**Project Report Format**

1. **INTRODUCTION**

1.1 Project Overview

1.2 Purpose

2. **IDEATION & PROPOSED SOLUTION**

2.1 Problem Statement Definition

2.2 Empathy Map Canvas

2.3 Ideation & Brainstorming

2.4 Proposed Solution

3. **REQUIREMENT ANALYSIS**

3.1 Functional requirement

3.2 Non-Functional requirements

**4. PROJECT DESIGN**

4.1 Data Flow Diagrams

4.2 Solution & Technical Architecture

4.3 User Stories

**5. CODING**

**6. RESULTS**

**7. ADVANTAGE & DISADVANTAGE**

**8.CONCLUSION**

**9.FUTURE SCOPE**

**10. APPENDIX**

Source Code

GitHub & Project Video Demo Link

1. **INTRODUCTION**

1.1 Project Overview

Student performance analysis and prediction using datasets has become an essential component of modern education systems. With the increasing availability of data on student demographics, academic history, and other relevant factors, schools and universities are using advanced analytics and [machine learning algorithms](https://www.analyticsvidhya.com/blog/2022/01/machine-learning-algorithms/) to gain insights into student performance and predict future outcomes. This approach helps educators identify areas of improvement, personalize learning experiences, and provide targeted support to struggling students. Furthermore, student performance analysis and prediction can also aid in decision-making processes for school administrators and policymakers, helping them allocate resources more effectively. In this article, we will explore the benefits of using datasets for student performance analysis and prediction and discuss some of the methods and tools used in this field.

1.2 Purpose

This project understands how the student’s performance (test scores) is affected by other variables such as Gender, Ethnicity, Parental level of education, and Lunch and Test preparation course.

The primary objective of higher education institutions is to impart quality education to their students. To achieve the highest level of quality in the education system, knowledge must be discovered to predict student enrollment in specific courses, identify issues with traditional classroom teaching models, detect unfair means used in online examinations, detect abnormal values in student result sheets, and predict student performance. This knowledge is hidden within educational datasets and can be extracted through data mining techniques.

2. **IDEATION & PROPOSED SOLUTION**

2.1 Problem Statement Definition

As an educator or administrator, you want to understand how well your students are performing academically so that you can identify areas for improvement and provide targeted support to help them succeed. However, you may be struggling with time-consuming and manual processes for collecting and analyzing student performance data. This can make it difficult to gain a comprehensive view of student progress and to identify patterns and trends in performance that may require action. Additionally, you may find it challenging to effectively communicate student progress to parents and other stakeholders. Therefore, you are looking for an efficient and effective solution that can automate the process of analyzing student performance data and provide meaningful insights that can be easily shared with others.

2.2 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user’s behaviors and attitudes.

It is a useful tool to helps teams better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user’s perspective along with his or her goals and challenges.

# Example

Sure! Here is an Empathy Map Canvas for student performance analysis:

**Who are we empathizing with?**: Educators/Administrators who are responsible for analyzing student performance data.

**What do they see?**:

* Varying levels of student academic performance
* Time-consuming and manual data collection and analysis processes
* Difficulty in identifying patterns and trends in student performance data
* Inability to provide targeted support due to lack of actionable insights
* Challenges in effectively communicating student progress to parents and other stakeholders

**What do they hear?**:

* Student progress reports from teachers and other educators
* Concerns and feedback from parents and guardians
* Suggestions from other administrators on how to improve student performance
* Frustration and stress from trying to manage and analyze large amounts of data

**What do they say and do?**:

* Collect and analyze student performance data to identify areas for improvement
* Seek out resources and support to help analyze and interpret data effectively
* Work collaboratively with teachers and other educators to identify targeted support for struggling students
* Communicate student progress to parents and other stakeholders using various formats (e.g., reports, meetings, etc.)
* Monitor student progress over time and adjust support strategies as needed

**What do they think and feel?**:

* Concerned about student progress and success
* Overwhelmed by the amount of data to manage and analyze
* Frustrated with time-consuming and manual processes
* Motivated to find new and innovative ways to support student success
* Dedicated to improving student outcomes and achieving academic goals

Overall, the empathy map highlights the need for educators and administrators to have an efficient and effective solution for analyzing student performance data that can provide actionable insights and support targeted interventions for student success.

Top of Form

Regenerate response

* Inability to provide targeted support due to lack of actionable insights
* Challenges in effectively communicating student progress to parents and other stakeholders

**What do they hear?**:

* Student progress reports from teachers and other educators
* Concerns and feedback from parents and guardians
* Suggestions from other administrators on how to improve student performance
* Frustration and stress from trying to manage and analyze large amounts of data

**What do they say and do?**:

* Collect and analyze student performance data to identify areas for improvement
* Seek out resources and support to help analyze and interpret data effectively
* Work collaboratively with teachers and other educators to identify targeted support for struggling students
* Communicate student progress to parents and other stakeholders using various formats (e.g., reports, meetings, etc.)
* Monitor student progress over time and adjust support strategies as needed

**What do they think and feel?**:

* Concerned about student progress and success
* Overwhelmed by the amount of data to manage and analyze
* Frustrated with time-consuming and manual processes
* Motivated to find new and innovative ways to support student success
* Dedicated to improving student outcomes and achieving academic goals

Overall, the empathy map highlights the need for educators and administrators to have an efficient and effective solution for analyzing student performance data that can provide actionable insights and support targeted interventions for student success.

Top of Form

Regenerate response

2.3 Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

# Step-1: Team Gathering, Collaboration and Select the Problem Statement

Great! Here's a possible approach for a team gathering and collaboration session to select the problem statement for student performance analysis:

1. Define the purpose of the session: The purpose of this session is to collaboratively identify and select the most pressing problem statement related to student performance analysis that our team can address through a project or initiative.
2. Review the empathy map canvas: Share the empathy map canvas that we created earlier with the team, and ask them to review it individually for a few minutes. Encourage them to add any additional thoughts, feelings, or actions that they think should be included.
3. Share the problem statements: Based on the empathy map canvas, ask the team to brainstorm and write down as many problem statements related to student performance analysis as they can think of. These can be broad or specific, and should be written on separate sticky notes or index cards.
4. Group and categorize the problem statements: Ask the team to group the problem statements into categories or themes that capture the main issues or challenges related to student performance analysis. This can be done through a group discussion, or by physically moving the sticky notes or index cards into piles based on common themes.

**Step-2: Brainstorm, Idea Listing and Grouping**

1. Brainstorm ideas: Encourage the team to individually brainstorm and write down as many ideas as they can think of for addressing the problem statement. These can be broad or specific, and should be written on separate sticky notes or index cards.
2. Group and categorize the ideas: Once the brainstorming time is up, ask the team to group and categorize the ideas into themes or categories. This can be done through a group discussion or by physically moving the sticky notes or index cards into piles based on common themes.
3. Discuss and refine the ideas: As a group, discuss each idea and identify any potential strengths, weaknesses, or opportunities. Refine and clarify the ideas as necessary to ensure that they align with the problem statement and have the potential to address the key issues and challenges related to student performance analysis.

**Step-3: Idea Prioritization**

1. Discuss and refine the ranking: As a group, discuss the ranking of the ideas and identify any potential strengths, weaknesses, or opportunities associated with each idea. Refine and adjust the ranking as necessary to ensure that the most feasible, impactful, and aligned ideas are at the top of the list.
2. Select the top ideas: Once the ranking has been refined and adjusted, select the top ideas that we will focus on implementing to address the problem statement related to student performance analysis. These should be the ideas that have the highest scores based on the defined criteria and are feasible to implement within our resources and time frame.
3. Define next steps: Once the top ideas have been selected, work together as a team to define the next steps for implementing them, including assigning responsibilities, setting deadlines, and identifying any necessary resources or support.

3. **REQUIREMENT ANALYSIS**

3.1 Functional requirement

Jupyter Notebook

3.2 Non-Functional requirements

Dataset about student details.

**4. PROJECT DESIGN**

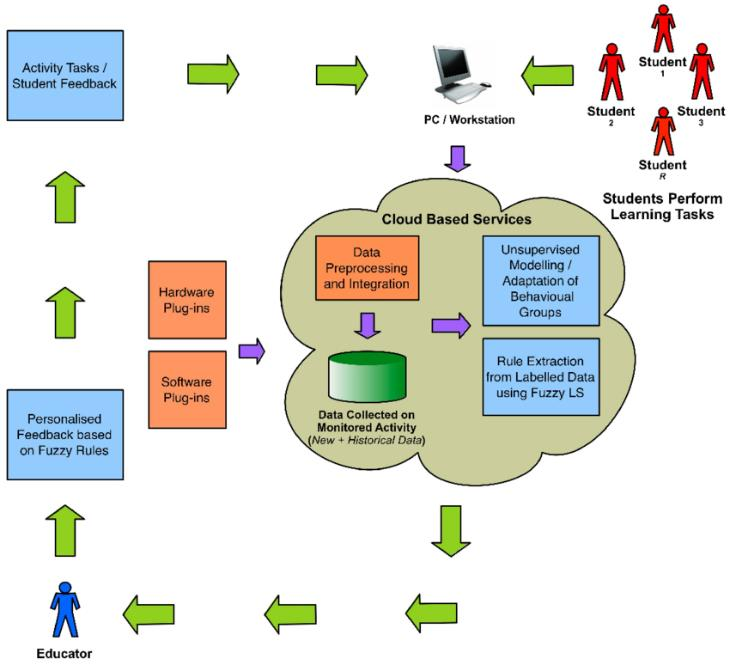
4.1 Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right

amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is

stored.

