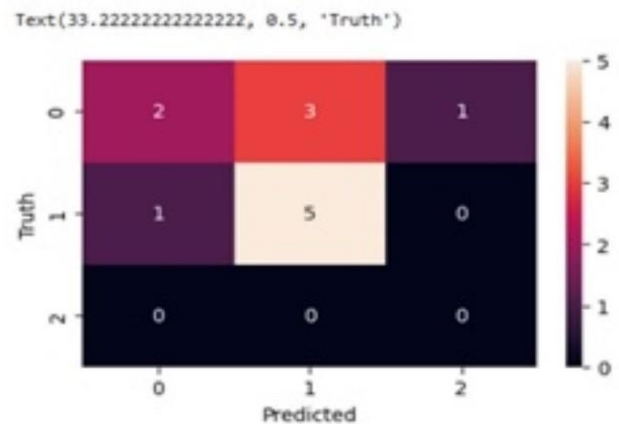
```
Plt.xlabel("Predicted")
```

this sets the label for the x axis of the plot, indicating predicted class labels.

```
Plt.ylabel("Truth")
```

this sets the label for the y axis of the plot, indicating true (actual) class labels.

Figure 2 provides a more visual representation of the confusion\_matrix. The cells are shaded in colors based on their values. Darker colors typically indicate higher values, and lighter colors indicate lower values (depending on the color map). The numbers in the cells represent how many instances were classified correctly or incorrectly. Accurate predictions are noticed in diagonal.



**Figure 2 : Heatmap of logistic regression.**

**3.5.2. K-Nearest Neighbors (KNN):**

KNN is a straightforward algorithm that makes predictions by considering the majority class of the k nearest data points in the feature space. It is simple to implement and performs well with small datasets. KNN is a non-parametric model, which means it does not assume anything about the underlying data distribution. By comparing a student's scores with those of students who have similar assignment and attendance records, KNN can classify a student's grade. This algorithm classifies based on the closest k training examples in the feature space and assigns unlabelled observations to the class of the most similar labelled examples. Characteristics of observations are collected for both training and test dataset.

**Model fitting and prediction:**

```
from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_Test)
```

**Evaluation of accuracy:**

```
accuracy = accuracy score (y_Test, y_pred)
print ("Accuracy:", accuracy)
```

Accuracy: 0.8333333333333334

**Classification report:**

```
print ("\nClassification Report:\n",
classification_report(y_Test, y_pred))
```

**Table 4 : classification report of KNN**

Class	precision	recall	f1 score	Support
0	0.00	0.00	0.00	0
1	0.83	0.83	0.83	6
2	1.00	0.83	0.91	6
accuracy	1.00			12
macro avg	0.61	0.56	0.58	12
weighted avg	0.92	0.83	0.87	12

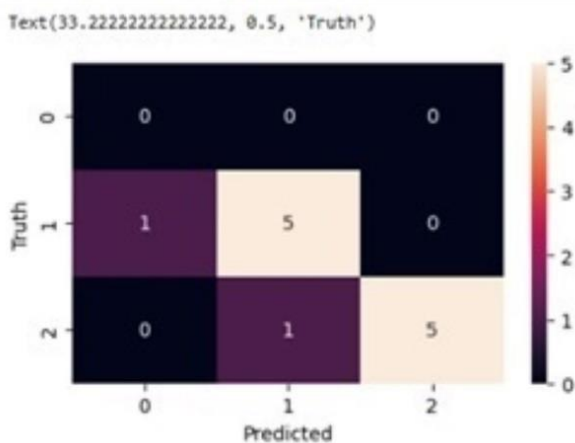
Table 4 **Table 4** illustrates that 0,1,2 are encoded numbers for grades 'A', 'B', 'C' each grade considered as an individual class for which the metrics are calculated, support column illustrates that in the selected fold there are no records of class 0, 6 records of class 1 and 6 records of class 2.

**Confusion matrix:**

```
array ([[0, 0, 0],
[1, 5, 0],
[0, 1, 5]])
```

**Plotting the confusion\_matrix:**

Figure 3 provides a more visual representation of the confusion\_matrix, for detailed information refer Figure 2



**Figure 3 : Heatmap of KNN**

**3.5.3. Decision tree:**

decision trees divide the dataset into branches based on feature values, creating a tree like structure. Each node represents a decision, and the leaves indicate the final prediction. It captures nonlinear relationships and is straightforward to interpret. It is capable of handling both categorical and continuous variables, making it well suited for this dataset, which contains a mix of numeric scores and categorical variables like gender. decision trees can identify intricate patterns, such as how the combination of internal scores and attendance affects a student's grade. It is a supervised machine learning algorithm that employs branching methodology to display all potential outcomes of a decision based on specific parameters. Each tree branch represents one or more outcomes from the original dataset.

