MACHINE LEARNING

COURSE PROJECT REPORT

(PHASE - 2)

TITLE OF THE PROJECT: STUDENT PERFORMANCE

NAME: ANISH SRIRAM B S

REG NO: 220200048

EMAIL: anishsriram.b-26@scds.saiuniversity.edu.in

ML CATEGORY: REGRESSION

1) <u>INTRODUCTION:</u>

The main goal of this problem is to predict the G3 (final year grade) values of students in secondary education of two Portuguese schools. We are asked to predict the G3 values using the given information like the student's demographics, their social and background-related information, previous grades and information related to their school. Using different machine learning models, the values of G3 are predicted and the accuracy scores of the models are compared.

2) <u>DATASET AND FEATURES:</u>

The student performance dataset contains two distinct datasets in it. One dataset is the student performance in maths course and other is the student performance in Portuguese course. Out of the two datasets, I chose the maths course dataset for the project.

The dataset has 33 features(including the target variable G3) and 395 samples in it. Out of those 33 features, 17 are non-numeric and the rest other are numeric. The dataset does not have any missing values in it. The features of this dataset includes attributes like the student's personal information, family information, interests & hobbies, extra-curricular activities, health, previous grades and their social and background information. From the correlation matrix we come to know that the target variable G3 has a high correlation with the previous grades G1 and G2.

3) METHODS:

Before performing any experiment, the input attributes are stored in X and the target variable(G3) is stored in y. The dataset is split into 75% training and 25% testing. Relevant feature scaling is performed on the data for models to make better predictions.

All the experiments are performed using the default settings of the Scikit-Learn library to produce baseline results.

After performing each experiment, to evaluate and to determine the accuracy of the regression model we compute the R^2 score.

The R^2 score is given by:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

$$SS_{res} = \sum_{i}^{n} (y_i - f(x_i))^2 \quad \text{sum of squares explained by model}$$

$$SS_{tot} = \sum_{i=1}^{n} (y_i - \bar{y})^2 \quad \text{sum of squares around the mean}$$

3.1) LINEAR REGRESSION:

Linear regression is a supervised machine learning model which is used to predict a feature (target variable) based on other features in the dataset by drawing a linear relationship between the target variable and other features. This machine learning model represents a linear relationship between the dependent and independent variables to make predictions.

There are two types of Linear Regression models:

- Univariate: In this type of linear regression model only one feature is used to predict the target variable.

The equation is given by:

$$\hat{y} = \theta_0 + \theta_1 x$$

- Multivariate: In this type of linear regression model more than one feature is used to predict the target variable.

The equation is given by:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

In our case, the model is a multivariate linear regression since more than one feature is used to predict the target variable G3.

The cost function for linear regression is the mean squared error function. The equation is given by:

$$\mathrm{MSE}\,(\mathbf{X}, h_{\boldsymbol{\theta}}) = \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)})^2$$

R^2 SCORE: 0.801706

3.2) SUPPORT VECTOR MACHINES (SVM):

Support Vector Machines (SVM) are robust machine learning algorithms which can perform both regression and classification on linear and non linear data. The SVM finds the optimal hyperplanes to perform regression and classification tasks. In our case, the SVM is used to perform regression. Feature scaling is more important in the context of Support Vector Machines.

There are 3 types of kernels in SVMs (RBF,linear,polynomial).

- SVM(Linear) is used when the data is linearly separable.
- SVM(polynomial) is used when the data is non linear.
- SVM(RBF) also known as Gaussian kernel,creates a non linear combinations of original features and projects them onto higher dimensional space so that data becomes separable by a hyperplane.

R^2 SCORE (RBF) : 0.634897

R^2 SCORE (LINEAR) : 0.80191

R^2 SCORE (POLYNOMIAL) : 0.487093

3.3) DECISION TREES:

Decision Trees are non parametric based machine learning algorithms that are capable of performing both regression and classification. The model represents a tree like structure, where it is used to predict the class in case of classification problem or is used to predict value in case of regression problem. The decision trees are trained using the CART(Classification and Regression Tree) algorithm. Feature scaling is not necessary in the context of Decision Trees.