Example: DFD Level 0 (Industry Standard)



4.2 Solution & Technical Architecture

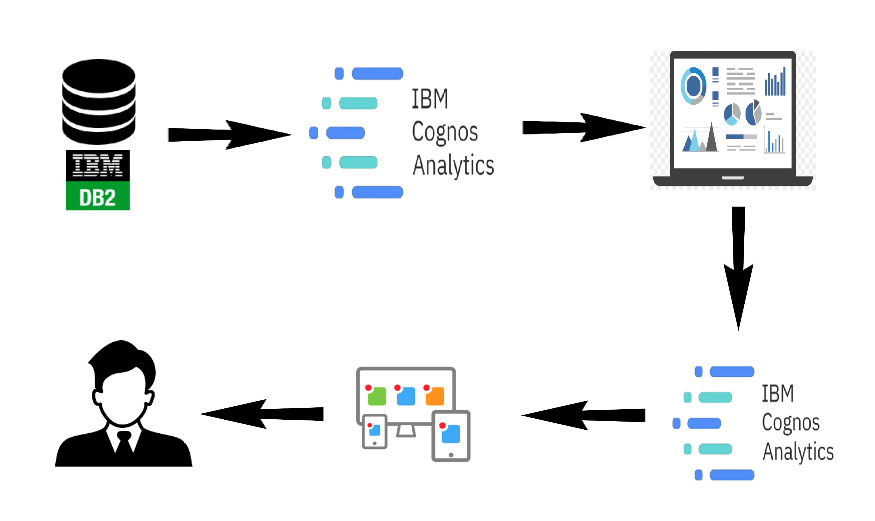
Solution architecture is a complex process – with many sub-processes – that bridges the

gap between business problems and technology solutions. Its goals are to:

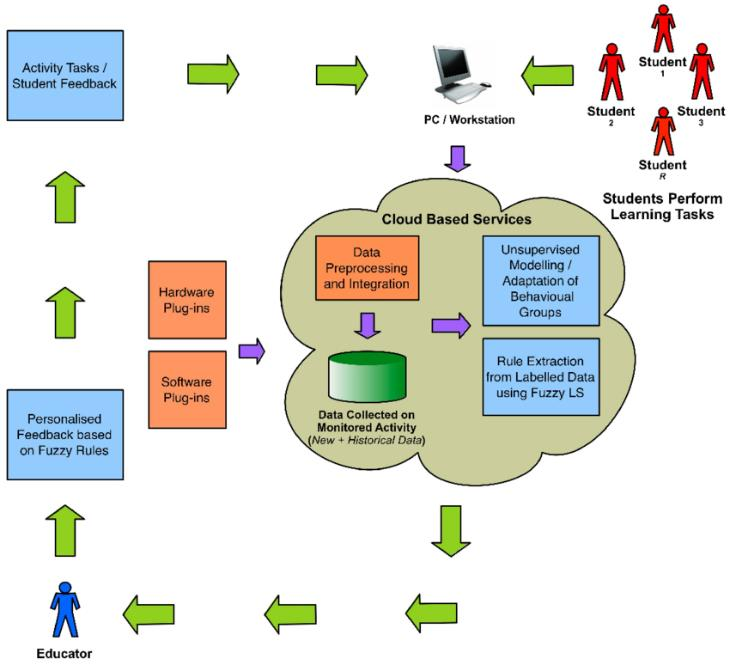
* By integrating data analytics tools to an LMS educators and administrators can gain deeper insights to the student performance Overall LMS with data analytics capabilities can help educators and administrators make more informed decisions about student performance and improve outcome for all students .
* When communicating with the project stake holders about the software project it is clearly important to provide concise description of a structures ,characteristics, behaviour and other aspects of software.
* Feature: ability to track the student progress,analyze student performance data provide personalised feedback and generate reports.
* Development phases: planning, requirement gathering,design development testing and deployment
* Solution Requirements: Includes functional requirements such as ability to track student progress and generate reports as well as non -functional requirements like scalability ,security and useability .

* specifications according to which the solution is defined, managed, and delivered here requirements, technology stack ,architecture ,project management,testing,deployment,documentation.

**Example - Solution Architecture Diagram:**



4.3 User Stories



**5. CODING**

Importing Pandas, Numpy, Matplotlib, Seaborn and Warings Library.

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** warnings

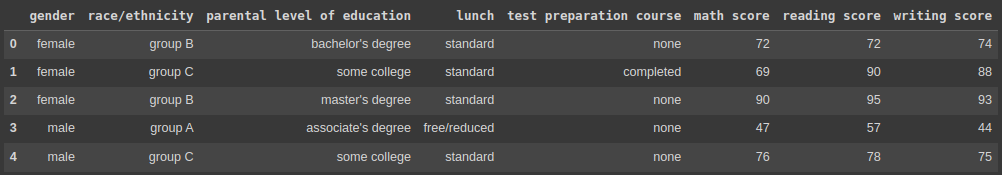
warnings.**filterwarnings**("ignore")

Import the CSV Data as Pandas DataFrame

df = pd.read\_csv("data/StudentsPerformance.csv")

Show the top 5 Recoreds

df.head()



show the top 5 records on the dataset and look at the features.

To see the shape of the dataset

df.shape

https://cdn.analyticsvidhya.com/wp-content/uploads/2023/04/df-shape.png

And it will help to find the shape of the dataset.

## Dataset Information

* gender: sex of students -> (Male/female)
* race/ethnicity: ethnicity of students -> (Group A, B, C, D, E)
* parental level of education: parents’ final education ->(bachelor’s degree, some college, master’s degree, associate’s degree)
* lunch: having lunch before test (standard or free/reduced)
* test preparation course: complete or not complete before test
* math score
* reading score
* writing score

After that, we check the data as the next step. There are a number of categorical features contained in the dataset, including multiple [missing value](https://www.analyticsvidhya.com/blog/2021/10/handling-missing-value/) kinds, duplicate values, check data types, and a number of unique value types.

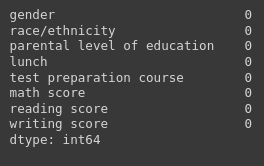
## Data Checks to Perform

* Check Missing values
* Check Duplicates
* Check data type
* Check the number of unique values in each column
* Check the statistics of the data set
* Check various categories present in the different categorical column

## Check Missing Values

To check every column of the missing values or null values in the dataset.

df.isnull().sum()



If there are no missing values in the dataset.

## Check Duplicates

If checking the our dataset has any duplicated values present or not

df.duplicated().sum()

https://cdn.analyticsvidhya.com/wp-content/uploads/2023/04/df-duplicated-sum-1.png

There are no duplicates values in the dataset.

## Check the Data Types

To check the information of the dataset like datatypes, any null values present in the dataset.

#check the null and Dtypes

df.info()

## Data types analysis | student performance

## Check the Number of Unique Values in Each Column

df.nunique()

## Unique values analysis | student performance

## Check Statistics of the Data Set

To examine the dataset’s statistics and determine the data’s statistics.

## Insight

* The numerical data shown above shows that all means are fairly similar to one another, falling between 66 and 68.05.
* The range of all standard deviations, between 14.6 and 15.19, is also narrow.
* While there is a minimum score of 0 for math, the minimums for writing and reading are substantially higher at 10 and 17, respectively.
* We don’t have any duplicate or missing values, and the following codes will provide a good data checking.

## Exploring Data

print("Categories in 'gender' variable: ",end=" ")

print(df["gender"].unique())

print("Categories in 'race/ethnicity' variable: ",end=" ")

print(df["race/ethnicity"].unique())

print("Categories in 'parental level of education' variable: ",end=" ")

print(df["parental level of education"].unique())

print("Categories in 'lunch' variable: ",end=" ")

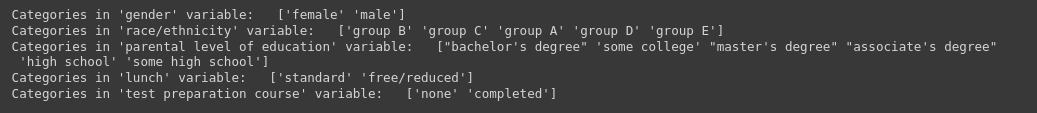
print(df["lunch"].unique())

print("Categories in 'test preparation course' variable: ",end=" ")

print(df["test preparation course"].unique())

The unique values in the dataset will be provided and presented in a pleasant way in the code above.

