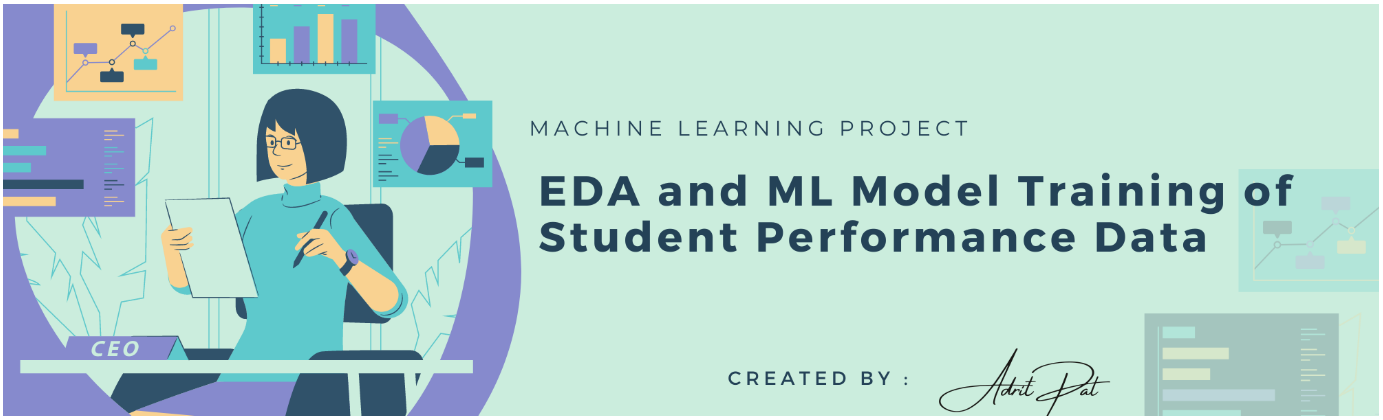
In [189…

```
from IPython.display import Image
Image("/kaggle/input/github-repo1/Cover Image.png")
```

Out[189]:



1. Problem Statement

- The problem statement is how the student's performance(test score) is affected by other variables such as gender, race/ethinicity, parental level of education, lunch and test preparation course.

2. Data Collection and Import Required Packages

In [146…

```
# Basic Import
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import os

# 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, Ridge,Lasso
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.model_selection import RandomizedSearchCV
from catboost import CatBoostRegressor
from xgboost import XGBRegressor

import warnings
warnings.filterwarnings('ignore')
```

In [147…

```
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        student_performance_data=os.path.join(dirname, filename)
#         print(os.path.join(dirname, filename))
df = pd.read_csv(student_performance_data)
print("Data Shape is :",df.shape)
print("\nShow Top 10 Records")
df.head(10)
```

```
Data Shape is : (1000, 8)

Show Top 10 Records
```

Out[147]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
5	female	group B	associate's degree	standard	none	71	83	78
6	female	group B	some college	standard	completed	88	95	92
7	male	group B	some college	free/reduced	none	40	43	39
8	male	group D	high school	free/reduced	completed	64	64	67
9	female	group B	high school	free/reduced	none	38	60	50

3. Dataset Checking to perform

- Check Missing Value
- Check Duplicate
- Check Datatype
- Check the number of unique values of each column
- check statistics of data set
- check various categories present in the different categorical column

3.1 Checking Missing Values

In [148…

```
df.isna().sum()
```

Out[148]:

```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

Result: There are no missing values in the data set.

3.2 Checking Duplicates

In [149…

```
df.duplicated().sum()
```

Out[149]: 0

Result: There are no duplicates values in the data set

3.3 Checking Data Types

In [150…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64 
 6   reading score                1000 non-null   int64 
 7   writing score                1000 non-null   int64 
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

3.4 Checking the number of unique values of each column

In [151…

```
df.nunique()
```

Out[151]:

```
gender                          2
race/ethnicity                  5
parental level of education     6
lunch                           2
test preparation course         2
math score                     81
reading score                  72
writing score                  77
dtype: int64
```

3.5 Print numerical and categorical columns

In [152…

```
# Define numerical & categorical columns
numeric_columns = [column for column in df.columns if df[column].dtype != 'O']
categorical_columns = [column for column in df.columns if df[column].dtype == 'O']

# print columns
print('We have {} numerical columns(features) : {}'.format(len(numeric_columns), numeric
print('\nWe have {} categorical columns(features) : {}'.format(len(categorical_columns),
```

```
We have 3 numerical columns(features) : ['math score', 'reading score', 'writing score']

We have 5 categorical columns(features) : ['gender', 'race/ethnicity', 'parental level o
f education', 'lunch', 'test preparation course']
```

3.6 Print the number of unique values of each categorical column

In [153…

```
# print("Categories in 'gender' variable:     ",end=" " )
# print(df['gender'].unique())
for feature in df.columns :
    if df[feature].dtype == 'O':
        print('Categories in {} variable : {}'.format(feature,df[feature].unique()))
```

```
Categories in gender variable : ['female' 'male']
Categories in race/ethnicity variable : ['group B' 'group C' 'group A' 'group D' 'group
E']
Categories in parental level of education variable : ["bachelor's degree" 'some college'
"master's degree" "associate's degree"
 'high school' 'some high school']
Categories in lunch variable : ['standard' 'free/reduced']
Categories in test preparation course variable : ['none' 'completed']
```

3.7 Checking statistics of data set

In [154…
```
df.describe()
```

Out[154]:

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

Insight

- From above description all means are very close to each other: Between 66 and 69.16
- All standard deviations are also close to each other: Between 14.60 and 15.19
- Minimum score for math is 0, Minimum score for reading is 17, Minimum score for writing is 10

3.8 Adding 'Total' and 'Average' Columns

In [155…
```
df['total score']= df['math score'] + df['reading score'] + df['writing score']
df['avg score'] = df['total score']/3
df.head()
```

Out[155]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	total score	avg score
0	female	group B	bachelor's degree	standard	none	72	72	74	218	72.666667
1	female	group C	some college	standard	completed	69	90	88	247	82.333333
2	female	group B	master's degree	standard	none	90	95	93	278	92.666667
3	male	group A	associate's degree	free/reduced	none	47	57	44	148	49.333333
4	male	group C	some college	standard	none	76	78	75	229	76.333333

3.9 Counting the total number of students who obtained full marks and those who scored less than 25 marks in Mathematics, Reading, and

Writing.

In [156…
```
math_full_score = df[df['math score']==100]['math score'].count()
reading_full_score = df[df['reading score']==100]['reading score'].count()
writing_full_score =  df[df['writing score']==100]['writing score'].count()

print(f'Number of students with full marks in Maths: {math_full_score }')
print(f'Number of students with full marks in Reading: {reading_full_score}')
print(f'Number of students with full marks in Writing: {writing_full_score}')
```

```
Number of students with full marks in Maths: 7
Number of students with full marks in Reading: 17
Number of students with full marks in Writing: 14
```

In [157…
```
math_less_25 = df[df['math score'] <= 25]['math score'].count()
reading_less_25 = df[df['reading score'] <= 25]['reading score'].count()
writing_less_25 = df[df['writing score'] <= 25]['writing score'].count()


print(f'Number of students with less than 25 marks in Maths: {math_less_25}')
print(f'Number of students with less than 25 marks in Reading: {reading_less_25}')
print(f'Number of students with less than 25 marks in Writing: {writing_less_25}')
```

```
Number of students with less than 25 marks in Maths: 7
Number of students with less than 25 marks in Reading: 4
Number of students with less than 25 marks in Writing: 5
```

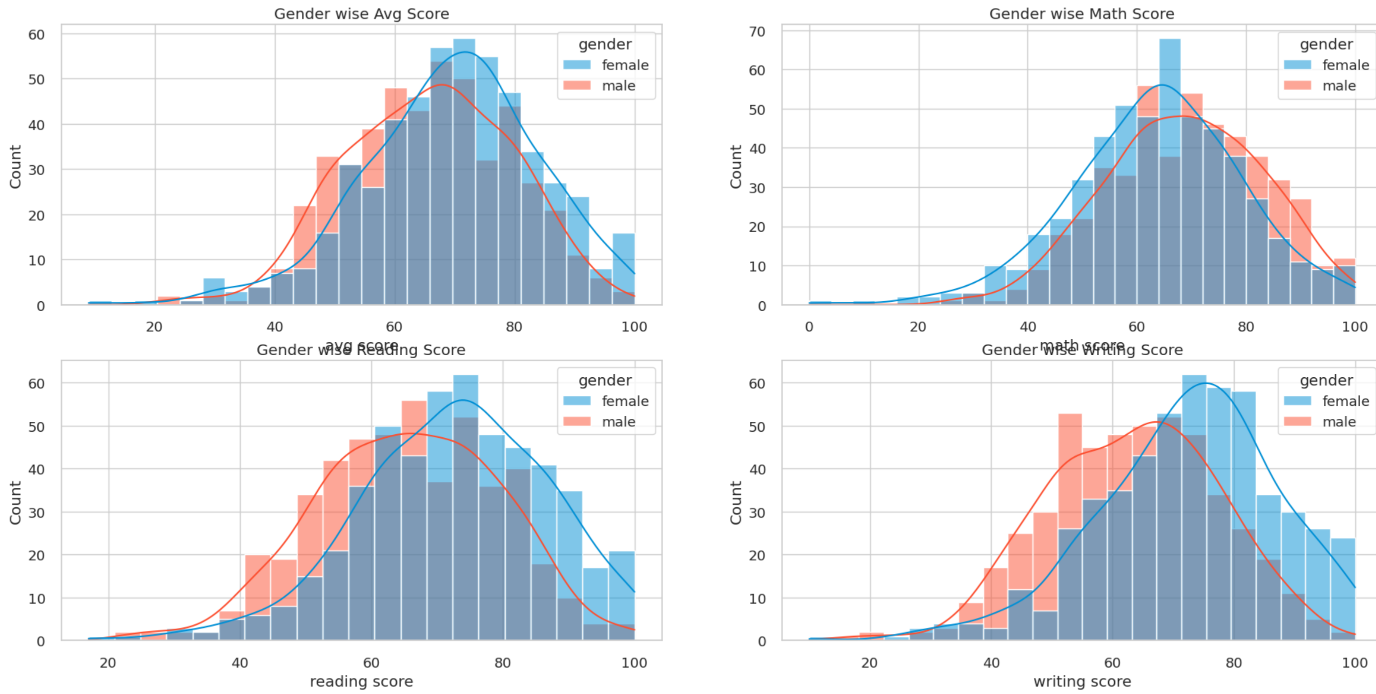
Insight

- From above values we get students have performed the worst in Maths
- Best performance is in reading section

4. Visualizing the Data

4.1 Gender wise Average Score, Math Score, Reading Score, Writing Score distribution

In [158…
```
fig, axs = plt.subplots(2, 2, figsize=(20, 10))