CART Algorithm:

- The training data is splitted into 2 subsets based on a single feature k and threshold value t_k.
- The pair (k,t_k) is chosen in such a way that it produces a pure subset.
- This process is recursively repeated until max_depth is reached or there is no further split possible.

The cost function of CART algorithm for classification is given by:

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

$$\text{where } \begin{cases} G_{\text{left/right}} \text{ measures the impurity of the left/right subset,} \\ m_{\text{left/right}} \text{ is the number of instances in the left/right subset.} \end{cases}$$

The cost function of CART algorithm for regression is given by:

$$J(k, t_k) = \frac{m_{\text{left}}}{m} \text{MSE}_{\text{left}} + \frac{m_{\text{right}}}{m} \text{MSE}_{\text{right}} \quad \text{where } \begin{cases} \text{MSE}_{\text{node}} = \sum_{i \in \text{node}} \left(\hat{y}_{\text{node}} - y^{(i)}\right)^2 \\ \hat{y}_{\text{node}} = \frac{1}{m_{\text{node}}} \sum_{i \in \text{node}} y^{(i)} \end{cases}$$

The cart algorithm tries to reduce the Gini impurity in case of classification problem and tries to reduce the MSE function in case of regression problem.

R^2 SCORE: 0.806642

3.4) RANDOM FOREST:

Random Forest is a powerful ensemble learning method which is a group of decision trees. It aggregates the prediction of each decision tree to yield the final prediction. This machine learning model uses bagging technique to train data. The model is used for performing both classification and regression tasks. In the case of classification, the final prediction of the model is based on the mode of classes. While in the case of regression, the final prediction of the model is based on the mean of the classes. In our case, the model is used for regression.

R^2 SCORE: 0.875369

3.5) ADA BOOST:

Ada Boost is an ensemble learning method which combines several weak learners to obtain a strong learner. In Ada Boost the predictors are trained sequentially. At each iteration, a new predictor is created to correct its predecessor's underfitted instances. The model assigns higher weights to the underfitted instances thereby making the new predictor to focus more and more on hard cases. This helps in improving the accuracy of the model.

R^2 SCORE: 0.862255

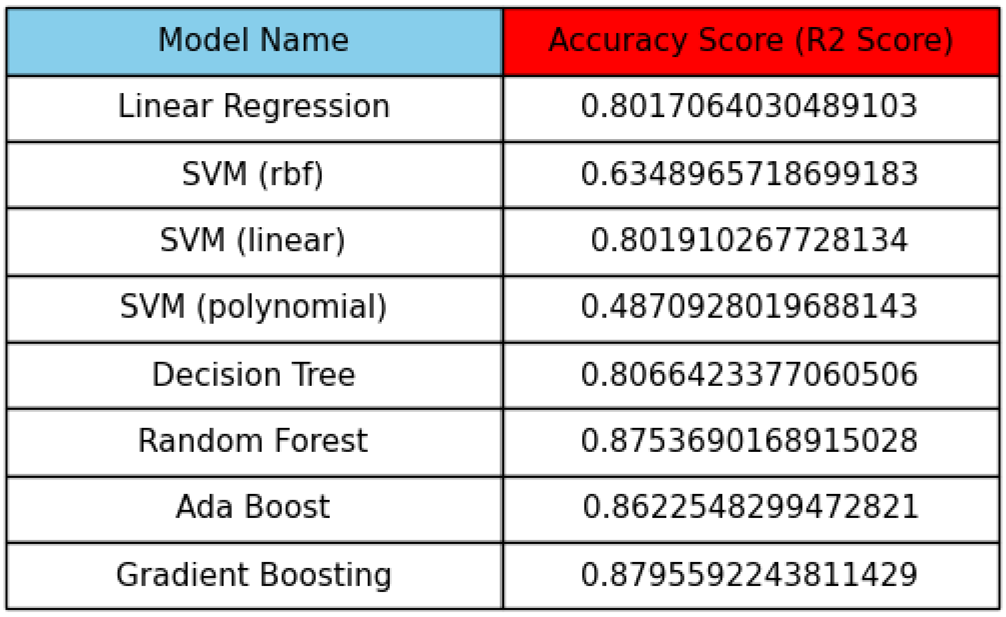
3.6) GRADIENT BOOSTING:

Gradient Boosting is also an ensemble learning method which combines several weak learners to get a strong learner. The model works by sequentially adding predictors based on the residual error. At each iteration, a new predictor is created to correct the residual errors made by the previous predictor. The model reduces the residual error at each iteration thereby minimizing the overall residual error and improving the model's accuracy.