The output will following:



We define the numerical and categorical columns:

#define numerical and categorical columns

numeric\_features = [feature **for** feature **in** df.columns **if** df[feature].dtype != "object"]

categorical\_features = [feature **for** feature **in** df.columns **if** df[feature].dtype == "object"]

print("We have {} numerical features: {}".format(len(numeric\_features),numeric\_features))

print("We have {} categorical features: {}".format(len(categorical\_features),categorical\_features))

The above code will use separate the numerical and categorical features and count the feature values.

https://cdn.analyticsvidhya.com/wp-content/uploads/2023/04/df-numerical-and-categorical-columns.png

## Exploring Data (Visualization)

#### Visualize Average Score Distribution to Make Some Conclusion

* Histogram
* Kernel Distribution Function (KDE)
* Histogram & KDE

## Gender Column

How is distribution of Gender?

Is gender has any impact on student’s performance?

# Create a figure with two subplots

f,ax=plt.subplots(1,2,figsize=(8,6))

# Create a countplot of the 'gender' column and add labels to the bars

sns.countplot(x=df['gender'],data=df,palette ='bright',ax=ax[0],saturation=0.95)

**for** container **in** ax[0].containers:

ax[0].bar\_label(container,color='black',size=15)

# Set font size of x-axis and y-axis labels and tick labels

ax[0].set\_xlabel('Gender', fontsize=14)

ax[0].set\_ylabel('Count', fontsize=14)

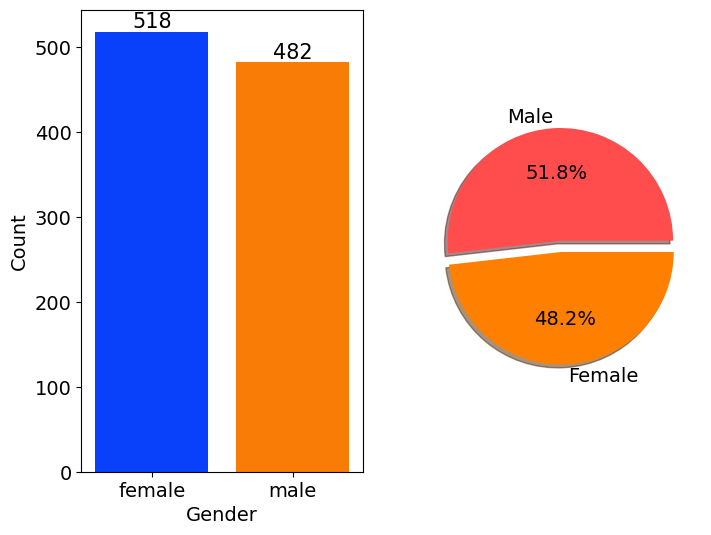
ax[0].tick\_params(labelsize=14)

# Create a pie chart of the 'gender' column and add labels to the slices

plt.pie(x=df['gender'].value\_counts(),labels=['Male','Female'],explode=[0,0.1],autopct='%1.1f%%',shadow=True,colors=['#ff4d4d','#ff8000'], textprops={'fontsize': 14})

# Display the plot

plt.show()



Gender has balanced data with female students are 518 (48%) and male students are 482 (52%)

## Race/Ethnicity Column

# Define a color palette for the countplot

colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']

# blue, orange, green, red, purple are respectiively the color names for the color codes used above

# Create a figure with two subplots

f, ax = plt.subplots(1, 2, figsize=(12, 6))

# Create a countplot of the 'race/ethnicity' column and add labels to the bars

sns.countplot(x=df['race/ethnicity'], data=df, palette=colors, ax=ax[0], saturation=0.95)

**for** container **in** ax[0].containers:

ax[0].bar\_label(container, color='black', size=14)

# Set font size of x-axis and y-axis labels and tick labels

ax[0].set\_xlabel('Race/Ethnicity', fontsize=14)

ax[0].set\_ylabel('Count', fontsize=14)

ax[0].tick\_params(labelsize=14)

# Create a dictionary that maps category names to colors in the color palette

color\_dict = dict(zip(df['race/ethnicity'].unique(), colors))

# Map the colors to the pie chart slices

pie\_colors = [color\_dict[race] **for** race **in** df['race/ethnicity'].value\_counts().index]

# Create a pie chart of the 'race/ethnicity' column and add labels to the slices

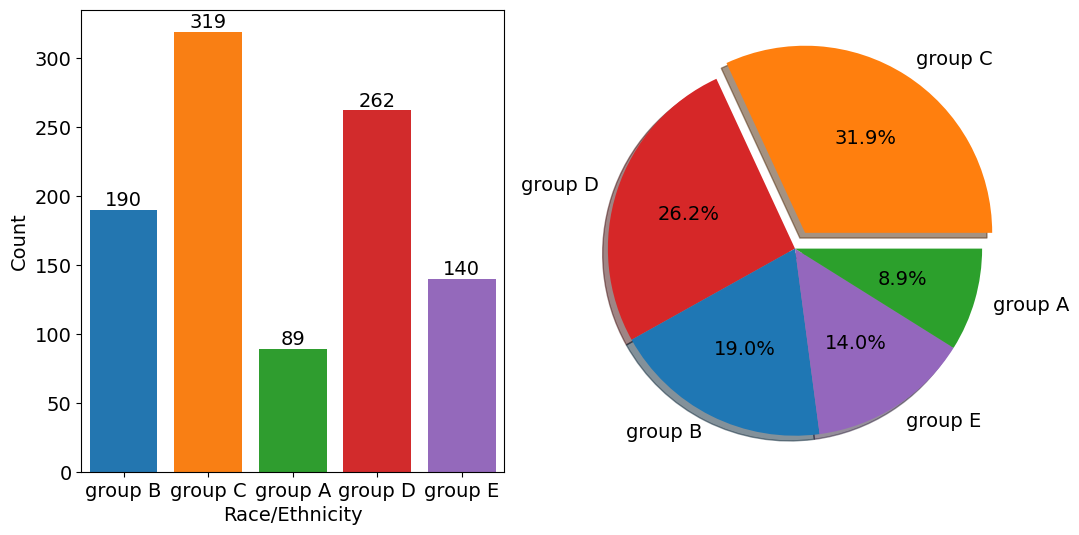
plt.pie(x=df['race/ethnicity'].value\_counts(), labels=df['race/ethnicity'].value\_counts().index, explode=[0.1, 0, 0, 0, 0], autopct='%1.1f%%', shadow=True, colors=pie\_colors, textprops={'fontsize': 14})

# Set the aspect ratio of the pie chart to 'equal' to make it a circle

plt.axis('equal')

# Display the plot

plt.show()



**id = Insights>Insights**

* Most of the student belonging from group C /group D.
* Lowest number of students belong to group A.

## Parental Level of Education Column

plt.rcParams['figure.figsize'] = (15, 9)

plt.style.use('fivethirtyeight')