**Implementation as follows:**

**Model fitting and prediction:**

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier (random state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_Test)
```

**Evaluation of accuracy:**

```
accuracy = accuracy score (y_Test, y_pred)
print ("Accuracy:", accuracy)
print ("\nClassification Report:\n",
classification_report (y_Test, y_pred))
Accuracy: 1.0
```

**Classification report:**

**Table 5 : Classification Report for decision tree.**

Class	precision	recall	f1 score	Support
1	1.00	1.00	1.00	6
2	1.00	1.00	1.00	6
accuracy	1.00			12
macro avg	1.00	1.00	1.00	12
weighted avg	1.00	1.00	1.00	12

Table 5 illustrates that 1,2 are encoded numbers for grades 'B', 'C' each grade considered as an individual class for which the metrics are calculated, support column illustrates that in the selected fold there are 6 records of class 1 and 6 records of class 2.

**Confusion matrix:**

```
array ([[6, 0],
[0, 6]])
```

**Plotting the confusion\_matrix:**

Figure 4 provides a more visual representation of the confusion\_matrix, for detailed information refer Figure 2

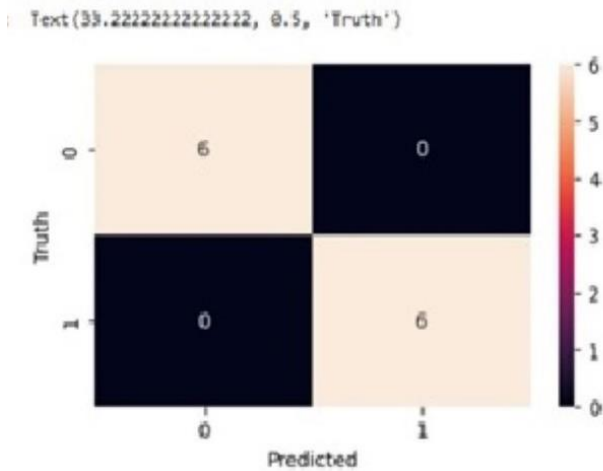


Figure 4 : Heatmap of decision tree

**3.5.4. Random forest:**

The ensemble method called random forest constructs multiple decision trees and then averages their results to enhance accuracy and mitigate overfitting. It is resilient against overfitting and generally yields superior accuracy compared to a single decision tree. random forests are adept at handling intricate datasets with noisy features and can account for interaction effects between variables, such as Seminar participation and Internal scores. Introduced by L. Breiman in 2001, the random forest algorithm excels in both classification and regression by averaging predictions from multiple randomized decision trees. It performs effectively in high dimensional data, adjusts to large-scale scale problems, and provides insights into variable importance. This review focuses on recent advancements in its theory, parameter selection, resampling methods, and variable importance, with the aim of making these concepts more accessible to non-experts.

**Implementation as follows:**

**Model fitting and prediction:**

```
from sklearn.ensemble import
RandomForestClassifier
rf model = RandomForestClassifier()
rf model.fit(X_train, y_train)
y_pred = rf model.predict(X_Test)
```

accuracy, confusion\_matrix, classification report and plotting the confusion\_matrix(heatmap) are same as a decision tree algorithm, for classification report refer Table 5, for heatmap refer Figure 4

**3.5.5. Naive Bayes:**

Bayes theorem forms the basis of Naive Bayes, which assumes that features are independent. This model is probabilistic and works effectively with categorical data, providing quick predictions. It is computationally

efficient and performs well on small datasets. Despite its simplifying assumptions, it can surprisingly perform well, especially when the features are mostly independent. Naive Bayes can make grade predictions by assuming that features like attendance and internal scores contribute independently to the final grade, which is a reasonable approximation in some educational datasets. All input features are considered to be independent of each other. If the conditional assumption of independence holds true, then an NB classifier can converge faster than other models, such as logistic regression.

**Implementation as follows:**

**Model fitting and prediction:**