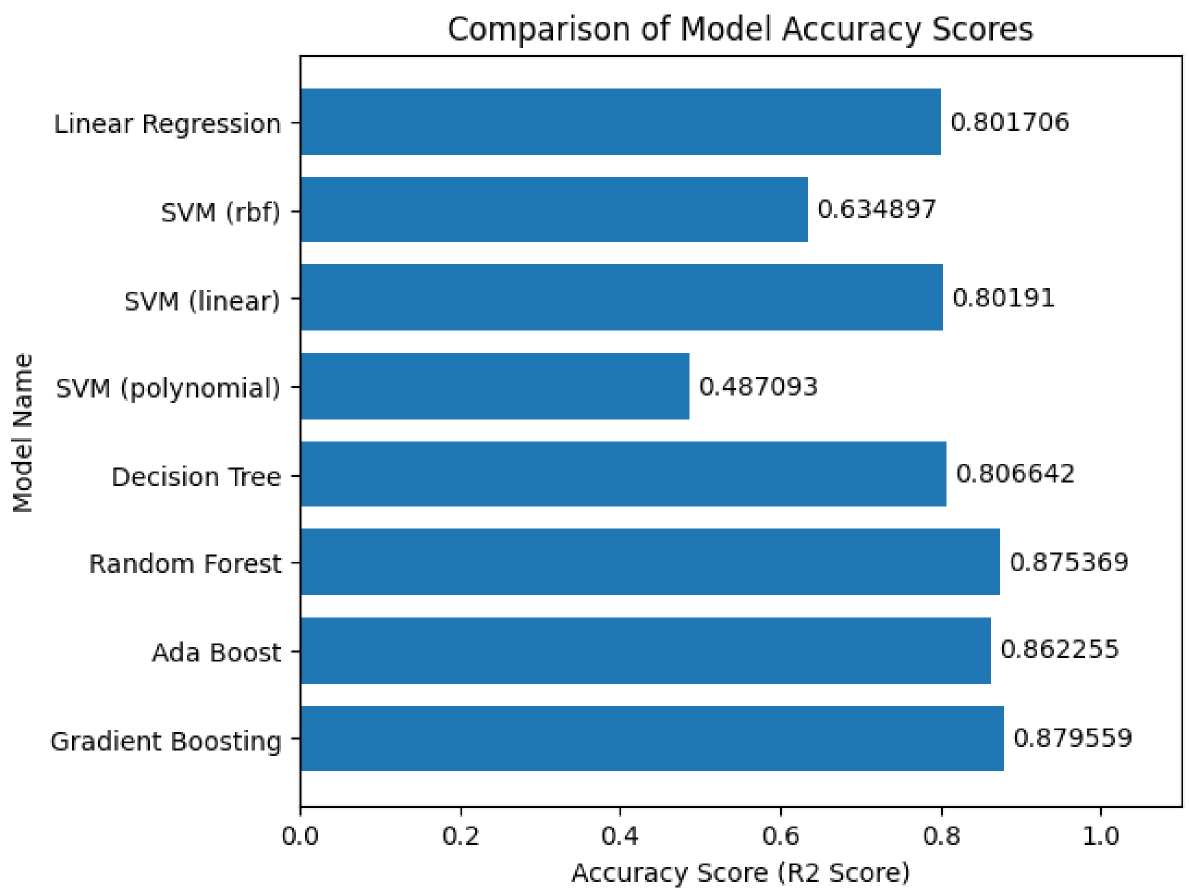
R^2 SCORE: 0.879559

4) RESULTS:

The table below represents the tabular layout of the Accuracy scores (R2 score) of different machine learning models.