sns.histplot(data=df,x='avg score',kde=True,hue='gender',ax=axs[0, 0])
axs[0, 0].set_title('Gender wise Avg Score')

sns.histplot(data=df,x='math score',kde=True,hue='gender',ax=axs[0, 1])
axs[0, 1].set_title('Gender wise Math Score')

sns.histplot(data=df,x='reading score',kde=True,hue='gender',ax=axs[1, 0])
axs[1, 0].set_title('Gender wise Reading Score')

sns.histplot(data=df,x='writing score',kde=True,hue='gender',ax=axs[1, 1])
axs[1, 1].set_title('Gender wise Writing Score')

plt.show()
```



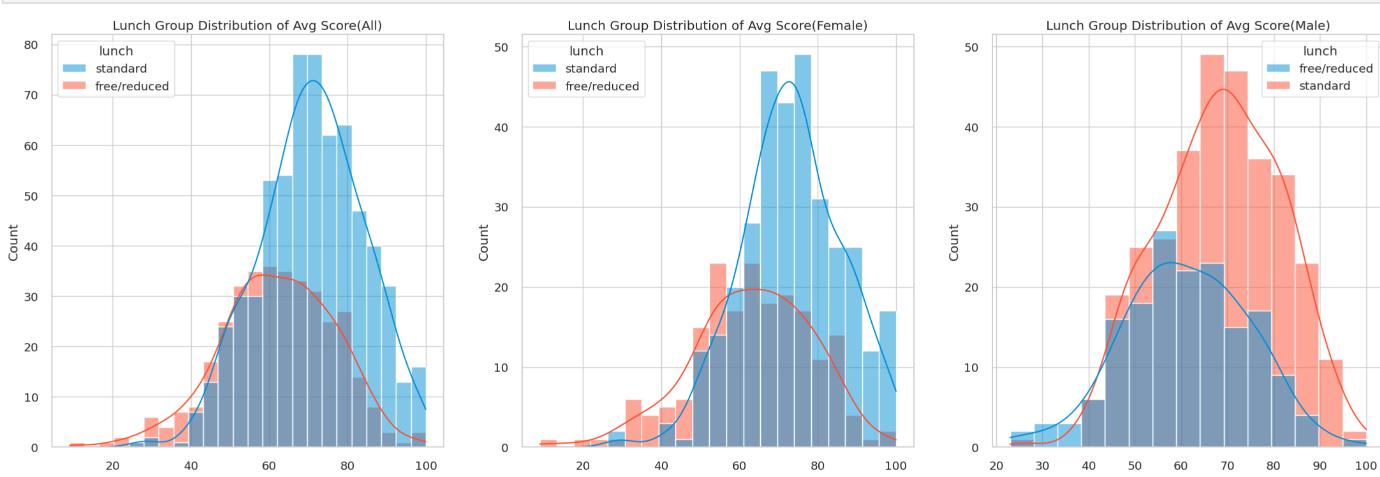
Insights

- Female students tend to perform well then male students

4.2 Lunch Group wise Score Distribution

In [159…

```
fig, axs = plt.subplots(1, 3, figsize=(24,8))

axs[0].set_title('Lunch Group Distribution of Avg Score(All)')
sns.histplot(data=df, x='avg score', kde=True, hue='lunch', ax=axs[0])

axs[1].set_title('Lunch Group Distribution of Avg Score(Female)')
sns.histplot(data=df[df.gender=='female'], x='avg score', kde=True, hue='lunch', ax=axs[

axs[2].set_title('Lunch Group Distribution of Avg Score(Male)')
sns.histplot(data=df[df.gender=='male'], x='avg score', kde=True, hue='lunch', ax=axs[2]

plt.show()
```



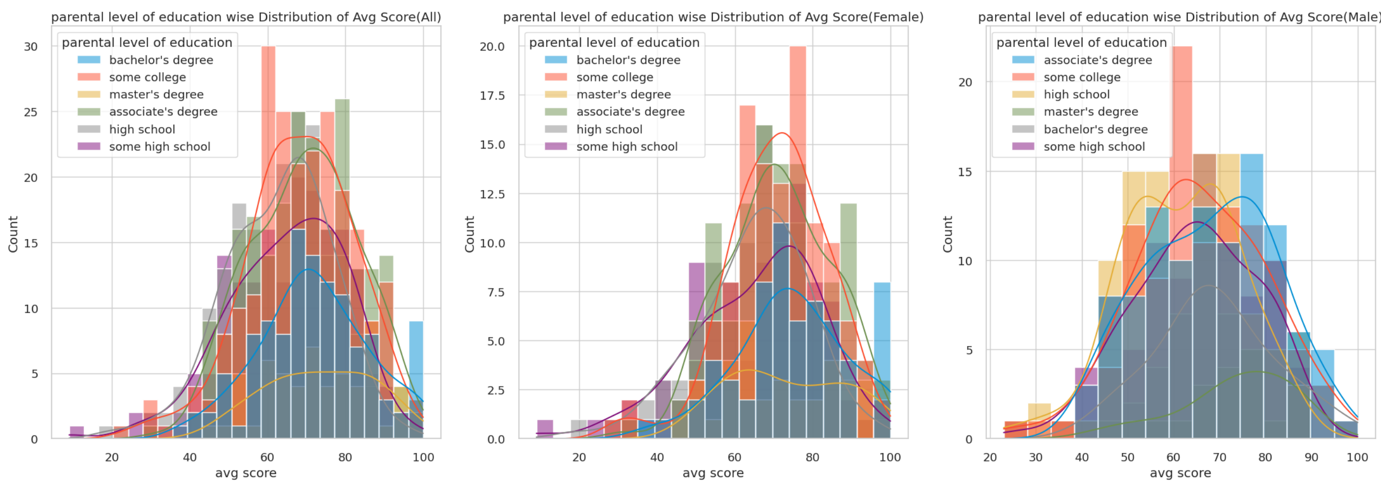
Insights

- Standard lunch helps perform well in exams.
- Standard lunch helps perform well in exams be it a male or a female.

4.3 parental level of education wise Score Distribution

In [160…

```
fig, axs = plt.subplots(1, 3, figsize=(24,8))

axs[0].set_title('parental level of education wise Distribution of Avg Score(All)')
sns.histplot(data=df, x='avg score', kde=True, hue='parental level of education', ax=axs

axs[1].set_title('parental level of education wise Distribution of Avg Score(Female)')
sns.histplot(data=df[df.gender=='female'], x='avg score', kde=True, hue='parental level

axs[2].set_title('parental level of education wise Distribution of Avg Score(Male)')
sns.histplot(data=df[df.gender=='male'], x='avg score', kde=True, hue='parental level of

plt.show()
```



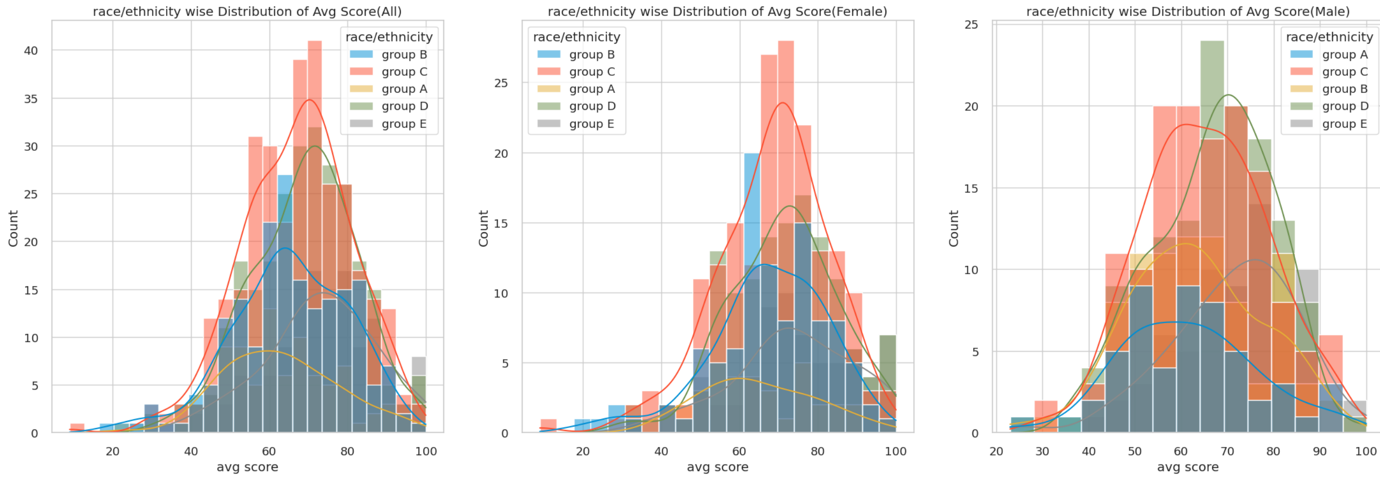
Insights

- In general parent's education don't help student perform well in exam.
- 2nd plot shows that parent's whose education is of associate's degree or master's degree their male child tend to perform well in exam
- 3rd plot we can see there is no effect of parent's education on female students.

4.4 race/ethnicity wise Score Distribution

In [161…