```
from sklearn.naive bayes import GaussianNB
nb model = GaussianNB()
nb model.fit(X_train, y_train)
y_pred = nb model.predict(X_Test)
```

accuracy, confusion\_matrix, classification report and plotting the confusion\_matrix(heatmap) are same as a decision tree algorithm, for classification report refer Table 5, for heatmap refer Figure 4

**3.5.6. Support vector machines (SVM):**

SVM determines the optimal hyperplane to separate distinct classes, like various grades, by maximizing the margin between them. It has the ability to utilize kernel functions to capture nonlinear associations. SVM performs well in high dimensional spaces and is effective for datasets of small to medium sizes. It is particularly valuable when the data points cannot be separated linearly. In our dataset, SVM can effectively depict the intricate relationships between features, such as Assignment and Internal scores, to forecast whether a student will achieve a high or low grade. A classifier that finds the hyperplane which maximizes the margin between different classes. SVM embodies the concept of systemic risk minimisation. SVMs have been applied to many regression, classification and outlier detection fields.

**Implementation as follows:**

**Model fitting and prediction:**

```
from sklearn.svm import SVC
svm model = SVC (kernel='linear', random state=42)
# 'linear' kernel is used for simplicity
svm model.fit(X_train, y_train)
y_pred = svm model.predict(X_Test)
```

accuracy, confusion matrix, classification report and plotting the confusion\_matrix(heatmap) are same as a decision tree algorithm, for classification report refer Table 5, for heatmap refer Figure 4

### 3.6. K-Fold cross validation:

Cross validation is applied to each of these models to evaluate performance reliably. By splitting the data into 5 folds, each model is trained and tested on different subsets of the data, ensuring that the model's performance is not biased by any particular train test split. This process helps assess the generalisability of each model to unseen student data, reducing the risk of overfitting. These models are being compared in the study to assess their strengths and weaknesses in predicting student performance, allowing us to recommend the most suitable algorithm for educational predictions.

To ensure robustness and avoid overfitting 5-fold cross-validation was applied. In this technique, the dataset is split into 5 equally sized "folds" or subsets. The model is trained on 4 of these folds and tested on the remaining fold. This process is repeated 5 times, with each fold being used as the test set once. The overall performance is the average of the 5 test results. This method helps assess how the model generalizes to unseen data by reducing the variance caused by a single train test split. It is recommended to use 5-fold cross validation with a large number of folds and a small number of repetitions for evaluating the performance of classification algorithms.

#### 5-Fold cross-validation process for splitting data for training and testing:

This code illustrates how 5-fold cross-validation works using the KFold class from the sklearn.model\_selection module, and how to evaluate the performance of a machine learning model using a custom function, get\_score.

```
from sklearn.model_selection import KFold
kf=KFold(n_splits=5)
KFold(n_splits=5, random_state=None, shuffle=False)

this creates a K-Fold cross-validator object with
n_splits=5, meaning the dataset will be split into 5 parts
(folds).

shuffle=False: The data will not be shuffled before
splitting, so it retains its original order.

random_state=None: No random seed is used

def get_score(model, X_train, X_test, y_train, y_test):
model.fit(X_train, y_train) # Fit the model on the
training data
return model.score(X_test, y_test) # Evaluate the model
on the test data

for train_index, test_index in kf.split(range(60)):
    print (train_index, test_index)

kf.split(range(60)): This splits a dataset of size 60
(indexed from 0 to 59) into 5 parts dividing into two
```

sets, one set with 48 records(four folds) used for training purpose and set with 12 records(one fold) used for testing.

For each iteration, the loop gives:

train\_index: The indices used for training (48 records).

test\_index: The indices used for testing (12 records).

#### Evaluating model accuracy:

The function get\_score is used to train a model and calculate its accuracy score for each fold:

```
def get_score(midel, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    return model.score(X_test, y_test)
```

Thus, the KFold splits the data into 5 folds and outputs the indices of the train and test sets for each fold. The get\_score function fits a model on the training set and returns its accuracy on the test set. This setup allows for model evaluation using cross-validation, which helps ensure that the model generalizes well to unseen data.

#### Implementation of algorithms:

The code uses the cross\_val\_score function from sklearn.model\_selection to perform 5-fold cross-validation for different machine learning models on the dataset by importing cross\_val\_score model and evaluates the accuracy scores for each fold and average accuracy

```
from sklearn.model_selection import cross_val_score
```

#### Logistic regression:

```
cross_val_score(LogisticRegression(), X, y)
array ([0.75, 0.58333333, 0.66666667, 0.5,
0.41666667])
Average Accuracy: 0.583
```

#### Decision tree:

```
cross_val_score (DecisionTreeClassifier(), X, y)
array ([1., 1., 0.91666667, 1., 0.91666667])
Average Accuracy: 0.967
```

#### Random forest:

```
cross_val_score(RandomForestClassifier(),X,y)
array ([0.83333333, 1., 0.91666667, 0.91666667, 0.75 ])
Average Accuracy: 0.883
```

#### Support vector classifier (SVC):

```
cross_val_score(SVC(), X, y)
array ([0.41666667, 0.5,0.5, 0.41666667, 0.41666667])
Average Accuracy: 0.45
```

#### K-nearest neighbors (KNN) classifier:

```
cross_val_score(KNeighborsClassifier(), X, y)
array ([0.75 , 0.83333333, 0.66666667, 0.83333333,
0.75 ])
Average Accuracy: 0.767
```

**Gaussian Naive Bayes (GaussianNB) :**

```
cross_val_score(GaussianNB(), X, y)
```

```
array ([0.91666667,0.75,0.83333333, 0.75,
0.83333333])
```

Average Accuracy: 0.817

In conclusion tree based methods (decision tree and random forest) perform the best, followed by GaussianNB and KNN. Linear methods like logistic regression and SVC perform poorly, likely because the relationships in the data are nonlinear.

**3.7. Results and Discussions:**

In this study, a comparative analysis of six supervised learning algorithms logistic regression, decision tree classifier, random forest classifier, SVC, KNN, and Gaussian Naive Bayes was conducted to predict student performance based on a dataset of 60 students with features such as gender, assignment scores, seminar participation, internal marks, attendance, average score, and grade. The 5-fold cross-validation technique was employed to evaluate model performance, and the following key results were obtained:

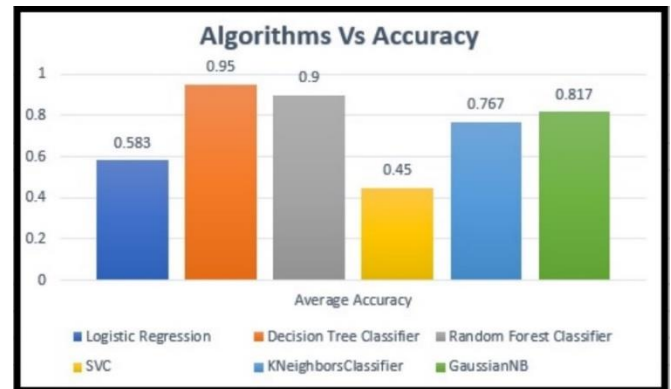
**Table 6 : Prediction of an accuracy of each algorithm.**

Algorithm	Average Accuracy
Logistic regression	0.583
Decision tree classifier	0.967
Random forest classifier	0.883
SVC	0.45
KNN	0.767
GaussianNB	0.817

Table 6 lists all algorithms with their average accuracy after conducting an 5-fold cross-validation technique.

- **Decision tree** has the highest accuracy at 96%, followed closely by **random forest** at 88%. This indicates that tree-based models are better at handling the dataset, likely due to their ability to capture complex decision boundaries.
- **Gaussian Naive Bayes** performs quite well with an accuracy of 81.7%, suggesting that despite its simplicity, it can be effective in this case.
- **K-nearest neighbors** have a reasonable accuracy of 76.7%, which may indicate that the dataset has some local clusters that KNN is able to exploit.
- **Logistic regression** performs moderately at 58.3%. It may struggle with nonlinear relationships in the data, as it's a linear model.
- **SVC** has the lowest accuracy at 45%, implying that it might not be well suited for the dataset, potentially due to its sensitivity to the data distribution or lack of tuning for hyperparameters.

Figure 5 shows the graphical representation of an average accuracy of each algorithm after 5-fold cross-validation process.



**Figure 5 : Accuracy comparison of different supervised algorithms**

This study demonstrates that tree-based models (decision tree and random forest) outperform other classifiers in predicting student performance, likely due to their ability to capture complex feature interactions. GaussianNB and KNN also performed reasonably well, while logistic regression and SVC were less effective, indicating that nonlinear classifiers are better suited for this task.

**Future work could focus on:**

- Hyperparameter tuning for models like SVC and KNN to improve accuracy.
- Incorporating feature importance analysis to better understand the impact of different student characteristics on performance.