Model Name	Accuracy Score (R2 Score)
Linear Regression	0.8017064030489103
SVM (rbf)	0.6348965718699183
SVM (linear)	0.801910267728134
SVM (polynomial)	0.4870928019688143
Decision Tree	0.8066423377060506
Random Forest	0.8753690168915028
Ada Boost	0.8622548299472821
Gradient Boosting	0.8795592243811429



The graph above represents the visualization of Accuracy Scores (R2 Score) of different machine learning models.

From the above results, we can conclude that Gradient Boosting has the highest accuracy score when compared to other regression models. Subsequently to that, Random Forest and Ada Boost have the second and third highest accuracy scores respectively. Among all the models, SVM (polynomial) has the least accuracy score.

5) <u>Hyperparameter Tuning:</u>

The hyperparameter tuning is a method which is used to find the optimal values of hyperparameters for a model. This method improves the accuracy of the model by finding the best values of the hyperparameters for the model. There are two types of techniques for hyperparameter tuning.

- ❖ Grid Search: It uses all combinatorial set of hyperparameter values to find the best values of the hyperparameter for the model.
- ❖ Random Search: It uses random combinatorial set of hyperparameter values to find the best values of the hyperparameter for the model.

The hyperparameter tuning is performed for 5 different regression models. Below are the following regression models for which hyperparameter tuning is done:

- ❖ SVM(RBF)
- ❖ Decision Tree
- ❖ Random Forest
- ❖ Ada Boost
- ❖ Gradient Boosting

Grid Search or Random Search whichever is appropriate for each model is performed on scaled data with 5-fold cross validation. In my case, Grid Search is performed for SVM(RBF) and Ada Boost while Random Search is performed for Decision Tree,Random Forest and Gradient Boosting.

After performing hyperparameter tuning experiment, the results of the experiment are used to train the model and then the R2 score of the model is obtained.

5.1) SVM(RBF):

The hyperparameters tuned in SVM(RBF) are:

- **C:** This hyperparameter is used to determines the amount of margin to be adjusted to find a balance between correct classification of samples and maximizing the margins.
- **gamma:** This hyperparameter is used to control the influence of single sample on the decision boundary.

Grid Search is used to tune the hyperparameters of the SVM(RBF) model.

Parameter Grid:

[{'C': [1, 2, 3, 4, 5, 6, 7, 8, 9], 'gamma': array([0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1])}]

Result obtained for the best configuration:

Best Parameters: {'C': 9, 'gamma': 0.01}

Best Score: 0.8089248160418396

Best Estimator: SVR(C=9, gamma=0.01)

Result for the 25% testing dataset:

R^2 SCORE: 0.786492

5.2) DECISION TREE:

The hyperparameters tuned in Decision Tree are:

- **max_depth:** This hyperparameter is used to determine the maximum depth of the tree.
- **max_features:** This hyperparameter is used to control the maximum number of features that should be considered to split a node.
- **max_leaf_nodes:** This hyperparameter is used to determine the maximum number of leaf nodes in the tree.
- **min_samples_split:** This hyperparameter is used to determine the minimum number of samples needed to split a node.
- **min_samples_leaf:** This hyperparameter is used to determine the minimum number of samples need to be in a leaf node.

Note: random_state parameter is assigned a seed value 42 for the model.

- The random_state parameter is used to determine the randomness of the model. When a seed value is specified for the random_state parameter in the model, the randomness stays constant for multiple runs.

Random Search is used to tune the hyperparameters of the Decision Tree model.

Parameter Distribution:

```
{'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18, 20],
 'max_features': [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32],
 'max_leaf_nodes': [20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40],
 'min_samples_split': [2, 4, 6, 8, 10, 12, 14, 16, 18, 20],
 'min_samples_leaf': [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]}
```

Result obtained for the best configuration:

Best Parameters: {'min_samples_split': 8, 'min_samples_leaf': 7, 'max_leaf_nodes': 20, 'max_features': 28, 'max_depth': 8}

Best Score: 0.8184890874890671

Best Estimator: DecisionTreeRegressor(max_depth=8, max_features=28, max_leaf_nodes=20,min_samples_leaf=7, min_samples_split=8, random_state=42)