```
fig, axs = plt.subplots(1, 3, figsize=(24,8))

axs[0].set_title('race/ethnicity wise Distribution of Avg Score(All)')
sns.histplot(data=df, x='avg score', kde=True, hue='race/ethnicity', ax=axs[0])

axs[1].set_title('race/ethnicity wise Distribution of Avg Score(Female)')
sns.histplot(data=df[df.gender=='female'], x='avg score', kde=True, hue='race/ethnicity'

axs[2].set_title('race/ethnicity wise Distribution of Avg Score(Male)')
sns.histplot(data=df[df.gender=='male'], x='avg score', kde=True, hue='race/ethnicity',

plt.show()
```



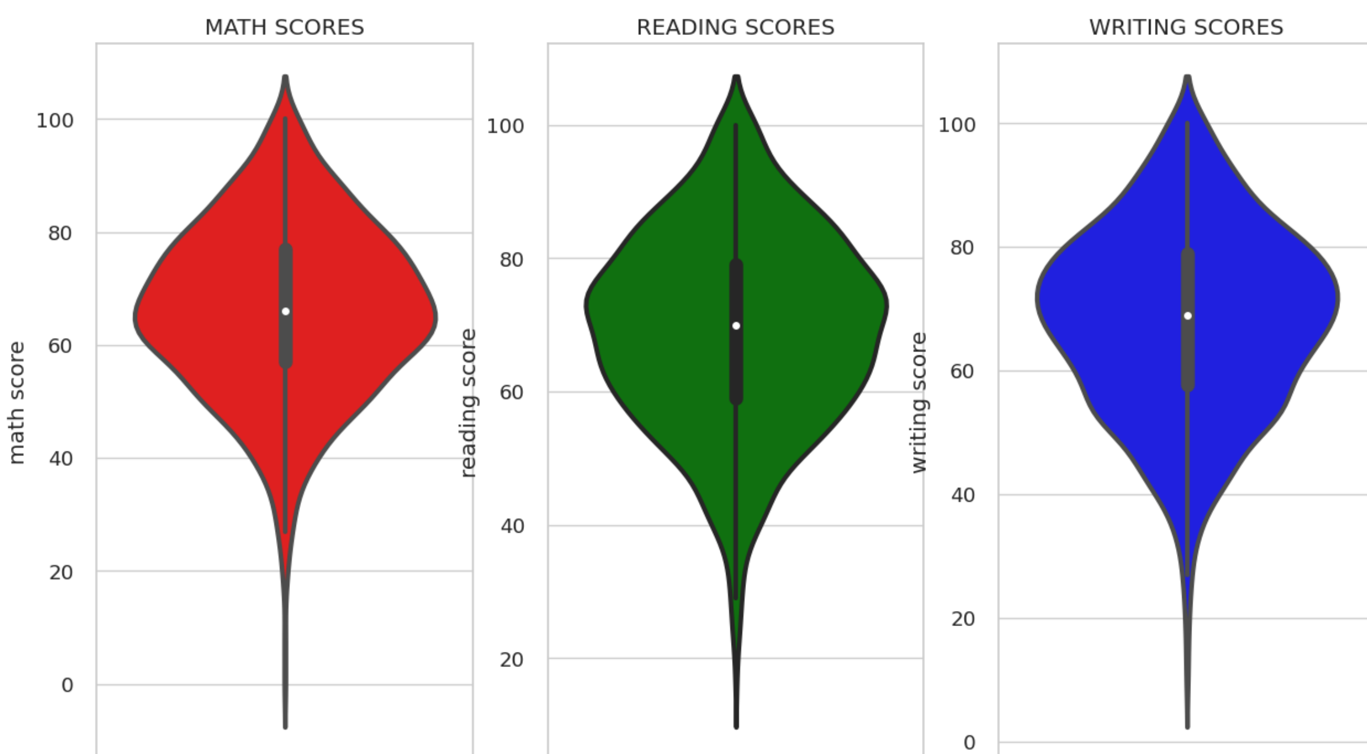
Insights

- Students of group A and group B tends to perform poorly in exam.
- Students of group A and group B tends to perform poorly in exam irrespective of whether they are male or female

4.5 score of students in all three subjects

In [162…

```
plt.figure(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.violinplot(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()
```



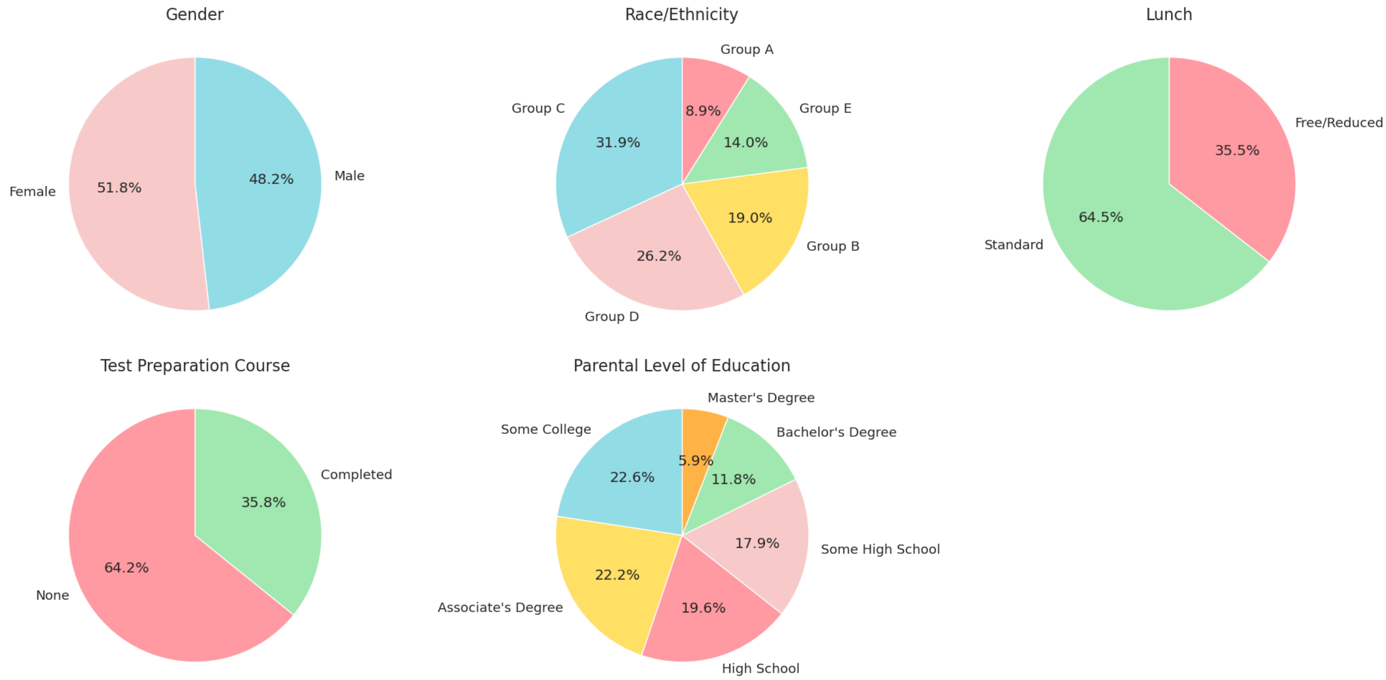
Insights

- 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

4.6 Multivariate analysis using pieplot

In [163…

```
plt.figure(figsize=(20, 10))

# Gender
plt.subplot(2, 3, 1)
size = df['gender'].value_counts()
labels = ['Female', 'Male']
colors = ['#F7CAC9', '#92DCE5']

plt.pie(size, colors=colors, labels=labels, autopct='%.1f%%', startangle=90)
plt.title('Gender',fontsize=16)

# Race/Ethnicity
plt.subplot(2, 3, 2)
size = df['race/ethnicity'].value_counts()
labels = ['Group C', 'Group D', 'Group B', 'Group E', 'Group A']
colors = ['#92DCE5', '#F7CAC9', '#FFDF64', '#A0E8AF', '#FF9AA2']

plt.pie(size, colors=colors, labels=labels, autopct='%.1f%%', startangle=90)
plt.title('Race/Ethnicity',fontsize=16)

# Lunch
plt.subplot(2, 3, 3)
size = df['lunch'].value_counts()
labels = ['Standard', 'Free/Reduced']
colors = ['#A0E8AF', '#FF9AA2']

plt.pie(size, colors=colors, labels=labels, autopct='%.1f%%', startangle=90)
plt.title('Lunch',fontsize=16)

# Test Preparation Course
plt.subplot(2, 3, 4)
size = df['test preparation course'].value_counts()
labels = ['None', 'Completed']
colors = ['#FF9AA2', '#A0E8AF']

plt.pie(size, colors=colors, labels=labels, autopct='%.1f%%', startangle=90)
plt.title('Test Preparation Course',fontsize=16)

# Parental Level of Education
plt.subplot(2, 3, 5)
size = df['parental level of education'].value_counts()
labels = ["Some College", "Associate's Degree", "High School", "Some High School", "Bach
colors = ['#92DCE5', '#FFDF64', '#FF9AA2', '#F7CAC9', '#A0E8AF', '#FFB347']

plt.pie(size, colors=colors, labels=labels, autopct='%.1f%%', startangle=90)
plt.title('Parental Level of Education',fontsize=16)

plt.tight_layout()
plt.show()
```



Insights

- Gender wise Number of Male and Female students is almost equal
- Race/Ethnicity wise Number students are greatest in Group C
- Lunch wise Number of students who have standard lunch are greater
- Test Preparation Course wise Number of students who have not enrolled in any test preparation course is greater
- Parental Level of Education wise Number of students whose parental education is "Some College" is greater followed closely by "Associate's Degree"

4.7 Feature wise Visualization

4.7.1 Gender Column

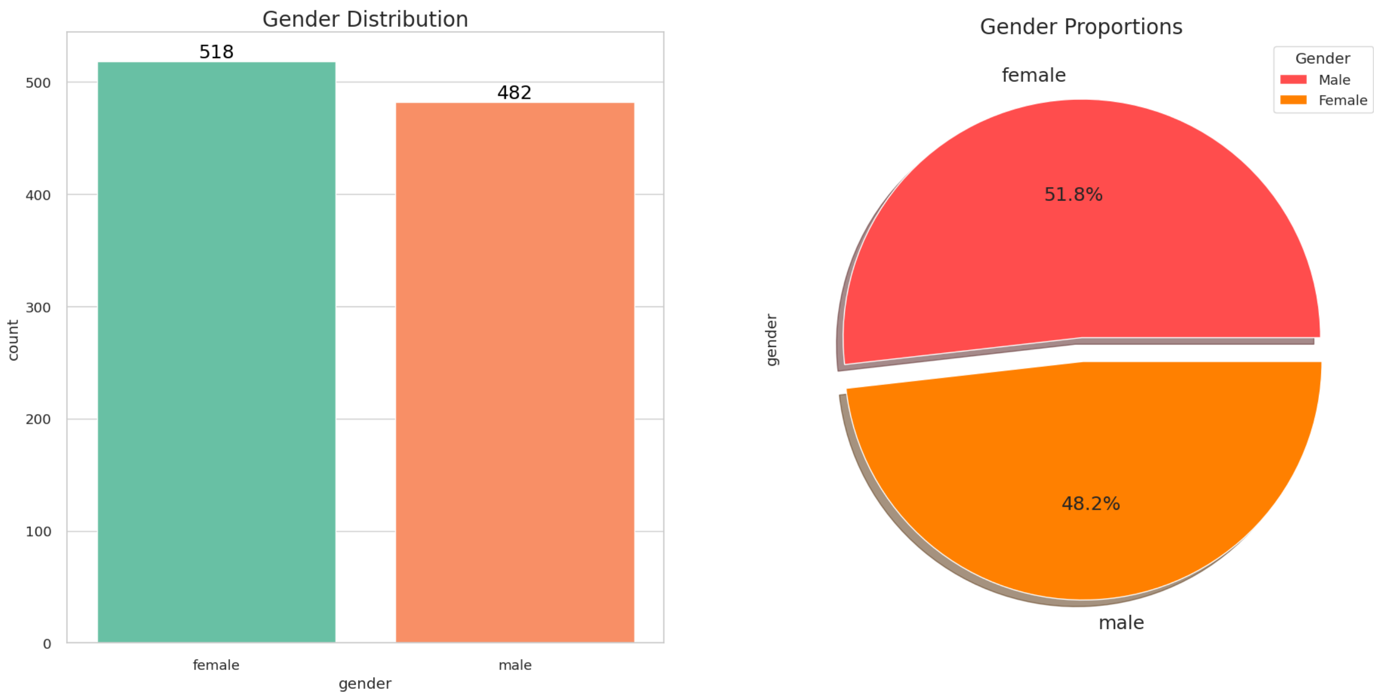
- How is distribution of Gender ?
- Is gender has any impact on student's performance?

How is distribution of Gender? **(Univariate Analysis)**

In [164…