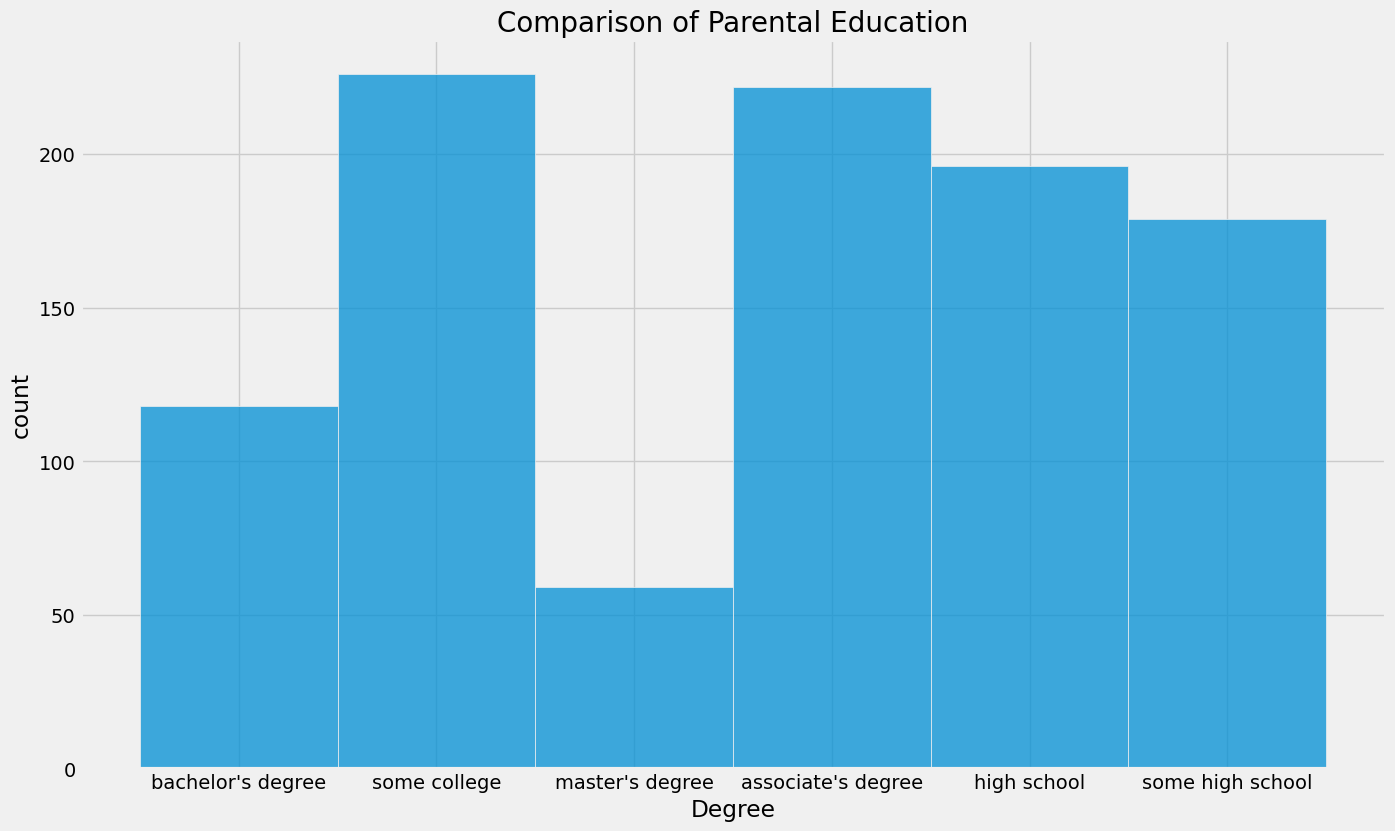
sns.histplot(df["parental level of education"], palette = 'Blues')

plt.title('Comparison of Parental Education', fontweight = 30, fontsize = 20)

plt.xlabel('Degree')

plt.ylabel('count')

plt.show()



**id = Insights>Insights**

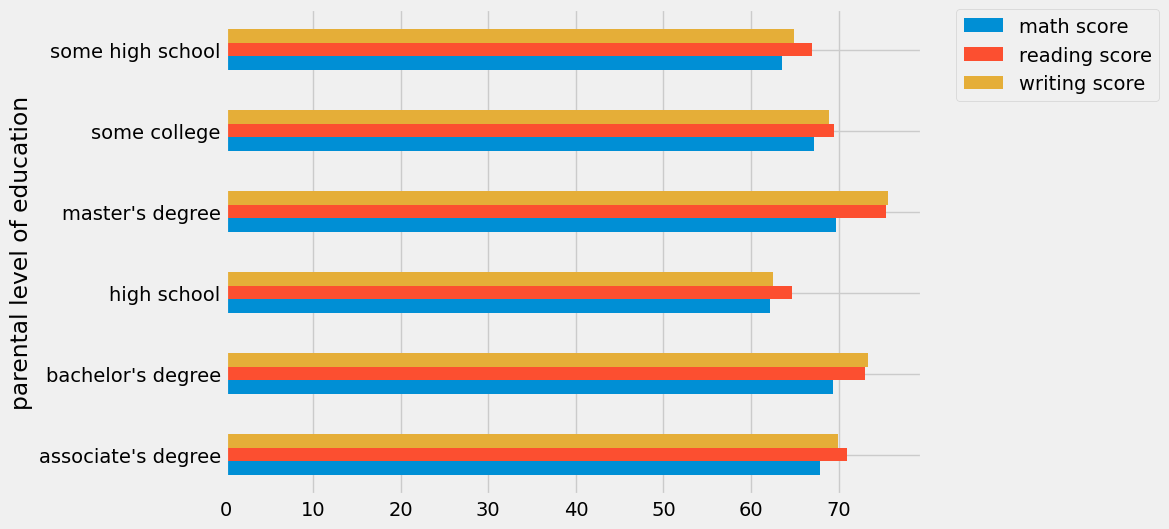
* Largest number of parents are from college.

## Bivariate Analysis

df.groupby('parental level of education').agg('mean').plot(kind='barh',figsize=(10,10))

plt.legend(bbox\_to\_anchor=(1.05, 1), loc=2, borderaxespad=0.)

plt.show()



**id = Insights>Insights**

* The score of student whose parents possess master and bachelor level education are higher than others.

## Maximum Score of Students in All Three Subjects

plt.fig

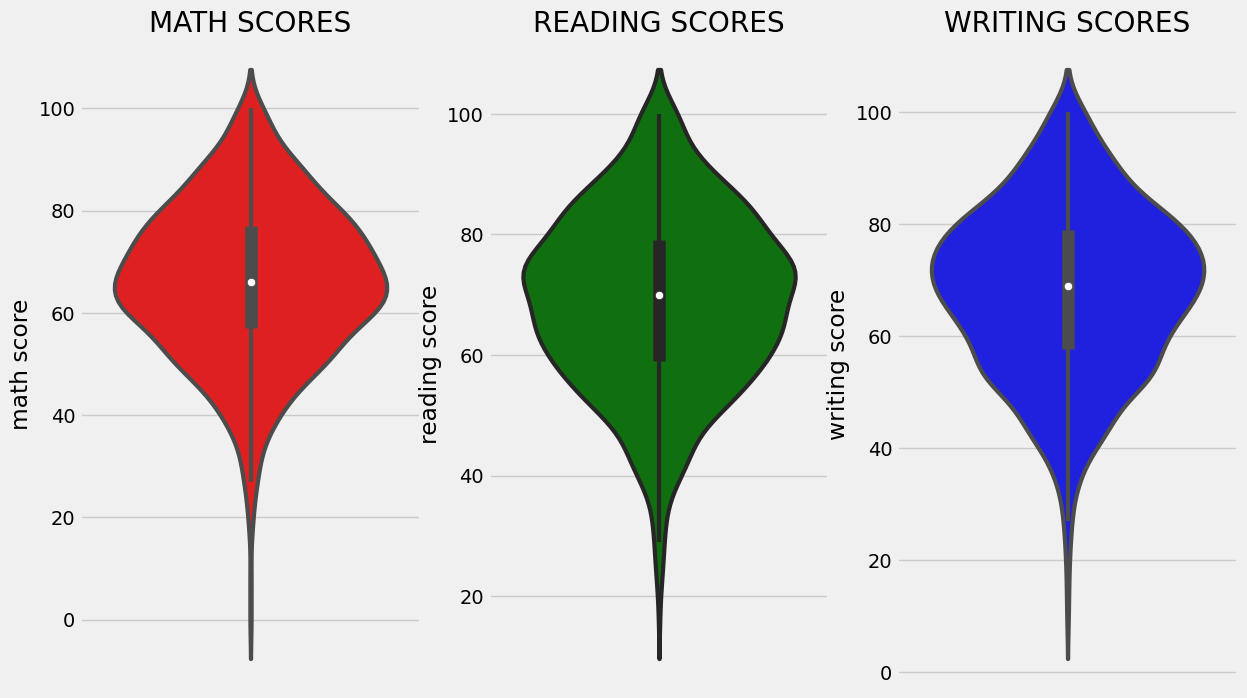
figsize=(18,8))

plt.subplot(1, 4, 1)

plt.title('MATH SCORES')

sns.violinplot(y='math score',data=df,color='red',linewidth=3) plt.subplot(1, 4, 2) plt.title('READING SCORES') sns.vi

plot(y='reading score',data=df,color='green',linewidth=3) plt.subplot(1, 4, 3) plt.title('WRITING SCORES') sns.violinplot(y='writing score',data=df,color='blue',linewidth=3) plt.show()



**id = Insights>Insights**

From the above three plots its clearly visible that most of the students score in between 60-80 in Maths whereas in reading and writing most of them score from 50-80.