Result for the 25% testing dataset:

R^2 SCORE: 0.88353

5.3) RANDOM FOREST:

The hyperparameters tuned in Random Forest are:

- **n_estimators:** This is hyperparameter is used to determine the number of trees used to train the Random Forest model.
- **max_depth:** This hyperparameter is used to determine the maximum depth of the tree.
- **max_features:** This hyperparameter is used to control the maximum number of features that should be considered to split a node.
- **max_leaf_nodes:** This hyperparameter is used to determine the maximum number of leaf nodes in the tree.
- **min_samples_split:** This hyperparameter is used to determine the minimum number of samples needed to split a node.
- **min_samples_leaf:** This hyperparameter is used to determine the minimum number of samples need to be in a leaf node.

Note: random_state parameter is assigned a seed value 42 and n_jobs parameter is assigned a value -1 for the model.

- The random_state parameter is used to determine the randomness of the model. When a seed value is specified for the random_state parameter in the model, the randomness stays constant for multiple runs.
- n_jobs parameter is used to determine the number of CPU cores to be used to train the model. When -1 is specified as value for n_jobs parameter,all the CPU cores will be used to train the model.

Random Search is used to tune the hyperparameters of the Random Forest model.

<u>Parameter Distribution:</u>

```
{'n_estimators': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
 'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18, 20],
 'max_leaf_nodes': [20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40],
 'max_features': [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30,
32],
 'min_samples_split': [2, 4, 6, 8, 10, 12, 14, 16, 18, 20],
 'min_samples_leaf': [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]}
```

<u>Result obtained for the best configuration:</u>

Best Parameters: {'n_estimators': 60, 'min_samples_split': 2, 'min_samples_leaf': 5, 'max_leaf_nodes': 22, 'max_features': 24, 'max_depth': 18}

Best Score: 0.8731178265571466

Best Estimator: RandomForestRegressor(max_depth=18, max_features=24, max_leaf_nodes=22, min_samples_leaf=5, n_estimators=60, random_state=42)

<u>Result for the 25% testing dataset:</u>

R^2 SCORE: 0.884079

5.4) ADA BOOST:

The hyperparameters tuned in Ada Boost are:

- **n_estimators:** This is hyperparameter is used to determine the number of trees used to train the Ada Boost model.
- **learning_rate:** This hyperparameter is used to determine how much should each new estimator learn from its predecessor in the ensemble.

Note: random_state parameter is assigned a seed value 42 for the model.

- The random_state parameter is used to determine the randomness of the model. When a seed value is specified for the random_state parameter in the model, the randomness stays constant for multiple runs.

Grid Search is used to tune the hyperparameters of the Ada Boost model.

Parameter Grid:

```
[{'n_estimators': [10, 30, 50, 70, 90, 110, 130, 150, 170, 190],
  'learning_rate': array([0.15, 0.2 , 0.25, 0.3 , 0.35, 0.4 , 0.45, 0.5 ])}]
```

Result obtained for the best configuration:

Best Parameters: {'learning_rate': 0.25, 'n_estimators': 70}

Best Score: 0.885361729868827

Best Estimator: AdaBoostRegressor(learning_rate=0.25, n_estimators=70, random_state=42)

Result for the 25% testing dataset:

R^2 SCORE: 0.855552

5.5) GRADIENT BOOSTING:

The hyperparameters tuned in Gradient Boosting are:

- **n_estimators:** This is hyperparameter is used to determine the number of trees used to train the Gradient Boosting model.
- **learning_rate:** This hyperparameter is used to determine how much should each new estimator learn from its predecessor in the ensemble.
- **max_depth:** This hyperparameter is used to determine the maximum depth of the tree.

Note: random_state parameter is assigned a seed value 42 for the model.

- The random_state parameter is used to determine the randomness of the model. When a seed value is specified for the random_state parameter in the model, the randomness stays constant for multiple runs.

Random Search is used to tune the hyperparameters of the Gradient Boosting model.