```
# set the color palette
colors = sns.color_palette('Set2')

# create the subplots
fig, ax = plt.subplots(1, 2, figsize=(20, 10))

# plot the count plot
sns.countplot(x=df['gender'], data=df, palette=colors, ax=ax[0], saturation=0.95)

# add labels to the bars
for container in ax[0].containers:
    ax[0].bar_label(container, color='black', fontsize=18)

# plot the pie chart
pie_colors = ['#ff4d4d', '#ff8000']
```

```
df['gender'].value_counts().plot(kind='pie', colors=pie_colors, explode=[0, 0.1],
                                  autopct='%1.1f%%', shadow=True, ax=ax[1], textprops={'f
# set title and legend
ax[0].set_title('Gender Distribution', fontsize=20)
ax[1].set_title('Gender Proportions', fontsize=20)
ax[1].legend(title='Gender', loc='best', labels=['Male', 'Female'])

# show the plot
plt.show()
```



Insights

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

Is gender has any impact on student's performance? **(Bivariate Analysis)**

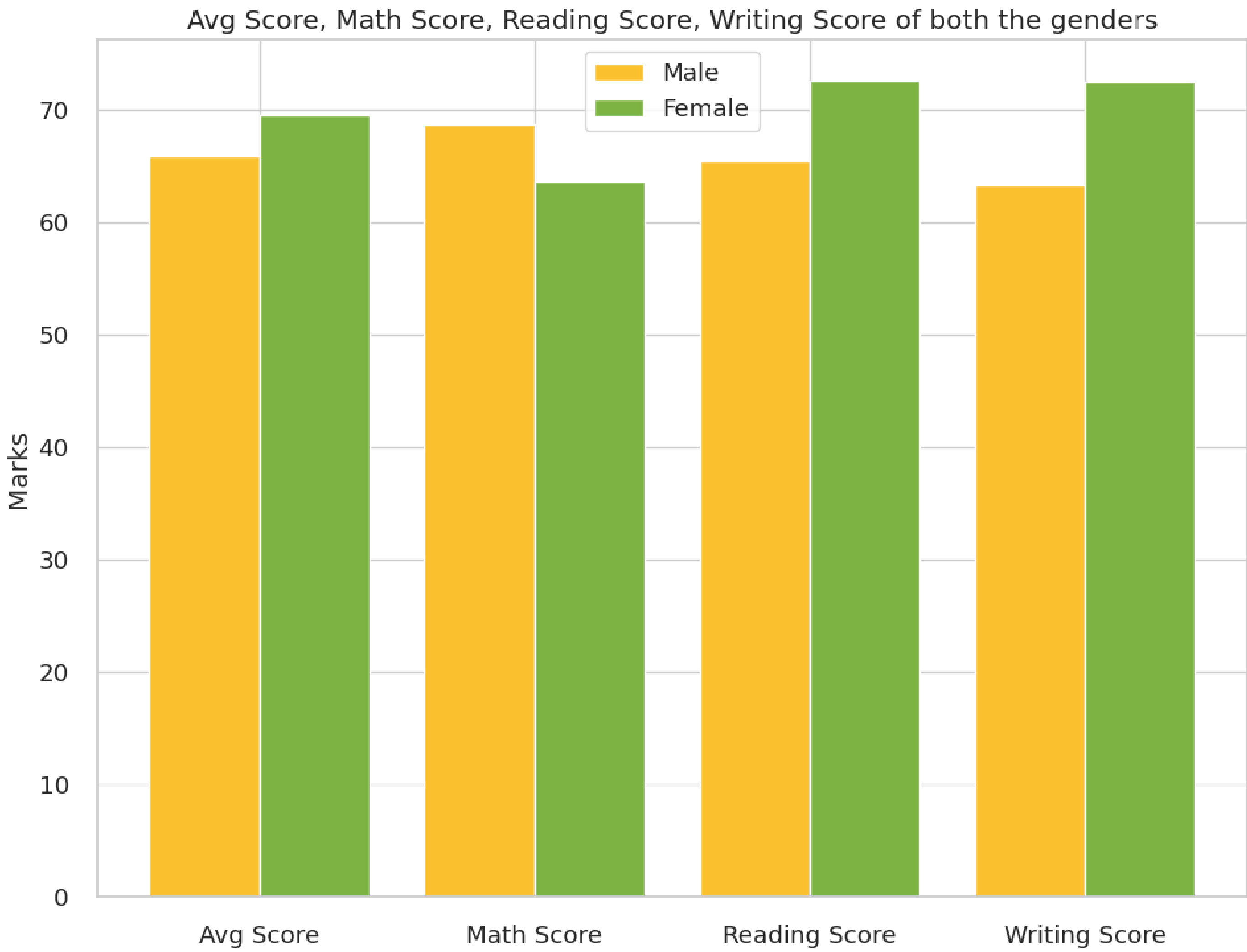
In [165…
```
gender_group = df.groupby('gender').mean()
gender_group
```

Out[165]:

	math score	reading score	writing score	total score	avg score
gender					
female	63.633205	72.608108	72.467181	208.708494	69.569498
male	68.728216	65.473029	63.311203	197.512448	65.837483

In [166…
```
sns.set_palette('Set2')
plt.figure(figsize=(10, 8))

X = ['Avg Score', 'Math Score', 'Reading Score', 'Writing Score']
female_scores = [gender_group['avg score'][0], gender_group['math score'][0], gender_gro
male_scores = [gender_group['avg score'][1], gender_group['math score'][1], gender_group

X_axis = np.arange(len(X))

plt.bar(X_axis - 0.2, male_scores, 0.4, label='Male', color='#FBC02D')
plt.bar(X_axis + 0.2, female_scores, 0.4, label='Female', color='#7CB342')

plt.xticks(X_axis, X)
```

```
plt.ylabel("Marks")
plt.title("Avg Score, Math Score, Reading Score, Writing Score of both the genders")
plt.legend()
plt.show()
```



Insights

- On an average females have a better overall score than men.
- whereas males have scored higher in Maths.

4.7.2 Race/Ethnicity Column

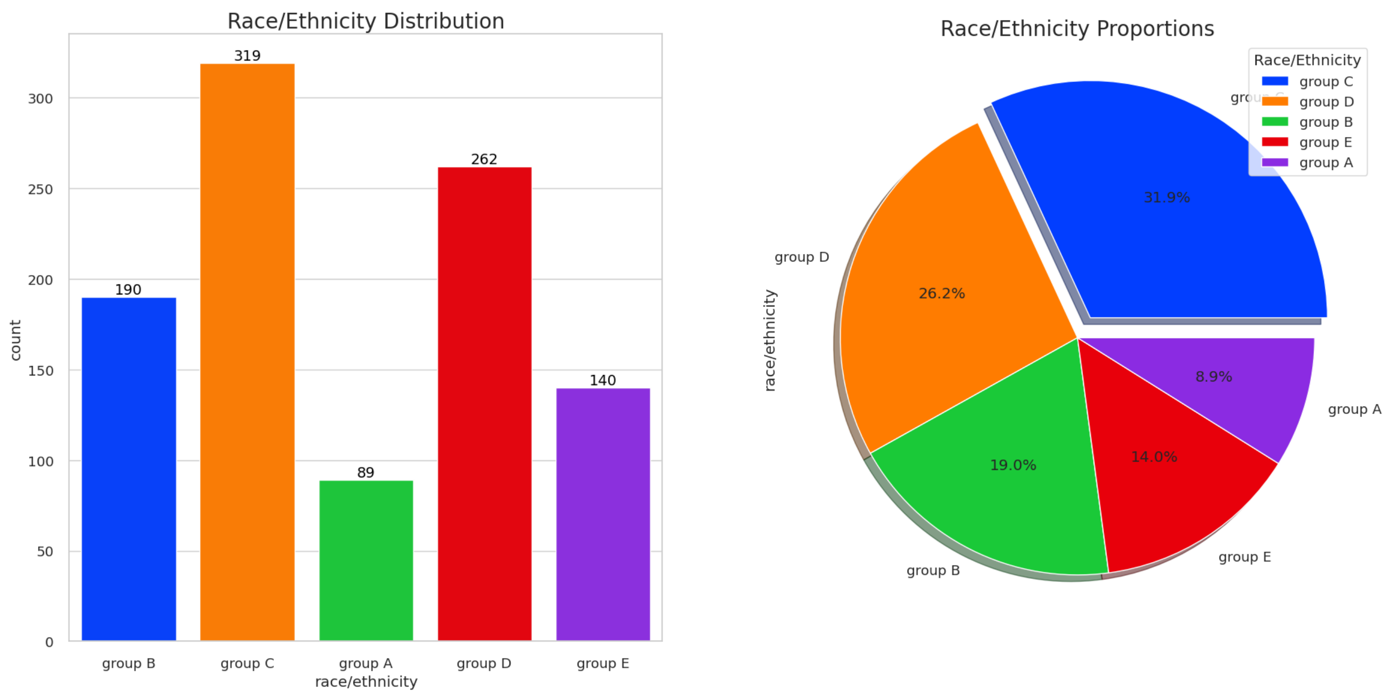
- How is Group wise distribution ?
- Is race/ethnicity has any impact on student's performance?

How is Group wise distribution? **(Univariate Analysis)**

In [167…