- Testing with larger datasets to validate model scalability and generalization across different student populations.

**4. CONCLUSIONS:**

The study presents a comprehensive comparison of various supervised learning algorithms for predicting student performance based on a range of academic and behavioural indicators, including gender, assignment scores, seminar participation, internal marks, attendance, average score, and final grade. The primary finding from the analysis indicates that tree-based models, specifically the decision tree classifier and the random forest classifier, significantly outperform other models, achieving high prediction accuracies of 96% and 88% respectively. This suggests that models capable of handling nonlinear relationships and complex interactions between features are most effective for predicting student performance. While Gaussian Naive Bayes and KNN also demonstrated satisfactory

performance, indicating that even simpler models can produce acceptable results when the dataset is sufficiently informative, logistic regression and SVC performed poorly, with accuracies of 58.3% and 45% respectively, highlighting their limitations in handling this particular dataset due to its complexity and non-linearity. For educational institutions seeking to forecast student performance for interventions or targeted support, tree-based models such as decision trees and random forests provide the most reliable results. Future research could focus on fine tuning hyperparameters, assessing feature importance, and testing on larger datasets to corroborate these findings and enhance practical implementation.

### ACKNOWLEDGEMENT :

I would like to extend my sincere appreciation to several individuals and organizations for their significant contribution to the successful completion of this research study. Foremost, I am grateful to Dr. Ajith Prasad, the Principal and Secretary of JSS SMI UG and PG Studies Dharwad, for providing the inspiration and support necessary for the completion of this research paper.

I also extend my heartfelt thanks to Dr. Suraj Jain, director of JSS SM MCA Institute and the Vice Principal of SMI UG and PG Studies, Dharwad, for their inspiration and support.

Lastly, I express my deep gratitude to my family for their unwavering love and support.

### REFERENCES :

- [1] M. D. & D. C. Reisig, "Using GRE scores and prior GPA to predict academic performance among criminal justice graduate students," *Journal of Criminal Justice Education*, p. 37-59., 2005.
- [2] A. A. S. & S. R. Amida, "Testing the relationships of motivation, time management and career aspirations on graduate students' academic success," *Journal of Applied Research in Higher Education*, p. 1305-1322, 2020.
- [3] L. C. G. C. A. & G. Coromina, "Effect of Background, Attitudinal and Social Network Variables on PhD Students' Academic Performance. A Multimethod Approach1," *Estudios Sobre Educación.*, p. 233-253, 2010.
- [4] M. N. N. H. M. Y. N. Z. K. S. D. S. S. S. M. S. M. & H. L. M. Asiah, "A Review on Predictive Modeling Technique for Student Academic Performance Monitoring," in *MATEC Web of Conferences*, 2019.
- [5] S. S. P. B. V. P. G. G. S. A. D. S. B. N. & S. R. M. Gadde, "Performance Prediction of Students Using Machine Learning Algorithms," *springer nature singapore.*, pp. 405-411, 2022.
- [6] M. & K. K. L. Koutina, "Predicting Postgraduate Students' Performance Using Machine Learning Techniques," p. 159-168, 2011.
- [7] R. Z. G. X. C. S. a. L. Y. B. Guo, "Predicting students performance in educational data mining," in *2015 International Symposium on Educational Technology (ISET)*, pp. 125- 128, 2015, pp.
- [8] A. K. H. J. L. a. I. B. I. A. Najm, "Machine Learning Prediction Approach to Enhance Congestion Control in 5G IoT Environment," *Electronics*, vol. 8, p. 607, 2019.