## Multivariate Analysis Using Pie Plot

# Set figure size

plt.rcParams['figure.figsize'] = (12, 9)

# First row of pie charts

plt.subplot(2, 3, 1)

size = df['gender'].value\_counts()

labels = 'Female', 'Male'

color = ['red','green']

plt.pie(size, colors=color, labels=labels, autopct='%.2f%%')

plt.title('Gender', fontsize=20)

plt.axis('off')

plt.subplot(2, 3, 2)

size = df['race/ethnicity'].value\_counts()

labels = 'Group C', 'Group D', 'Group B', 'Group E', 'Group A'

color = ['red', 'green', 'blue', 'cyan', 'orange']

plt.pie(size, colors=color, labels=labels, autopct='%.2f%%')

plt.title('Race/Ethnicity', fontsize=20)

plt.axis('off')

plt.subplot(2, 3, 3)

size = df['lunch'].value\_counts()

labels = 'Standard', 'Free'

color = ['red', 'green']

plt.pie(size, colors=color, labels=labels, autopct='%.2f%%')

plt.title('Lunch', fontsize=20)

plt.axis('off')

# Second row of pie charts

plt.subplot(2, 3, 4)

size = df['test preparation course'].value\_counts()

labels = 'None', 'Completed'

color = ['red', 'green']

plt.pie(size, colors=color, labels=labels, autopct='%.2f%%')

plt.title('Test Course', fontsize=20)

plt.axis('off')

plt.subplot(2, 3, 5)

size = df['parental level of education'].value\_counts()

labels = 'Some College', "Associate's Degree", 'High School', 'Some High School', "Bachelor's Degree", "Master's Degree"

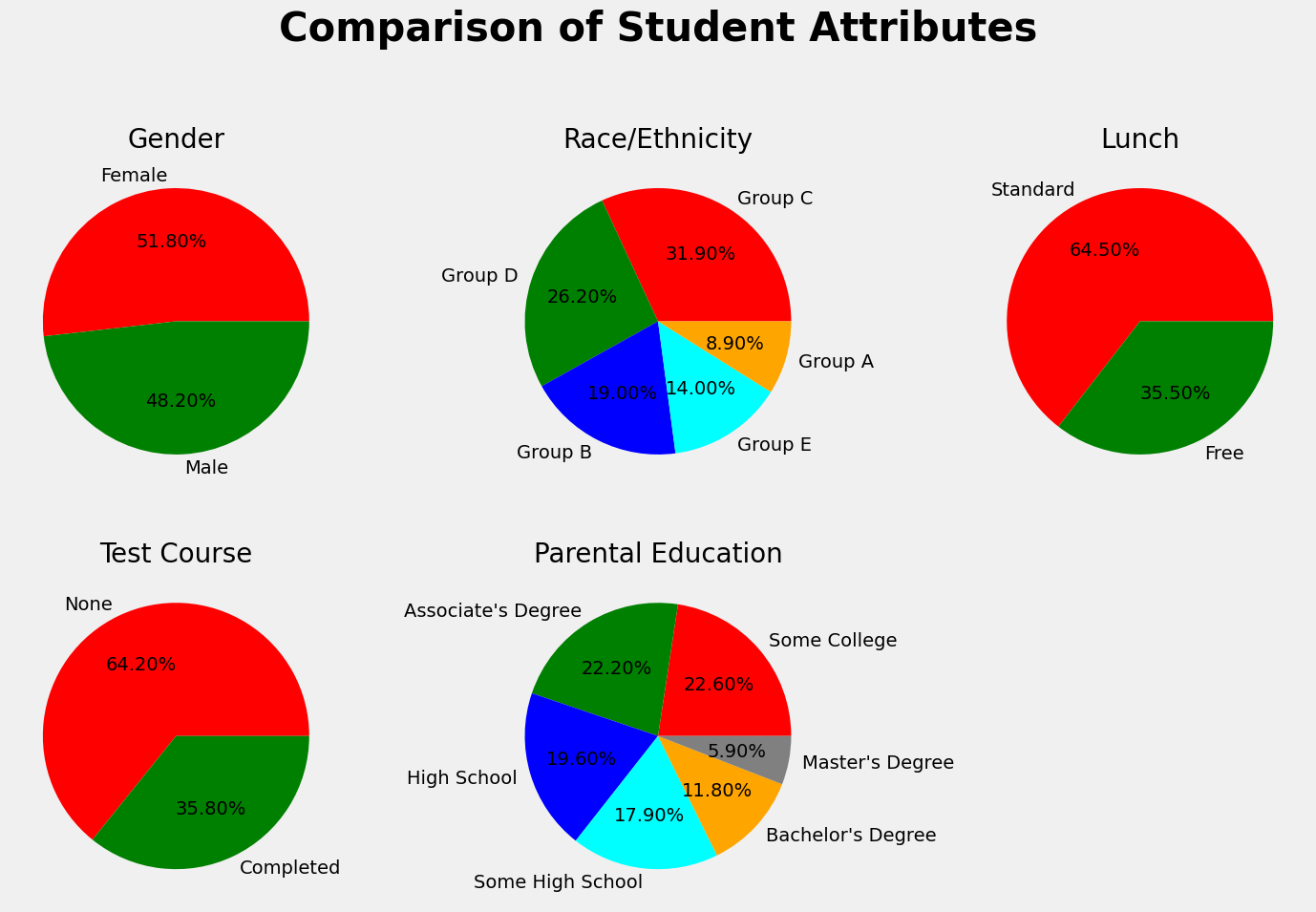
color = ['red', 'green', 'blue', 'cyan', 'orange', 'grey']

plt.pie(size, colors=color, labels=labels, autopct='%.2f%%')

plt.title('Parental Education', fontsize=20)

plt.axi

ff') # Remove extra subplot plt.subplot(2, 3, 6).remove() # Add super title plt.suptitle('Comparison of Student Attributes', fontsize=20, fontweight='bold') # Adjust layout and show plot # This is removed as there are only 5 subplots in this figure and we want to arrange them in a 2x3 grid. # Since there is no 6th subplot, it is removed to avoid an empty subplot being shown in the figure. plt.tight\_layout() plt.subplots\_adjust(top=0.85) plt.show()



**id = Insights>Insights**

* The number of Male and Female students is almost equal.
* The number of students is higher in Group C.
* The number of students who have standard lunch is greater.
* The number of students who have not enrolled in any test preparation course is greater.
* The number of students whose parental education is “Some College” is greater followed closely by “Associate’s Degree”.

From the above plot, it is clear that all the scores increase linearly with each other.

Student’s Performance is related to lunch, race, and parental level education.

* Females lead in pass percentage and also are top-scorers.
* Student Performance is not much related to test preparation course.
* The finishing preparation course is beneficial.

## Model Training

Import Data and Required Packages

Importing [scikit library algorithms](https://www.analyticsvidhya.com/blog/2021/07/15-most-important-features-of-scikit-learn/" \t "_blank) to import regression algorithms.

# Modelling

**from** sklearn.metrics **import** mean\_squared\_error, r2\_score

**from** sklearn.neighbors **import** KNeighborsRegressor

**from** sklearn.tree **import** DecisionTreeRegressor

**from** sklearn.ensemble **import** RandomForestRegressor,AdaBoostRegressor

**from** sklearn.svm **import** SVR

**from** sklearn.linear\_model **import** LinearRegression,Lasso

**from** sklearn.metrics **import** r2\_score, mean\_absolute\_error, mean\_squared\_error

**from** sklearn.model\_selection imp

**RandomizedSearchCV** **from** catboost **import** **CatBoostRegressor** **from** xgboost **import** **XGBRegressor** **import** warnings

## Splitting the X and Y Variables

This separation of the dependent variable(y) and independent variables(X) is one the most important in our project we use the math score as a dependent variable. Because so many students lack in math subjects it will almost 60% to 70% of students in classes 7-10 students are fear of math subjects that’s why I am choosing the math score as a dependent score.

It will use to improve the percentage of math scores and increase the grad f students and also remove fear in math.

X = df.drop(columns="math score",axis=1)

y = df["math score"]

## Create Column Transformer with 3 Types  of Transformers

num\_features = X.select\_dtypes(exclude="object").columns

cat\_features = X.select\_dtypes(**include**="object").columns

from sklearn.preprocessing import OneHotEncoder,StandardScaler

from sklearn.compose import ColumnTransformer

numeric\_transformer = StandardScaler()

oh\_transformer = OneHotEncoder()

preprocessor = Column

transformer( [ ("OneHotEncoder", oh\_transformer, cat\_features), ("StandardScaler", numeric\_transformer, num\_features), ] ) X = preprocessor.fit\_transform(X)

## Separate Dataset into Train and Test

To separate the dataset into train and test to identify the training size and testing size of the dataset.