```
# set the color palette
colors = sns.color_palette('bright')

# create the subplots
fig, ax = plt.subplots(1, 2, figsize=(20, 10))

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

# add labels to the bars
```

```python
for container in ax[0].containers:
    ax[0].bar_label(container, color='black', fontsize=14)

# plot the pie chart
explode = [0.1, 0, 0, 0, 0]
df['race/ethnicity'].value_counts().plot(kind='pie', colors=colors, explode=explode,
                                          autopct='%1.1f%%', shadow=True, ax=ax[1])
# set title and legend
ax[0].set_title('Race/Ethnicity Distribution', fontsize=20)
ax[1].set_title('Race/Ethnicity Proportions', fontsize=20)
ax[1].legend(title='Race/Ethnicity', loc='best', labels=df['race/ethnicity'].value_count

# show the plot
plt.show()
```



Insights

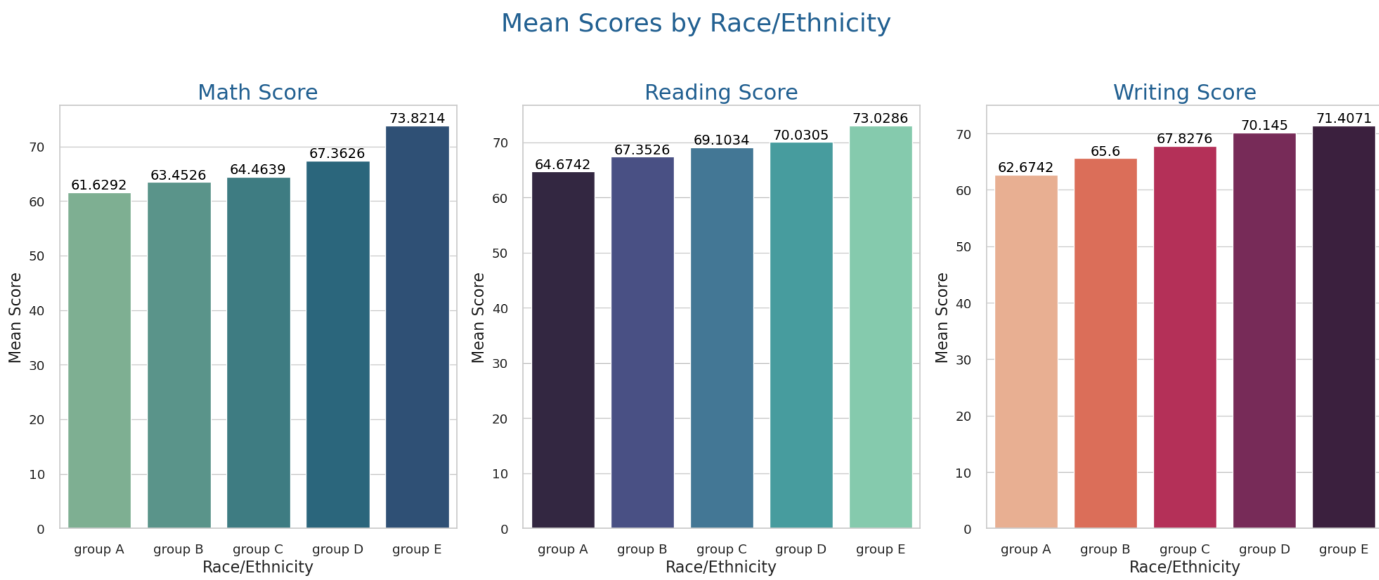
- Most of the student belonging from group C /group D.
- Lowest number of students belong to groupA.

Is race/ethnicity has any impact on student's performance? **(Bivariate Analysis)**

In [168…

```python
Group_data2 = df.groupby('race/ethnicity')

# create the subplots
fig, ax = plt.subplots(1, 3, figsize=(20, 8))

# plot the math scores
sns.barplot(x=Group_data2['math score'].mean().index, y=Group_data2['math score'].mean()
ax[0].set_title('Math Score', color='#236192', size=22)
ax[0].set_xlabel('Race/Ethnicity', fontsize=16)
ax[0].set_ylabel('Mean Score', fontsize=16)

# add labels to the bars
for container in ax[0].containers:
    ax[0].bar_label(container, color='black', fontsize=14)

# plot the reading scores
sns.barplot(x=Group_data2['reading score'].mean().index, y=Group_data2['reading score'].
```

```
ax[1].set_title('Reading Score', color='#236192', size=22)
ax[1].set_xlabel('Race/Ethnicity', fontsize=16)
ax[1].set_ylabel('Mean Score', fontsize=16)

# add labels to the bars
for container in ax[1].containers:
    ax[1].bar_label(container, color='black', fontsize=14)

# plot the writing scores
sns.barplot(x=Group_data2['writing score'].mean().index, y=Group_data2['writing score'].
ax[2].set_title('Writing Score', color='#236192', size=22)
ax[2].set_xlabel('Race/Ethnicity', fontsize=16)
ax[2].set_ylabel('Mean Score', fontsize=16)

# add labels to the bars
for container in ax[2].containers:
    ax[2].bar_label(container, color='black', fontsize=14)

plt.suptitle('Mean Scores by Race/Ethnicity', fontsize=26, color='#236192', y=1.03)
plt.tight_layout()
plt.show()
```



Insights

- Group E students have scored the highest marks.
- Group A students have scored the lowest marks.

4.7.3 Parental Level Of Education Column

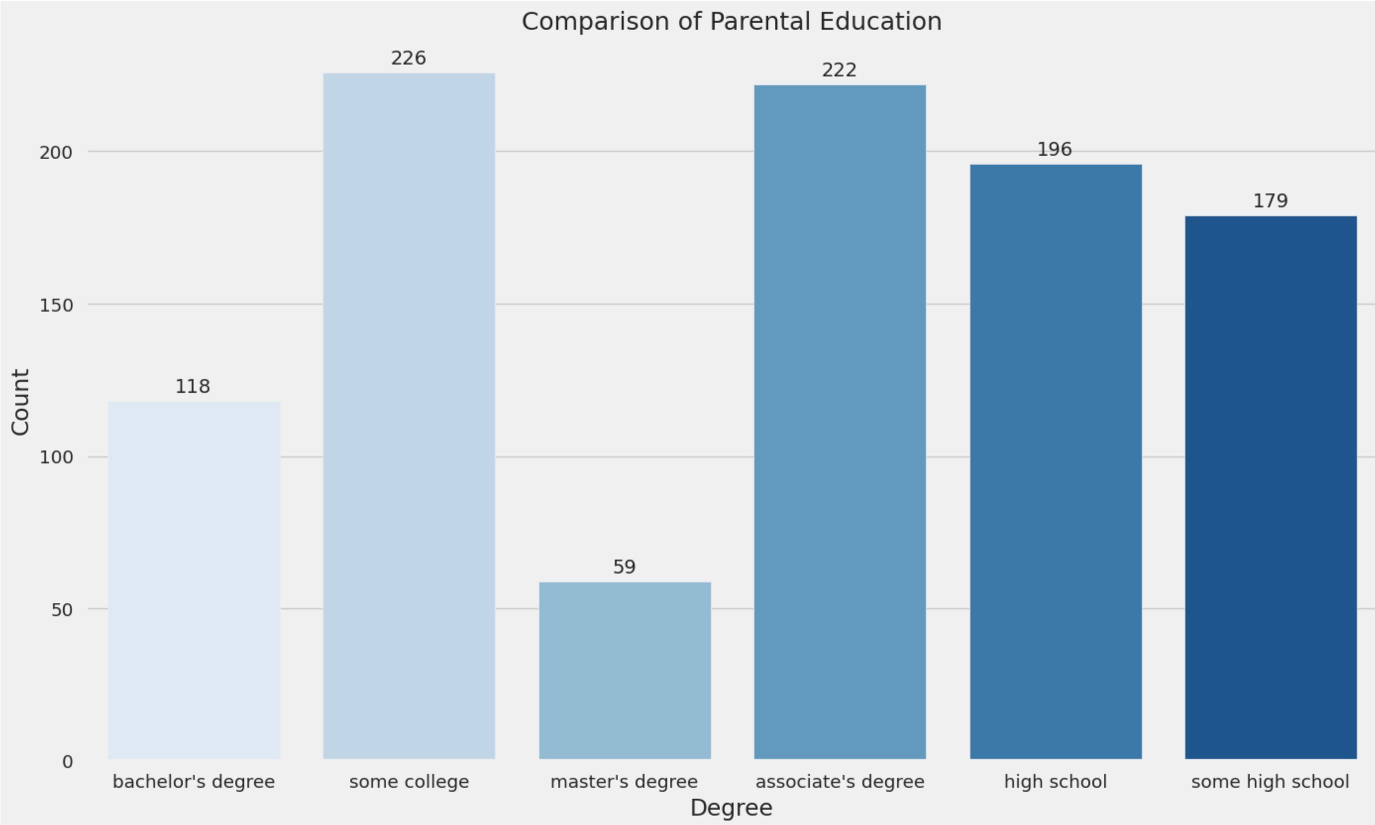
- What is the best educational background of the student's parents?
- Is parental education has any impact on student's performance?

What is the best educational background of the student's parents? **(Univariate Analysis)**

In [169…

```
plt.rcParams['figure.figsize'] = (15, 9)
plt.style.use('fivethirtyeight')
ax = sns.countplot(x='parental level of education', data=df, palette='Blues')
plt.title('Comparison of Parental Education', fontsize=18)
ax.set(xlabel='Degree', ylabel='Count')
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.0f'),
```