- [9] G. MeeraGandhi, "Machine learning approach for attack prediction and classification using supervised learning algorithms," *Int. J. Comput. Sci. Commun*, vol. 1, pp. 11465-11484, 2010.
- [10] J. P. a. M. K. J. Han, "Data mining: concepts and techniques," *Elsevier*, 2011.
- [11] S. Gaftandz, "Exploring Online Activities to Predict the Final Grade of Student," *Mathematics*, vol. 10, no. 20,, p. 3758, Oct 2022.
- [12] H. Alharthi, "Machine Learning Techniques to Predict Academic Performance of Health Sciences Students," *IEEE*, Dec 2021.
- [13] A. K. H. A. & A. A. W. Salah Hashim, "Student Performance Prediction Model based on Supervised Machine Learning Algorithms," *Materials Science and Engineering*, 2020.
- [14] L. Al-Alawi, "Using Machine Learning to Predict Factors Affecting Academic Performance: The Case of College Students on Academic Probation," *Education and Information Technologies*, vol. 28, no. 10, p. 12407-32, Mar 2023.
- [15] J. a. P. C. Gajwani, "Students' Performance Prediction Using Feature Selection and Supervised Machine Learning Algorithms," *springer singapore*, pp. 347-54, 2020.
- [16] B. P. X. L. N. J. R. H. X. G. P. & N. K. Jia, "Prediction for Student Academic Performance Using SMNaive Bayes Model," *springer*, p. 712-725, 2019.
- [17] Y. H. Y. A. G. F. C. L. A. H. A. A. & A.-A. R. Baashar, "Evaluation of postgraduate academic performance using artificial intelligence models," *Alexandria Engineering Journal*, 2022.

[18] M. Gopal, "Applied Machine Learning," McGraw-Hill Education, 2018.

[19] S. K. a. P. P. G. Kostopoulos, "Predicting student performance in distance higher education using semi-supervised techniques," in Model and data engineering, ed: Springer, pp. 259-270, 2015.

[20] K. P. Murphy, "Machine learning," a probabilistic perspective: MIT press, 2012.

[21] A. K. H. a. A. M. Humadi, "Student's Success Prediction Model Based on Artificial Neural Networks (ANN) and A Combination of Feature Selection Methods," Journal of Southwest Jiaotong University, p. 54, 2019.

[22] N. T. a. M. S. P. Guleria, Predicting student performance using decision tree classifiers and information gain in 2014 International Conference on Parallel, Distributed and Grid Computing, pp. 126-129, 2014.

35	Male	10	8	76	73	87	B
36	Female	7	6	80	87	79	C
37	Male	10	7	69	89	86	B
38	Female	9	9	75	78	88	B
39	Male	7	6	72	85	76	C
40	Female	6	6	62	71	67	D
41	Male	6	9	68	78	78	C
42	Female	10	9	69	95	93	A
43	Male	8	6	66	76	75	C
44	Female	6	8	67	94	79	C
45	Male	6	6	30	35	48	Fail
46	Female	6	9	68	75	78	C
47	Male	7	9	64	89	82	B
48	Female	9	7	73	76	82	B
49	Male	6	9	60	71	74	C
50	Female	9	6	75	94	84	B
51	Male	7	6	65	82	73	C
52	Female	8	8	71	79	82	B
53	Male	9	7	65	81	81	B
54	Female	9	6	79	77	81	B
55	Male	8	6	64	86	77	C
56	Female	8	6	80	90	83	B
57	Male	8	6	77	74	78	C
58	Female	8	6	76	71	77	C
59	Male	7	9	78	85	86	B
60	Female	8	7	72	98	85	B

1 Complete dataset used in the paper

Roll_no	Gender	Assigment	Seminar	Internal	Attendance	Average	Grade
1	Male	7	6	73	72	73	C
2	Female	10	7	75	100	91	A
3	Male	8	6	74	72	76	C
4	Female	9	6	65	88	80	C
5	Male	7	7	60	81	74	C
6	Female	9	6	76	85	83	B
7	Male	6	6	60	75	68	D
8	Female	9	6	74	80	81	B
9	Male	8	6	67	86	77	C
10	Female	8	9	72	72	83	B
11	Male	8	8	68	78	81	B
12	Female	9	9	73	87	90	B
13	Male	10	7	80	97	92	A
14	Female	10	7	60	100	86	B
15	Male	8	6	50	54	64	D
16	Female	10	8	61	70	82	B
17	Male	9	9	73	79	88	B
18	Female	10	8	60	78	83	B
19	Male	6	5	60	50	59	Pass
20	Female	8	9	68	82	84	B
21	Male	7	7	61	87	76	C
22	Female	6	6	60	65	65	D
23	Male	8	6	68	70	74	C
24	Female	9	7	71	71	80	C
25	Male	10	8	64	99	90	B
26	Female	9	7	75	88	85	B
27	Male	7	6	61	87	73	C
28	Female	9	9	79	98	94	A
29	Male	8	8	69	86	83	B
30	Female	10	8	61	85	85	B
31	Male	4	4	50	40	46	Fail
32	Female	7	9	75	79	83	B
33	Male	7	6	80	71	75	C
34	Female	7	7	70	78	76	C