**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=42)

X\_train.shape, X\_test.shape

https://cdn.analyticsvidhya.com/wp-content/uploads/2023/04/X-train-shape-X-test-shape-300x38.png

## Create an Evaluate Function for Model Training

This function is use to evaluate the model and build a good model.

def **evaluate\_model**(true, predicted):

mae = mean\_absolute\_error(true, predicted)

mse = mean\_squared\_error(true, predicted)

rmse = np.sqrt(mean\_squared\_error(true, predicted))

r2\_square = r2\_score(true, predicted)

**return** mae, mse, rmse, r2\_square

To create a models variable and form a dictionary formate.

models = {

"Linear Regression": LinearRegression(),

"Lasso": Lasso(),

"K-Neighbors Regressor": KNeighborsRegressor(),

"Decision Tree": DecisionTreeRegressor(),

"Random Forest Regressor": RandomForestRegressor(),

"Gradient Boosting": GradientBoostingRegressor(),

"XGBRegressor": XGBRegressor(),

"CatBoosting Regressor": CatBoostRegressor(verbose=False),

"AdaBoost Regressor": AdaBoostRegressor()

}

model\_list = []

r2\_list =[]

**for** i **in** range(len(list(models))):

model = list(models.values())[i]

model.fit(X\_train, y\_train) # Train model

# Make predictions

y\_train\_pred = model.predict(X\_train)

y\_test\_pred = model.predict(X\_test)

# Evaluate Train and Test dataset

model\_train\_mae, model\_train\_mse, model\_train\_rmse, model\_train\_r2 = evaluate\_model(y\_train, y\_train\_pred)

model\_test\_mae, model\_test\_mse, model\_test\_rmse, model\_test\_r2 = evaluate\_model(y\_test, y\_test\_pred)

print(list(models.keys())[i])

model\_list.append(list(models.keys())[i])

print('Model performance for Training set')

print("- Root Mean Squared Error: {:.4f}".format(model\_train\_rmse))

print("- Mean Squared Error: {:.4f}".format(model\_train\_mse))

print("- Mean Absolute Error: {:.4f}".format(model\_train\_mae))

print("- R2 Score: {:.4f}".format(model\_train\_r2))

print('----------------------------------')

print('Model performance for Test set')

print("- Root Mean Squared Error: {:.4f}".format(model\_test\_rmse))

print("- Mean Squared Error: {:.4f}".format(model\_test\_rmse))

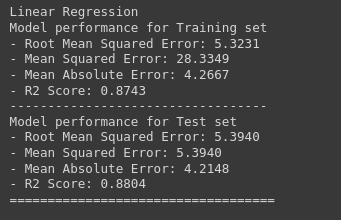
print("- Mean Absolute Error: {:.4f}".format(model\_test\_mae))

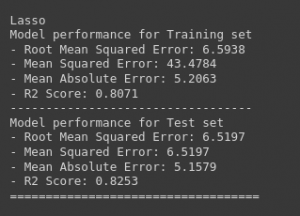
print("- R2 Score: {:.4f}".

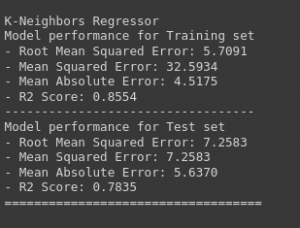
**for**

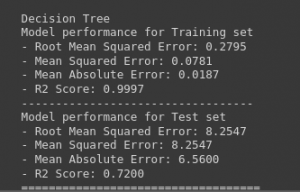
model\_test\_r2)) r2\_list.append(model\_test\_r2) print('='\*35) print('\n')

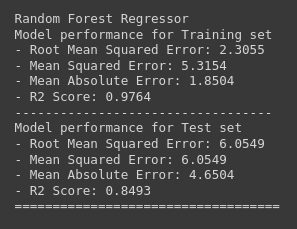
The output of before tuning all algorithms’ hyperparameters. And it provides the RMSE, MSE, MAE, and R2 score values for training and test data.

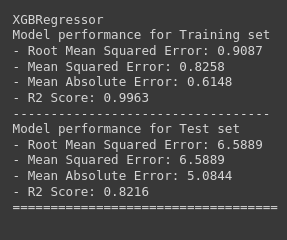


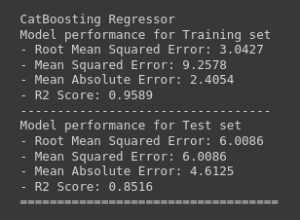


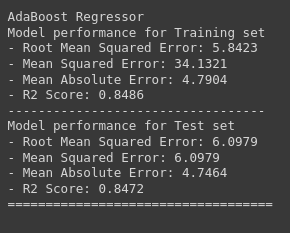












## Hyperparameter Tuning

It will give the model with most accurate predictions and improve prediction accuracy.

This will give the optimized value of [hyperparameters](https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/), which maximize your model predictive accuracy.

**from** sklearn.model\_selection **import** GridSearchCV, RandomizedSearchCV

**from** sklearn.metrics **import** make\_scorer

# Define hyperparameter ranges for each model

param\_grid = {

"Linear Regression": {},

"Lasso": {"alpha": [1]},

"K-Neighbors Regressor": {"n\_neighbors": [3, 5, 7],},

"Decision Tree": {"max\_depth": [3, 5, 7],'criterion':['squared\_error', 'friedman\_mse', 'absolute\_error', 'poisson']},

"Random Forest Regressor": {'n\_estimators': [8,16,32,64,128,256], "max\_depth": [3, 5, 7]},

"Gradient Boosting": {'learning\_rate':[.1,.01,.05,.001],'subsample':[0.6,0.7,0.75,0.8,0.85,0.9],

'n\_estimators': [8,16,32,64,128,256]},

"XGBRegressor": {'depth': [6,8,10],'learning\_rate': [0.01, 0.05, 0.1],'iterations': [30, 50, 100]},

"CatBoosting Regressor": {"iterations": [100, 500], "depth": [3, 5, 7]},

"AdaBoost Regressor": {'learning\_rate':[.1,.01,0.5,.001],'n\_estimators': [8,16,32,64,128,256]}

}

model\_list = []

r2\_list =[]

**for** model\_name, model **in** models.items():

# Create a scorer object to use in grid search

scorer = make\_scorer(r2\_score)

# Perform grid search to find the best hyperparameters

grid\_search = GridSearchCV(

model,

param\_grid[model\_name],

scoring=scorer,

cv=5,

n\_jobs=-1

)

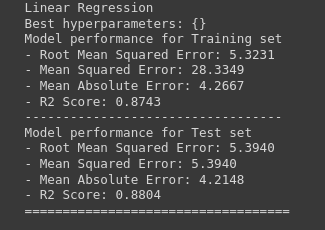
# Train the model with the best hyperparameters

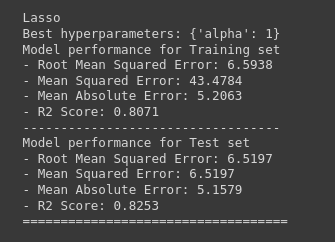
rid\_search.fit(X\_train, y\_train) # Make predictions y\_train\_pred = grid\_search.predict(X\_train) y\_test\_pred = grid\_search.predict(X\_test) # Evaluate Train **and** Test dataset model\_train\_mae, model\_train\_mse, model\_train\_rmse, model\_train\_r2 = evaluate\_model(y\_train, y\_train\_pred) model\_test\_mae, model\_test\_mse, model\_test\_rmse, model\_test\_r2 = evaluate\_model(y\_test, y\_test\_pred) print(model\_name) model\_list.append(model\_name) print('Best hyperparameters:', grid\_search.best\_params\_) print('Model performance for Training set') print("- Root Mean Squared Error: {:.4f}".format(model\_train\_rmse)) print("- Mean Squared Error: {:.4f}".format(model\_train\_mse)) print("- Mean Absolute Error: {:.4f}".format(model\_train\_mae)) print("- R2 Score: {:.4f}".format(model\_train\_r2)) print('----------------------------------') print('Model performance for Test set') print("- Root Mean Squared Error: {:.4f}".format(model\_test\_rmse)) print("- Mean Squared Error: {:.4f}".format(model\_test\_rmse)) print("- Mean Absolute Error: {:.4f}".format(model\_test\_mae)) print("- R2 Score: {:.4f}".**for**

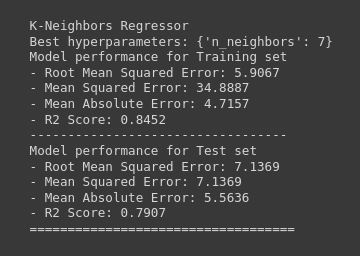
model\_test\_r2)) r2\_list.append(model\_test\_r2) print('='\*35) print('\n')

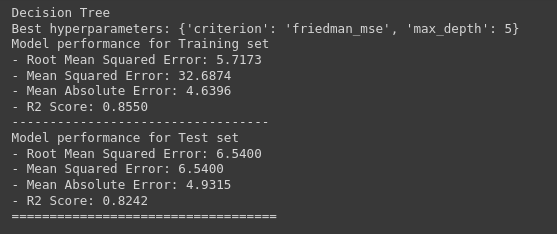
**6.RESULTS**

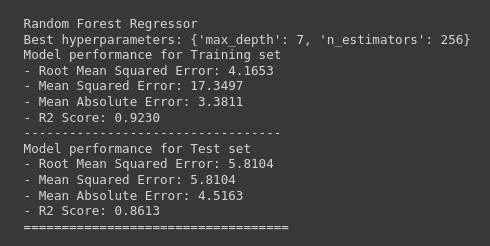
The output of after tuning all algorithms’ hyperparameters. And it provides the RMSE, MSE, MAE, and R2 score values for training and test data.