```
                 (p.get_x() + p.get_width() / 2., p.get_height()),
                 ha = 'center', va = 'center',
                 xytext = (0, 10),
                 textcoords = 'offset points')
plt.show()
```



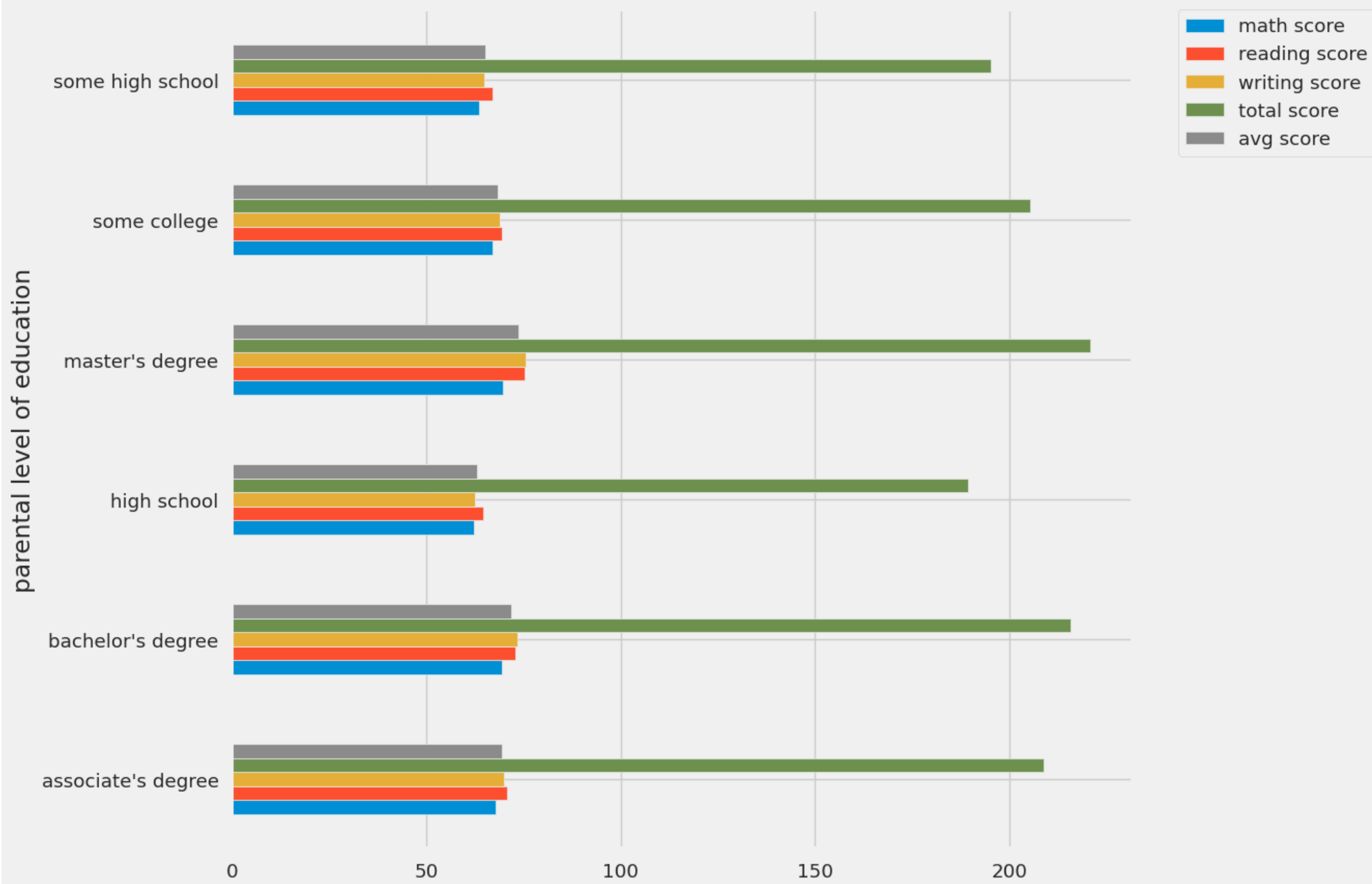
Insights

- Largest number of parents are from some college.

Is parental education has any impact on student's performance? **(Bivariate Analysis)**

In [170…

```
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()
```

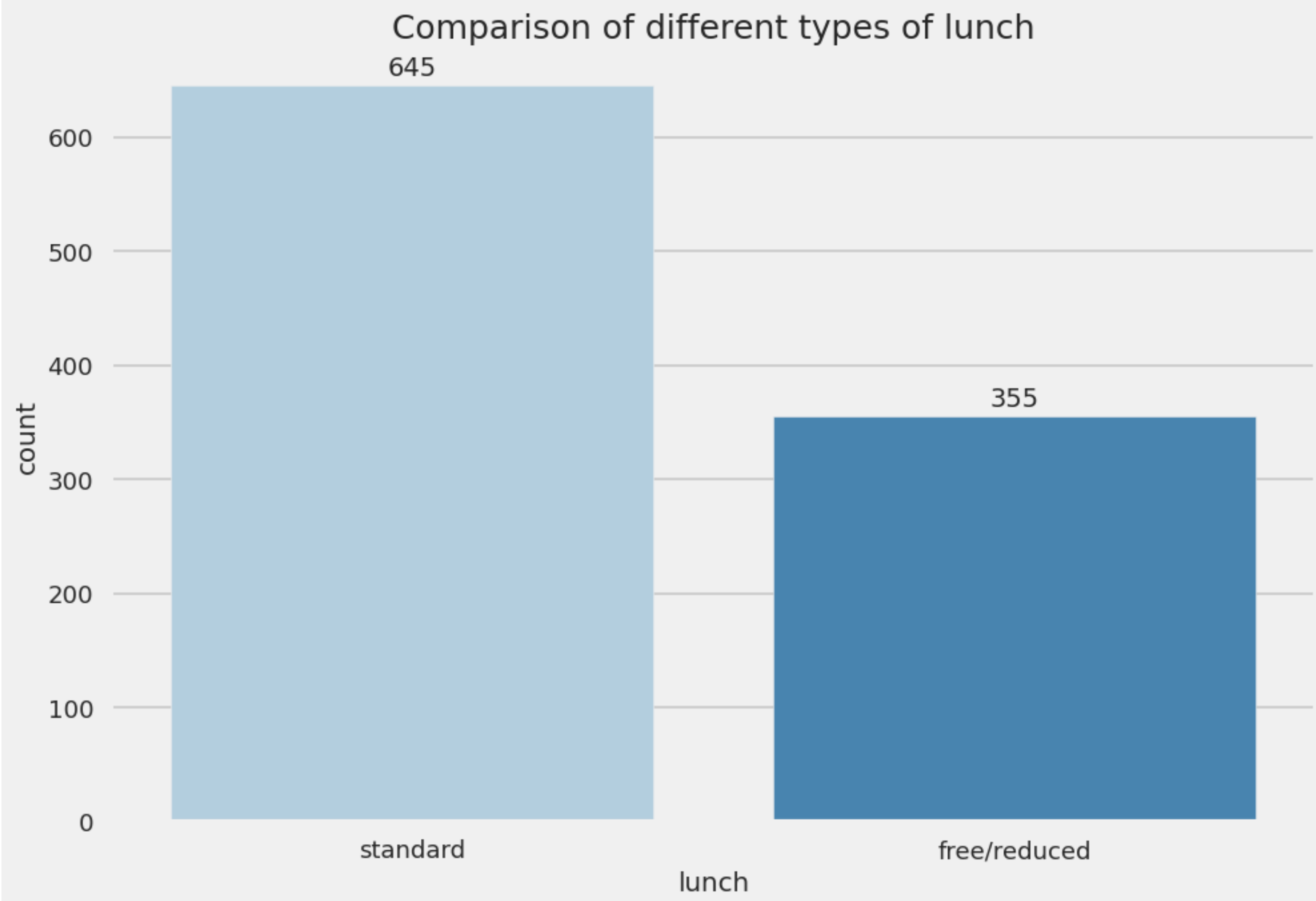


Insights

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

In [171…

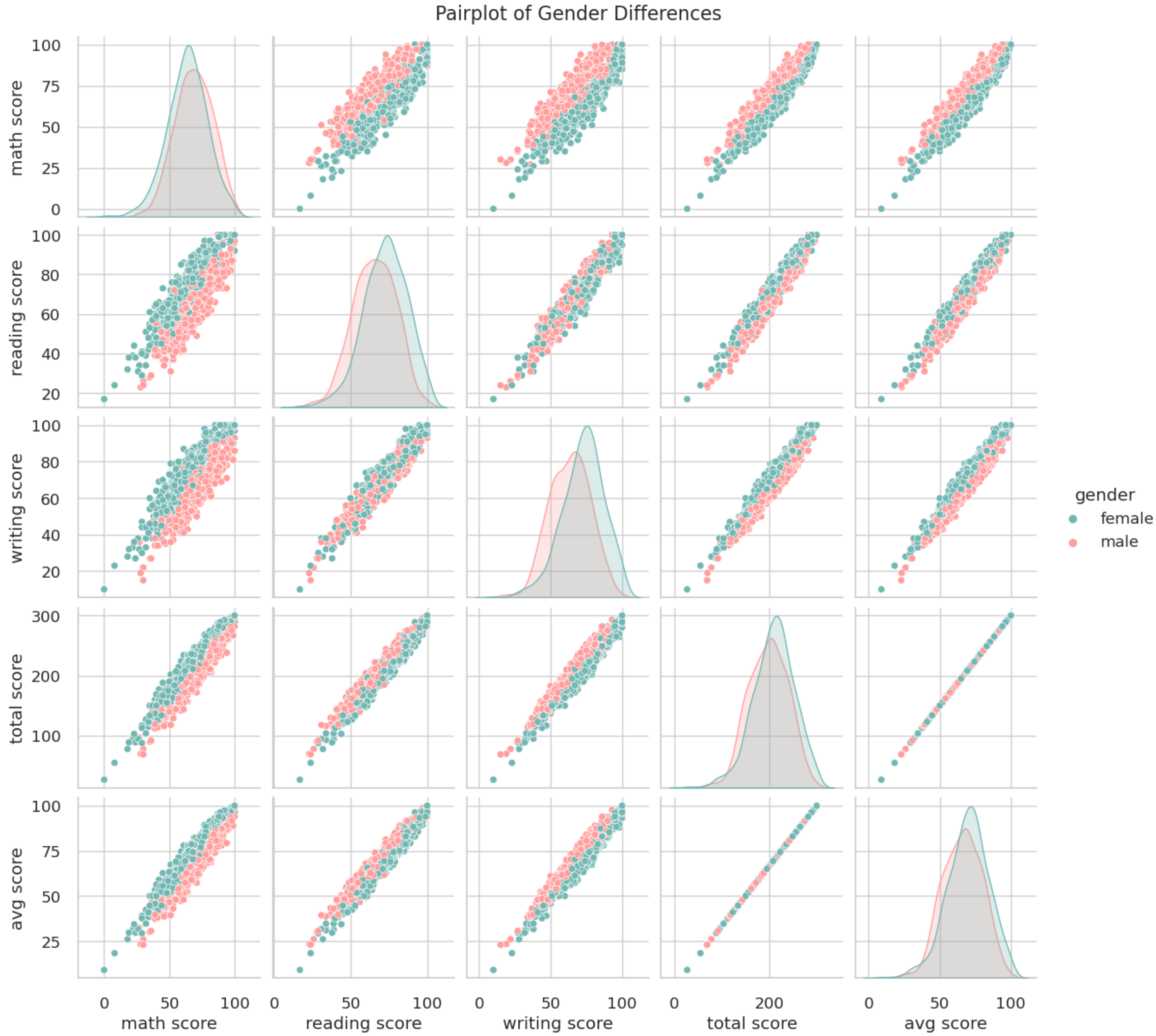
```
plt.rcParams['figure.figsize'] = (15, 9)
plt.style.use('seaborn-talk')
ax = sns.countplot(x='lunch', data=df, palette='Blues')
plt.title('Comparison of different types of lunch', fontsize=18)
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.0f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 10),
                textcoords = 'offset points')
plt.show()
```



4.8 Multivariate Analysis Using Pairplot

In [172…