Parameter Distribution:

```
{'n_estimators': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
 'learning_rate': array([0.1, 0.2, 0.3, 0.4, 0.5]),
 'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]}
```

Result obtained for the best configuration:

Best Parameters: {'n_estimators': 90, 'max_depth': 2, 'learning_rate': 0.1}

Best Score: 0.8719185490843626

Best Estimator: GradientBoostingRegressor(max_depth=2, n_estimators=90, random_state=42)

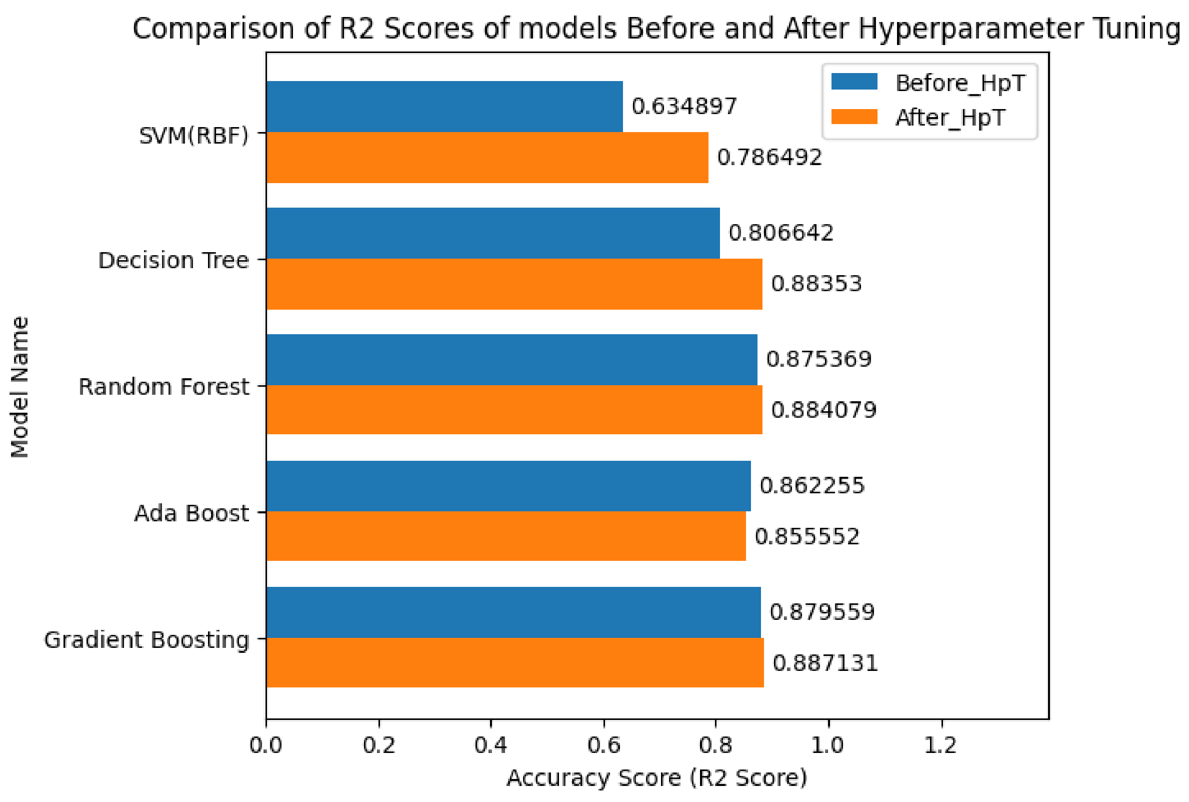
Result for the 25% testing dataset:

R^2 SCORE: 0.887131

6) **RESULTS AFTER HYPERPARAMETER TUNING:**

The table below represents the tabular layout of the comparison of Accuracy scores (R2 score) before and after hyperparameter tuning for different machine learning models.

Model Name	R2 Score (Before Hyperparameter Tuning)	R2 Score (After Hyperparameter Tuning)
SVM(RBF)	0.6348965718699183	0.7864921461037216
Decision Tree	0.8066423377060506	0.8835300244922746
Random Forest	0.8753690168915028	0.8840793240015129
Ada Boost	0.8622548299472821	0.8555518755065709
Gradient Boosting	0.8795592243811429	0.8871306397833539



The graph above represents the visualization of comparison of Accuracy Scores (R2 Score) before and after hyperparameter tuning for different machine learning models.