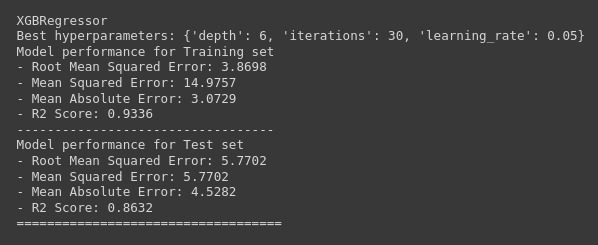


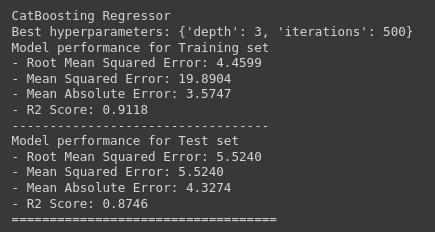


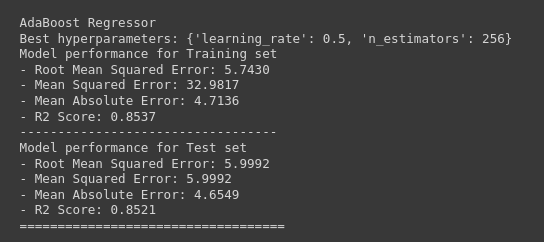












If we choose Linear regression as the final model because that model will get a training set r2 score is 87.42 and a testing set r2 score is 88.03.

**7. ADVANTAGE & DISADVANTAGE**

**Better Management of Student Data**  
A student management system is also a student [database management system](https://leadschool.in/blog/keep-information-at-your-fingertips-with-the-student-database-management-system-give-your-school-a-revamp-with-lead/). It helps you manage all the student-related data in a well-organized manner.

A Student Management System gives you a unique ID against every student. And using that ID, you can easily track the fee status, assignments, exam results, grades, parent info within seconds.

**Improves Overall Teacher Productivity**  
Similar to the students, the teachers also become more productive with a student management system. It’s because, with a system in place, teachers won’t have to manually take attendance, manage timetables and assignments.

Everything can be loaded on the [management software](https://leadschool.in/blog/how-does-the-student-management-system-help-with-multiple-tasks-why-is-it-the-most-effective-tool-to-run-a-school-seamlessly/), which can be accessed by all. This way, teachers can focus more on academics and learning than administrative tasks.

**Provide Deeper Insights to Parents**  
Parents find it quite hard to get all the info about their kids. [Parent-Teacher meetings](https://leadschool.in/parent-teacher-conference-amid-school-closures-its-possible) are the only times when they can talk to the teacher and ask for details.

But with a student management system in play, it all becomes easy. Parents can access their kid’s data on their phones whenever they like. This way, parents can always stay updated about their kid’s activities.

**Easy Fee Management**  
No one likes to call the parents to ask for a fee, right? Well, with a student management system, you won’t have to.

The parents can access their ward’s fee details on their mobiles. You can even enable notifications for parents regarding fee payments.

This way, the [parents won’t ever miss fee payments](https://leadschool.in/is-collecting-student-fees-an-issue-heres-what-schools-can-do). Also, you won’t have to direct teachers to remind the parents of fee payment.

**Save Cost**  
[Cost-cutting](https://leadschool.in/how-school-erp-software-maximises-productivity-and-minimises-cost) is quite crucial for schools but is quite hard. One cannot decide on what’s required and what’s not.

However, you can cut some costs with a student management system. We’re talking about the “paper costs.”

If you have a software managing all the records, you won’t have to purchase/maintain huge piles of paper based records. This way you can save some money.

Although it may not be a huge saving, “a penny saved is a penny earned.”

## *****What are the disadvantages of the student management system?*****

There aren’t any disadvantages or limitations of a student management system as such. A better word for disadvantages would be [“issues” faced by schools while using a student management system](https://leadschool.in/school-admission-management-system-13-faqs-with-answers-lead).

Fortunately, there aren’t many. The most common issue schools/teachers/students/parents face while using a student management system is: a complex user interface. And that’s a solvable issue.

We’re humans and we need some time to get a hang of things. If the tool you’re using is genuinely tough to use, you can go for a tool that’s simple or get a training.

And that’s it. There are benefits and benefits only of a student management system. You can remove the discrepancies, if any, by researching and finding an ideal student management system.

**8.CONCLUSION**

This brings us to an end to the student’s performance prediction. Let us review our work. First, we started by defining our problem statement, looking into the algorithms we were going to use and the regression implementation pipeline. Then we moved on to practically implementing the identification and regression algorithms like Linear Regression, Lasso, K-Neighbors Regressor, Decision Tree, Random Forest Regressor, XGBRegressor, CatBoosting Regressor, and AdaBoost Regressor. Moving forward, we compared the performances of these models. Lastly, we built a Linear regression model that proved that it works best for student performance prediction problems.

The key takeaways from this  student performance prediction are:

* Identification of student performance prediction is important for many institutions.
* Linear regression gives better accuracy compared to other regression problems.
* Linear regression is the best fit for the problem

**9.FUTURE SCOPE**

**This application can be further developed to show more student performance of a country or a certain region where it’s important for student to have education.**

**10. APPENDIX**

This brings us to an end to the student’s performance prediction. Let us review our work. First, we started by defining our problem statement, looking into the algorithms we were going to use and the regression implementation pipeline. Then we moved on to practically implementing the identification and regression algorithms like Linear Regression, Lasso, K-Neighbors Regressor, Decision Tree, Random Forest Regressor, XGBRegressor, CatBoosting Regressor, and AdaBoost Regressor. Moving forward, we compared the performances of these models. Lastly, we built a Linear regression model that proved that it works best for student performance prediction problems

SOURCE CODE:

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** warnings

warnings.**filterwarnings**("ignore")

df = pd.read\_csv("data/StudentsPerformance.csv")

.

df.shape

df.isnull().sum()

#check the null and Dtypes

df.info()

## Data types analysis | student performance

## Unique values analysis | student performance

print("Categories in 'gender' variable: ",end=" ")

print(df["gender"].unique())

print("Categories in 'race/ethnicity' variable: ",end=" ")

print(df["race/ethnicity"].unique())

print("Categories in 'parental level of education' variable: ",end=" ")

print(df["parental level of education"].unique())

print("Categories in 'lunch' variable: ",end=" ")

print(df["lunch"].unique())

print("Categories in 'test preparation course' variable: ",end=" ")

print(df["test preparation course"].unique())

#define numerical and categorical columns

numeric\_features = [feature **for** feature **in** df.columns **if** df[feature].dtype != "object"]

categorical\_features = [feature **for** feature **in** df.columns **if** df[feature].dtype == "object"]

print("We have {} numerical features: {}".format(len(numeric\_features),numeric\_features))

print("We have {} categorical features: {}".format(len(categorical\_features),categorical\_features))

# Create a figure with two subplots

f,ax=plt.subplots(1,2,figsize=(8,6))

# Create a countplot of the 'gender' column and add labels to the bars

sns.countplot(x=df['gender'],data=df,palette ='bright',ax=ax[0],saturation=0.95)

**for** container **in** ax[0].containers:

ax[0].bar\_label(container,color='black',size=15)

# Set font size of x-axis and y-axis labels and tick labels

ax[0].set\_xlabel('Gender', fontsize=14)

ax[0].set\_ylabel('Count', fontsize=14)

ax[0].tick\_params(labelsize=14)

# Create a pie chart of the 'gender' column and add labels to the slices

plt.pie(x=df['gender'].value\_counts(),labels=['Male','Female'],explode=[0,0.1],autopct='%1.1f%%',shadow=True,colors=['#ff4d4d','#ff8000'], textprops={'fontsize': 14})

# Display the plot

plt.show()

# Define a color palette for the countplot

colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']

# blue, orange, green, red, purple are respectiively the color names for the color codes used above

# Create a figure with two subplots

f, ax = plt.subplots(1, 2, figsize=(12, 6))

# Create a countplot of the 'race/ethnicity' column and add labels to the bars

sns.countplot(x=df['race/ethnicity'], data=df, palette=colors, ax=ax[0], saturation=0.95)

**for** container **in** ax[0].containers:

ax[0].bar\_label(container, color='black', size=14)