```
# Use a modern color palette
colors = ["#72b7b2", "#ff9e9d"]

# Set style and context
sns.set_style("whitegrid")
sns.set_context("notebook", font_scale=1.2)

# Create pairplot with hue and custom color palette
sns.pairplot(df, hue="gender", palette=colors)

# Add some visual enhancements
plt.subplots_adjust(top=0.95)
plt.suptitle("Pairplot of Gender Differences", fontsize=16)

# Show the plot
plt.show()
```



Insights

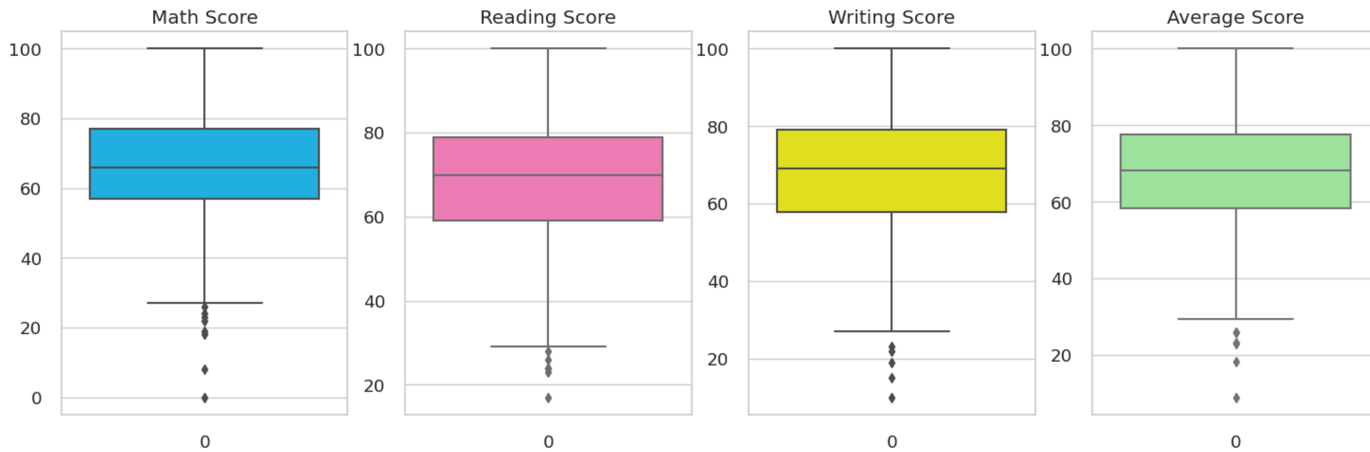
- From the above plot it is clear that all the scores increase linearly with each other.

4.9 Checking Outliers

In [173…

```
# Define modern color palettes
skyblue = "#00BFFF"
hotpink = "#FF69B4"
yellow = "#FFFF00"
lightgreen = "#90EE90"

# Create subplots with specified figure size
fig, axs = plt.subplots(1, 4, figsize=(16, 5))

# Plot boxplots for each score column and set color using the defined palettes
sns.boxplot(df['math score'], color=skyblue, ax=axs[0])
sns.boxplot(df['reading score'], color=hotpink, ax=axs[1])
sns.boxplot(df['writing score'], color=yellow, ax=axs[2])
sns.boxplot(df['avg score'], color=lightgreen, ax=axs[3])

# Set titles for each subplot
axs[0].set_title('Math Score')
```

```
axs[1].set_title('Reading Score')
axs[2].set_title('Writing Score')
axs[3].set_title('Average Score')

# Show the plot
plt.show()
```



5. Model Training

5.1 Show Top 10 Records

In [174…

```
df.head(10)
```

Out[174]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	total score	avg score
0	female	group B	bachelor's degree	standard	none	72	72	74	218	72.666667
1	female	group C	some college	standard	completed	69	90	88	247	82.333333
2	female	group B	master's degree	standard	none	90	95	93	278	92.666667
3	male	group A	associate's degree	free/reduced	none	47	57	44	148	49.333333
4	male	group C	some college	standard	none	76	78	75	229	76.333333
5	female	group B	associate's degree	standard	none	71	83	78	232	77.333333
6	female	group B	some college	standard	completed	88	95	92	275	91.666667
7	male	group B	some college	free/reduced	none	40	43	39	122	40.666667
8	male	group D	high school	free/reduced	completed	64	64	67	195	65.000000
9	female	group B	high school	free/reduced	none	38	60	50	148	49.333333

5.2 Preparing X and Y variables

In [175…

```
X =df.drop(columns=['total score','avg score','math score'],axis=1)
print("Data Shape is :",X.shape)
```

```
X.head()
```

```
Data Shape is : (1000, 7)
```

Out[175]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	74
1	female	group C	some college	standard	completed	90	88
2	female	group B	master's degree	standard	none	95	93
3	male	group A	associate's degree	free/reduced	none	57	44
4	male	group C	some college	standard	none	78	75

In [176…

```
Y = df['math score']
Y.head()
```

Out[176]:

```
0    72
1    69
2    90
3    47
4    76
Name: math score, dtype: int64
```

5.3 Create Column Transformer with 3 types of transformers

In [177…

```
# 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 = ColumnTransformer(
    [
        ("OneHotEncoder", oh_transformer, cat_features),
         ("StandardScaler", numeric_transformer, num_features),
    ]
)
```

In [178…

```
X = preprocessor.fit_transform(X)
```

In [179…

```
X.shape
```

Out[179]: (1000, 19)

5.4 Separate dataset into train and test

In [180…

```
# separate dataset into train and test
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
```

Out[180]: ((800, 19), (200, 19))

5.5 Create an Evaluate Function to give all metrics after model Training

In [181…

```
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, rmse, r2_square
```

5.6 Models Training

In [182…

```
models = {
    "Linear Regression": LinearRegression(),
    "Lasso": Lasso(),
    "Ridge": Ridge(),
    "K-Neighbors Regressor": KNeighborsRegressor(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest Regressor": RandomForestRegressor(),
    "XGBRegressor": XGBRegressor(),
    "CatBoosting Regressor": CatBoostRegressor(verbose=False),
    "AdaBoost Regressor": AdaBoostRegressor()
}
model_list = []
r2_list =[]

# Train model
for i in range(len(list(models))):
    model = list(models.values())[i]
    model.fit(X_train, Y_train)

    # 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_rmse, model_train_r2 = evaluate_model(Y_train, Y_train

    model_test_mae , 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 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 Absolute Error: {:.4f}".format(model_test_mae))
    print("- R2 Score: {:.4f}".format(model_test_r2))
    r2_list.append(model_test_r2)

    print('='*35)
    print('\n')
```

```
Linear Regression
Model performance for Training set
- Root Mean Squared Error: 5.3283
- Mean Absolute Error: 4.2698
- R2 Score: 0.8741
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 5.4227
- Mean Absolute Error: 4.2217
- R2 Score: 0.8792
===================================


Lasso
Model performance for Training set
- Root Mean Squared Error: 6.5938
- Mean Absolute Error: 5.2063
- R2 Score: 0.8071
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 6.5197
- Mean Absolute Error: 5.1579
- R2 Score: 0.8253
===================================


Ridge
Model performance for Training set
- Root Mean Squared Error: 5.3233
- Mean Absolute Error: 4.2650
- R2 Score: 0.8743
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 5.3904
- Mean Absolute Error: 4.2111
- R2 Score: 0.8806
===================================


K-Neighbors Regressor
Model performance for Training set
- Root Mean Squared Error: 5.7055
- Mean Absolute Error: 4.5122
- R2 Score: 0.8556
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 7.2634
- Mean Absolute Error: 5.6590
- R2 Score: 0.7832
===================================


Decision Tree
Model performance for Training set
- Root Mean Squared Error: 0.2795
- Mean Absolute Error: 0.0187
- R2 Score: 0.9997
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 7.7836
- Mean Absolute Error: 6.2950
- R2 Score: 0.7510
```

```
===================================


Random Forest Regressor
Model performance for Training set
- Root Mean Squared Error: 2.3225
- Mean Absolute Error: 1.8436
- R2 Score: 0.9761
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 6.0503
- Mean Absolute Error: 4.7040
- R2 Score: 0.8496
===================================


XGBRegressor
Model performance for Training set
- Root Mean Squared Error: 0.9087
- Mean Absolute Error: 0.6148
- R2 Score: 0.9963
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 6.5889
- Mean Absolute Error: 5.0844
- R2 Score: 0.8216
===================================


CatBoosting Regressor
Model performance for Training set
- Root Mean Squared Error: 3.0427
- Mean Absolute Error: 2.4054
- R2 Score: 0.9589
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 6.0086
- Mean Absolute Error: 4.6125
- R2 Score: 0.8516
===================================