From the above results, we can conclude that Gradient Boosting has the highest accuracy score after hyperparameter tuning when compared to other regression models. Subsequently to that, Random Forest and Decision Tree have approximately the same accuracy scores. Among all the models, SVM (RBF) has the least accuracy score. While all the models showed some improvement after hyperparameter tuning, Ada Boost did not show any improvement and its accuracy score after hyperparameter tuning is slightly lower than the accuracy score of the model before hyperparameter tuning.

7) <u>FEATURE REDUCTION:</u>

Since my dataset has 32 features(excluding target variable), PCA is applied on scaled data and the dataset is reduced to 16 features.

In my case, the best performing model from section 5 is considered to be Gradient Boosting.As a result, the Gradient Boosting model is trained on the PCA reduced data and the R2 score obtained is 0.6377915746612545.

R^2 SCORE: 0.637792

8) <u>FEATURE SELECTION:</u>

Since my dataset has 32 features(excluding target variable), SelectPercentile method is applied on scaled data to retain top 50% of the highest scoring features.

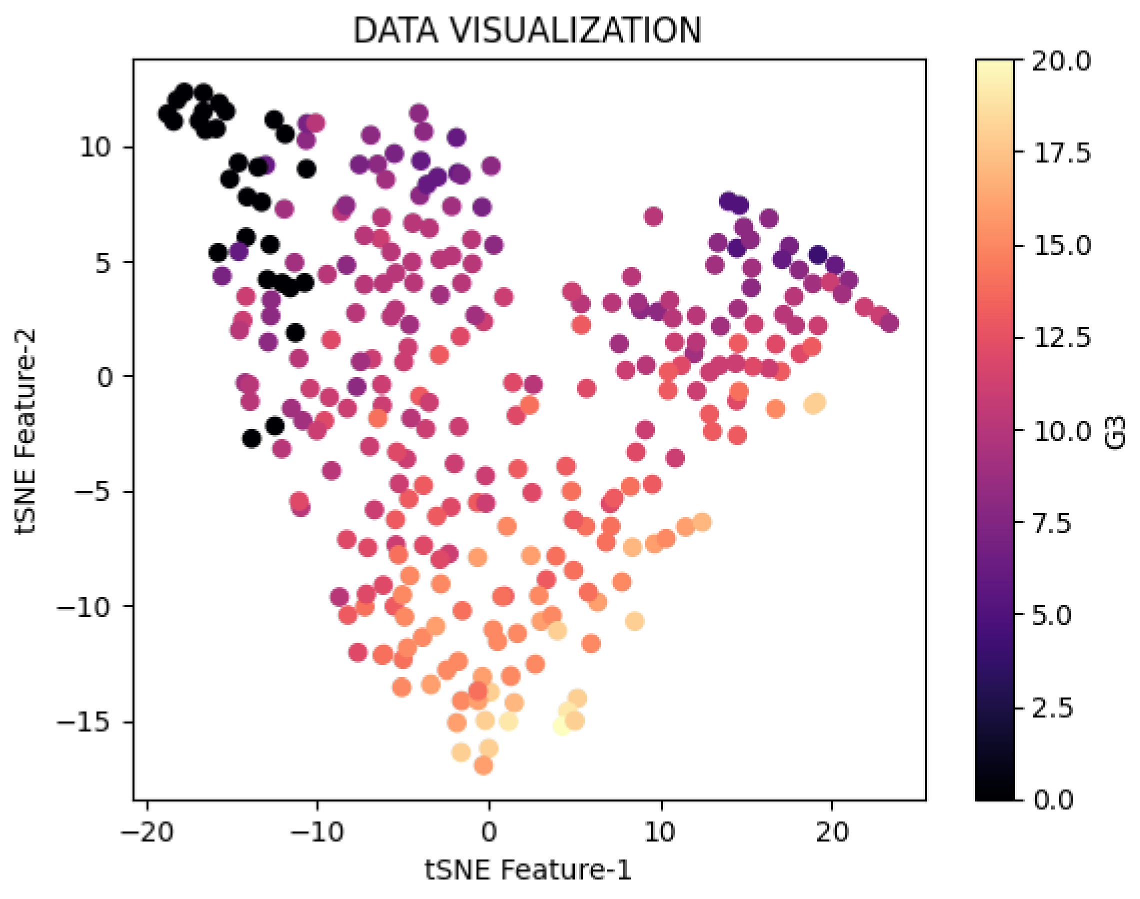
The top 16 features that are chosen are:
['age', 'Medu', 'Fedu', 'Mjob', 'reason', 'traveltime', 'studytime', 'failures', 'paid', 'higher', 'internet', 'romantic', 'goout', 'Dalc', 'G1', 'G2']