# Set font size of x-axis and y-axis labels and tick labels

ax[0].set\_xlabel('Race/Ethnicity', fontsize=14)

ax[0].set\_ylabel('Count', fontsize=14)

ax[0].tick\_params(labelsize=14)

# Create a dictionary that maps category names to colors in the color palette

color\_dict = dict(zip(df['race/ethnicity'].unique(), colors))

# Map the colors to the pie chart slices

pie\_colors = [color\_dict[race] **for** race **in** df['race/ethnicity'].value\_counts().index]

# Create a pie chart of the 'race/ethnicity' column and add labels to the slices

plt.pie(x=df['race/ethnicity'].value\_counts(), labels=df['race/ethnicity'].value\_counts().index, explode=[0.1, 0, 0, 0, 0], autopct='%1.1f%%', shadow=True, colors=pie\_colors, textprops={'fontsize': 14})

# Set the aspect ratio of the pie chart to 'equal' to make it a circle

plt.axis('equal')

# Display the plot

plt.show()

**id = Insights>Insights**

plt.rcParams['figure.figsize'] = (15, 9)

plt.style.use('fivethirtyeight')

sns.histplot(df["parental level of education"], palette = 'Blues')

plt.title('Comparison of Parental Education', fontweight = 30, fontsize = 20)

plt.xlabel('Degree')

plt.ylabel('count')

plt.show()

**id = Insights>Insights**

df.groupby('parental level of education').agg('mean').plot(kind='barh',figsize=(10,10))

plt.legend(bbox\_to\_anchor=(1.05, 1), loc=2, borderaxespad=0.)

plt.show()

**id = Insights>Insights**

plt.fig

figsize=(18,8))

plt.subplot(1, 4, 1)

plt.title('MATH SCORES')

sns.violinplot(y='math score',data=df,color='red',linewidth=3) plt.subplot(1, 4, 2) plt.title('READING SCORES') sns.vi

plot(y='reading score',data=df,color='green',linewidth=3) plt.subplot(1, 4, 3) plt.title('WRITING SCORES') sns.violinplot(y='writing score',data=df,color='blue',linewidth=3) plt.show()

**id = Insights>Insights**

# Set figure size

plt.rcParams['figure.figsize'] = (12, 9)

# First row of pie charts

plt.subplot(2, 3, 1)

size = df['gender'].value\_counts()

labels = 'Female', 'Male'

color = ['red','green']

plt.pie(size, colors=color, labels=labels, autopct='%.2f%%')

plt.title('Gender', fontsize=20)

plt.axis('off')

plt.subplot(2, 3, 2)

size = df['race/ethnicity'].value\_counts()

labels = 'Group C', 'Group D', 'Group B', 'Group E', 'Group A'

color = ['red', 'green', 'blue', 'cyan', 'orange']

plt.pie(size, colors=color, labels=labels, autopct='%.2f%%')

plt.title('Race/Ethnicity', fontsize=20)

plt.axis('off')

plt.subplot(2, 3, 3)

size = df['lunch'].value\_counts()

labels = 'Standard', 'Free'

color = ['red', 'green']

plt.pie(size, colors=color, labels=labels, autopct='%.2f%%')

plt.title('Lunch', fontsize=20)

plt.axis('off')

# Second row of pie charts

plt.subplot(2, 3, 4)

size = df['test preparation course'].value\_counts()

labels = 'None', 'Completed'

color = ['red', 'green']

plt.pie(size, colors=color, labels=labels, autopct='%.2f%%')

plt.title('Test Course', fontsize=20)

plt.axis('off')

plt.subplot(2, 3, 5)

size = df['parental level of education'].value\_counts()

labels = 'Some College', "Associate's Degree", 'High School', 'Some High School', "Bachelor's Degree", "Master's Degree"

color = ['red', 'green', 'blue', 'cyan', 'orange', 'grey']

plt.pie(size, colors=color, labels=labels, autopct='%.2f%%')

plt.title('Parental Education', fontsize=20)

plt.axi

ff') # Remove extra subplot plt.subplot(2, 3, 6).remove() # Add super title plt.suptitle('Comparison of Student Attributes', fontsize=20, fontweight='bold') # Adjust layout and show plot # This is removed as there are only 5 subplots in this figure and we want to arrange them in a 2x3 grid. # Since there is no 6th subplot, it is removed to avoid an empty subplot being shown in the figure. plt.tight\_layout() plt.subplots\_adjust(top=0.85) plt.show()

# Modelling

**from** sklearn.metrics **import** mean\_squared\_error, r2\_score

**from** sklearn.neighbors **import** KNeighborsRegressor

**from** sklearn.tree **import** DecisionTreeRegressor

**from** sklearn.ensemble **import** RandomForestRegressor,AdaBoostRegressor

**from** sklearn.svm **import** SVR

**from** sklearn.linear\_model **import** LinearRegression,Lasso

**from** sklearn.metrics **import** r2\_score, mean\_absolute\_error, mean\_squared\_error

**from** sklearn.model\_selection imp

**RandomizedSearchCV** **from** catboost **import** **CatBoostRegressor** **from** xgboost **import** **XGBRegressor** **import** warnings

X = df.drop(columns="math score",axis=1)

y = df["math score"]

## Create Column Transformer with 3 Types  of Transformers

num\_features = X.select\_dtypes(exclude="object").columns

cat\_features = X.select\_dtypes(**include**="object").columns

from sklearn.preprocessing import OneHotEncoder,StandardScaler

from sklearn.compose import ColumnTransformer

numeric\_transformer = StandardScaler()

oh\_transformer = OneHotEncoder()

preprocessor = Column

transformer( [ ("OneHotEncoder", oh\_transformer, cat\_features), ("StandardScaler", numeric\_transformer, num\_features), ] ) X = preprocessor.fit\_transform(X)

## Separate Dataset into Train and Test

To separate the dataset into train and test to identify the training size and testing size of the dataset.

**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=42)

X\_train.shape, X\_test.shape

https://cdn.analyticsvidhya.com/wp-content/uploads/2023/04/X-train-shape-X-test-shape-300x38.png

## Create an Evaluate Function for Model Training

This function is use to evaluate the model and build a good model.

def **evaluate\_model**(true, predicted):

mae = mean\_absolute\_error(true, predicted)

mse = mean\_squared\_error(true, predicted)

rmse = np.sqrt(mean\_squared\_error(true, predicted))

r2\_square = r2\_score(true, predicted)

**return** mae, mse, rmse, r2\_square

To create a models variable and form a dictionary formate.

models = {

"Linear Regression": LinearRegression(),

"Lasso": Lasso(),

"K-Neighbors Regressor": KNeighborsRegressor(),

"Decision Tree": DecisionTreeRegressor(),

"Random Forest Regressor": RandomForestRegressor(),

"Gradient Boosting": GradientBoostingRegressor(),

"XGBRegressor": XGBRegressor(),

"CatBoosting Regressor": CatBoostRegressor(verbose=False),

"AdaBoost Regressor": AdaBoostRegressor()

}

model\_list = []

r2\_list =[]

**for** i **in** range(len(list(models))):

model = list(models.values())[i]

model.fit(X\_train, y\_train) # Train model

# Make predictions

y\_train\_pred = model.predict(X\_train)

y\_test\_pred = model.predict(X\_test)

# Evaluate Train and Test dataset

model\_train\_mae, model\_train\_mse, model\_train\_rmse, model\_train\_r2 = evaluate\_model(y\_train, y\_train\_pred)

model\_test\_mae, model\_test\_mse, model\_test\_rmse, model\_test\_r2 = evaluate\_model(y\_test, y\_test\_pred)

print(list(models.keys())[i])

model\_list.append(list(models.keys())[i])

print('Model performance for Training set')

print("- Root Mean Squared Error: {:.4f}".format(model\_train\_rmse))

print("- Mean Squared Error: {:.4f}".format(model\_train\_mse))

print("- Mean Absolute Error: {:.4f}".format(model\_train\_mae))

print("- R2 Score: {:.4f}".format(model\_train\_r2))

print('----------------------------------')

print('Model performance for Test set')

print("- Root Mean Squared Error: {:.4f}".format(model\_test\_rmse))

print("- Mean Squared Error: {:.4f}".format(model\_test\_rmse))

print("- Mean Absolute Error: {:.4f}".format(model\_test\_mae))

print("- R2 Score: {:.4f}".

**for**

model\_test\_r2)) r2\_list.append(model\_test\_r2) print('='\*35) print('\n')

GITHUB LINK:

<https://github.com/naanmudhalvan-SI/PBL-NT-GP--7266-1681056033>

PROJECT VIDEO DEMO LINK:

https://drive.google.com/file/d/1VnxlnPWL

3TD9tZe

lbcEjWaFPBR3c6TD/view?usp=drivesdk