AdaBoost Regressor
Model performance for Training set
- Root Mean Squared Error: 5.8693
- Mean Absolute Error: 4.7958
- R2 Score: 0.8472
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 6.0838
- Mean Absolute Error: 4.7848
- R2 Score: 0.8479
===================================
```

5.7 Results

In [183…

```
pd.DataFrame(list(zip(model_list, r2_list)), columns=['Model Name', 'R2_Score']).sort_va
```

Out[183]:

	Model Name	R2_Score
2	Ridge	0.880593
0	Linear Regression	0.879159
7	CatBoosting Regressor	0.851632
5	Random Forest Regressor	0.849567
8	AdaBoost Regressor	0.847895
1	Lasso	0.825320
6	XGBRegressor	0.821589
3	K-Neighbors Regressor	0.783193
4	Decision Tree	0.751026

5.8 Linear Regression

In [184…

```
lin_model = LinearRegression(fit_intercept=True)
lin_model = lin_model.fit(X_train, Y_train)
Y_pred = lin_model.predict(X_test)
score = r2_score(Y_test, Y_pred)*100
print(" Accuracy of the model is %.2f" %score)
```

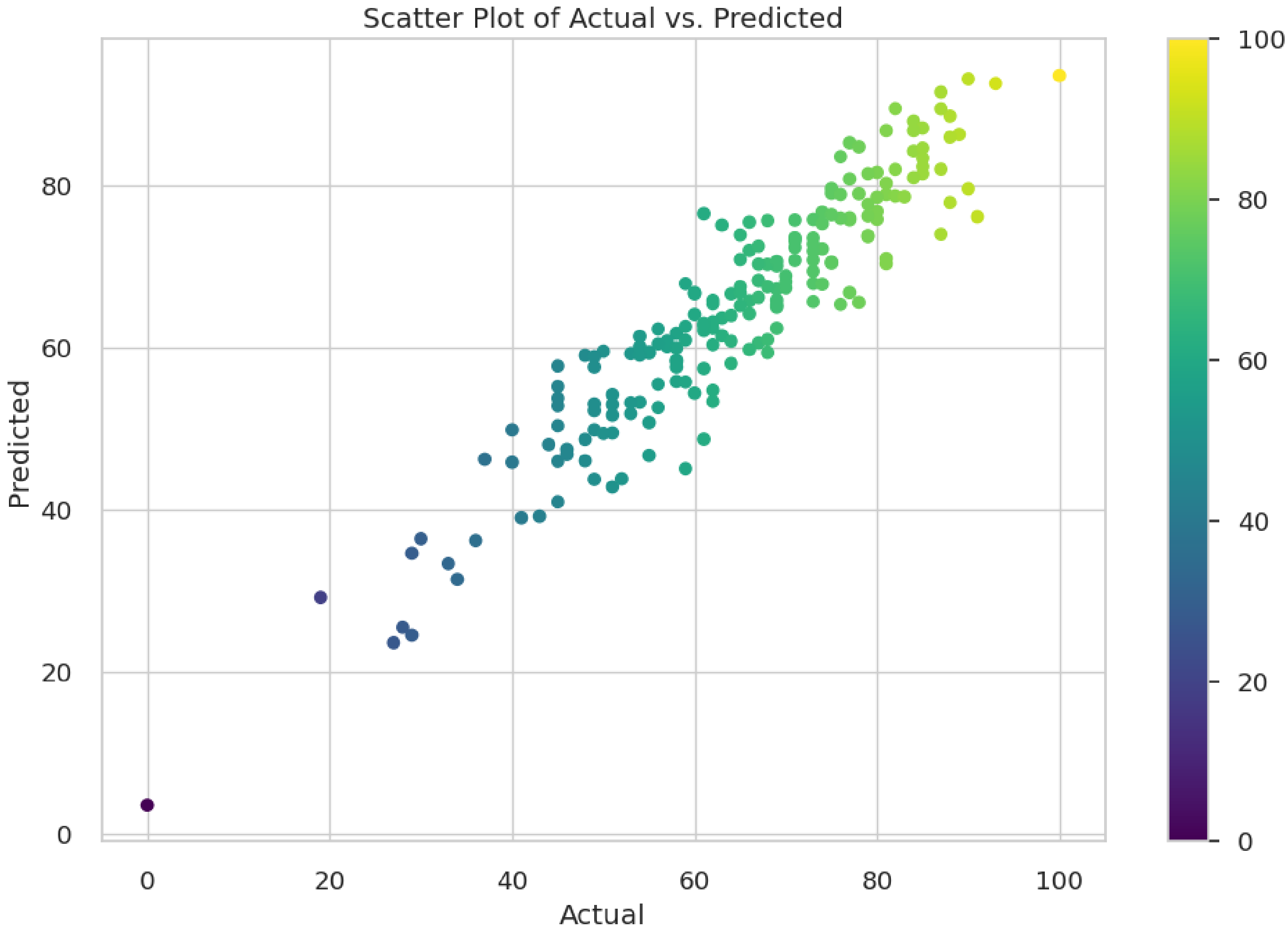
```
 Accuracy of the model is 87.92
```

5.9 Plot Y_pred and Y_test

Scatter Plot of Actual vs. Predicted

In [185…

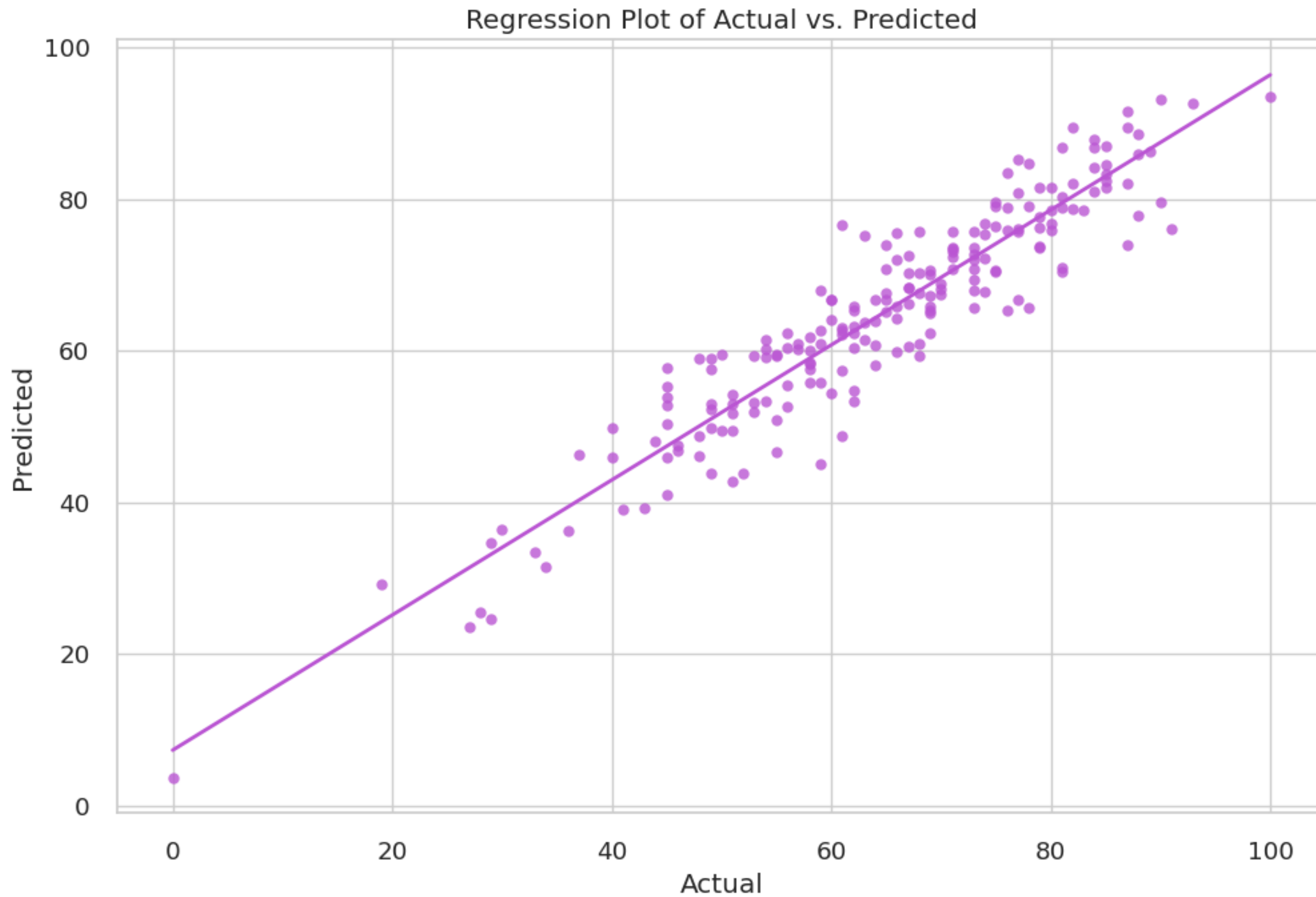
```
from matplotlib import cm
# Create the scatter plot
fig, ax = plt.subplots()
sc = ax.scatter(Y_test, Y_pred, c=Y_test, cmap=cm.viridis)
fig.colorbar(sc)

# Set the axis labels
ax.set_xlabel('Actual')
ax.set_ylabel('Predicted')
ax.set_title('Scatter Plot of Actual vs. Predicted')

plt.show()
```



Regression Plot of Actual vs. Predicted

In [186…

```
# Create the regression plot
sns.set_style('whitegrid')
sns.regplot(x=Y_test, y=Y_pred, ci=None, color='mediumorchid', line_kws={'lw':2})

# Set the axis labels and title
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Regression Plot of Actual vs. Predicted')

plt.show()
```



5.10 Difference between Actual and Predicted Values

In [187…

```
pred_df=pd.DataFrame({'Actual Value':Y_test,'Predicted Value':Y_pred,'Difference':Y_test
pred_df
```

Out[187]:

	Actual Value	Predicted Value	Difference
521	91	76.15625	14.84375
737	53	59.28125	-6.28125
740	80	76.81250	3.18750
660	74	76.71875	-2.71875
411	84	87.93750	-3.93750
...	...	...	...
408	52	43.84375	8.15625
332	62	62.40625	-0.40625
208	74	67.84375	6.15625
613	65	66.78125	-1.78125
78	61	62.68750	-1.68750

200 rows × 3 columns