In my case, the best performing model from section 5 is considered to be Gradient Boosting. As a result, the Gradient Boosting model is trained on the feature selected data and the R2 score obtained is 0.791959327976655.

R^2 SCORE: 0.791959

9) DATA VISUALIZATION:

Since my dataset has 32 features(excluding target variable), tSNE is applied and the features are reduced to 2D and they are plotted as point cloud.The figure below represents the visualization of training dataset.



10) <u>CONCLUSION:</u>

From the various experiments which we have performed in this project, we can observe the following things:

- ❖ SVM(RBF) model had a score of 0.63 before hyperparameter tuning. But its score was significantly increased to 0.79 after hyperparameter tuning. This evidently shows us that tuning the hyperparameters ‘C’ and ‘gamma’ makes the model much more efficient and better, thus increasing the model’s accuracy score.

- ❖ Decision Tree and Random Forest(which is an ensemble of decision trees) showed improvements and had approximately the same accuracy scores when their key parameters like ‘max_depth’, ‘min_samples_split’, ‘min_samples_leaf’, ‘max_features’, ‘max_leaf_nodes’ etc were tuned.

- ❖ The accuracy score after feature reduction is significantly less than the accuracy score after feature selection. This is because feature reduction reduces and transforms features while feature selection retains the top features which are strongly correlated to the target variable. As a result the model’s accuracy score is greater in feature selection compared to feature reduction.

- ❖ Ada Boost did not show any improvement after hyperparameter tuning. Infact, the model’s accuracy score after hyperparameter tuning is slightly lesser than the model’s accuracy score before hyperparameter tuning. This indicates the sensitivity of the model and the need to adjust hyperparameters to a specific range for better accuracy scores.

- ❖ Overall, Gradient Boosting has the highest accuracy score. It is considered to be the best performing model since it iteratively improves the model accuracy and efficiently handles bias and variance with an ensemble of weak learners.

CITATIONS:

UCI Machine Learning Repository.
archive.ics.uci.edu/dataset/320/student+performance.

Mansur, A. B. F., and N. Yusof. *The Latent of Student Learning Analytic With K-mean Clustering for Student Behaviour Classification*. 2018, api.semanticscholar.org/CorpusID:69596372.

Helwig, Nathaniel E. *Adding Bias to Reduce Variance in Psychological Results: A Tutorial on Penalized Regression*. 2017, api.semanticscholar.org/CorpusID:53974072.

Marcondes, D., et al. *Feature Selection Based on the Local Lift Dependence Scale*. 2017, www.semanticscholar.org/paper/Feature-Selection-based-on-the-Local-Lift-Scale-Marcondes-Simonis/b2af31331cd9199832a503b8901878ad1871343f.

---. www.semanticscholar.org/paper/Data-Mining-of-Students'-Performance-%3A-Turkish-as-a-Oyedotun-Tackie/1010f3d04b6d12e39ef8530e1359fb4ebe293d43.

Karagiannopoulos, M., et al. *A Wrapper for Reweighting Training Instances for Handling Imbalanced Data Sets*. 2007, api.semanticscholar.org/CorpusID:29126030.

Grouped Bar Chart With Labels — Matplotlib 3.9.0 Documentation. matplotlib.org/stable/gallery/lines_bars_and_markers/barchart.html#sphx-glr-gallery-lines-bars-and-markers-barchart-py.

Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. “Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition.” *O’Reilly | Safari*, 2019, www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/.