Student Performance Exam Analysis

Project Description

The primary objective of this project is to analyze student performance in exams using machine learning techniques. By leveraging a classification model, we aim to predict whether a student will perform well based on various features, such as gender, race/ethnicity, parental level of education, lunch type, and test preparation course completion. This analysis will help us understand the factors that influence student performance and identify areas where students may need additional support.

The dataset used for this project includes information about students' demographics and their scores in math, reading, and writing exams. Specifically, it contains columns for gender, race/ethnicity, parental level of education, lunch type, test preparation course status, and scores in math, reading, and writing.

The methodology for this project includes several steps. First, we load and preprocess the data by encoding categorical variables and handling any missing values. Next, we perform feature engineering by calculating the average score across the three subjects and creating a binary label indicating whether a student's average score is above or below a certain threshold (e.g., 70).

Exploratory Data Analysis (EDA) is conducted to generate descriptive statistics for the scores and visualize their distribution using box plots. We also analyze the difficulty level of each subject by calculating the percentage of students scoring below a certain threshold and assess the correlation between different scores using a heatmap.

We then build a classification model using RandomForestClassifier to predict student performance. The data is split into training and testing sets, and the model is trained and evaluated using accuracy score and classification report metrics.

To gain deeper insights, we utilize various visualizations. Box plots are used to show the distribution of exam scores, revealing the spread, median, and potential outliers. Bar charts display the difficulty levels, highlighting subjects where students struggle the most. A heatmap illustrates the correlation matrix of exam scores, showing relationships between different scores. A pie chart provides a visual representation of gender distribution, while bar charts compare average scores by gender, parental education level, and test preparation course. Additionally, a scatter plot examines the relationship between reading and writing scores.

The analysis concludes by summarizing the findings, discussing the model's performance, and providing insights and recommendations. This project demonstrates how machine learning can be applied to educational data to predict and analyze student performance, offering valuable insights for educators and policymakers.

Import Library

```
import pandas as pd  # For data manipulation and analysis
import numpy as np   # For numerical computations
import seaborn as sns  # For data visualization
import matplotlib.pyplot as plt  # For plotting graphs
from sklearn.model_selection import train_test_split  # For splitting the dataset
from sklearn.preprocessing import LabelEncoder  # For encoding categorical variables
from sklearn.ensemble import RandomForestClassifier  # For Random Forest classifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix  # For model evaluation
```

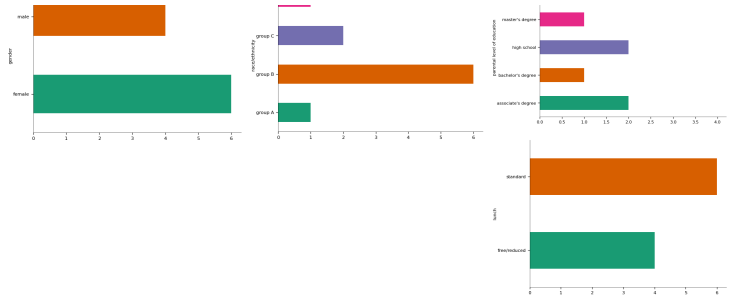
Uploading CSV file(Dateset)

```
df = pd.read_csv("14_Student Performance in Exam Analysis.csv")
```

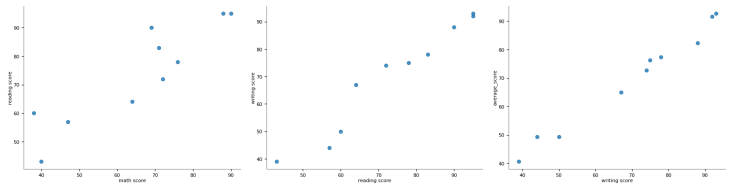
Data Preprocessing

.head()is prints the first sequence of rows of the DataFrame

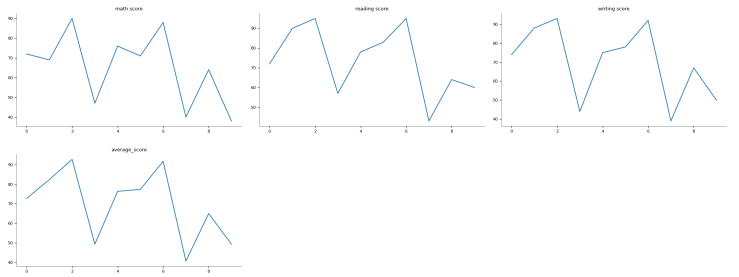
```
df.head(10)
```



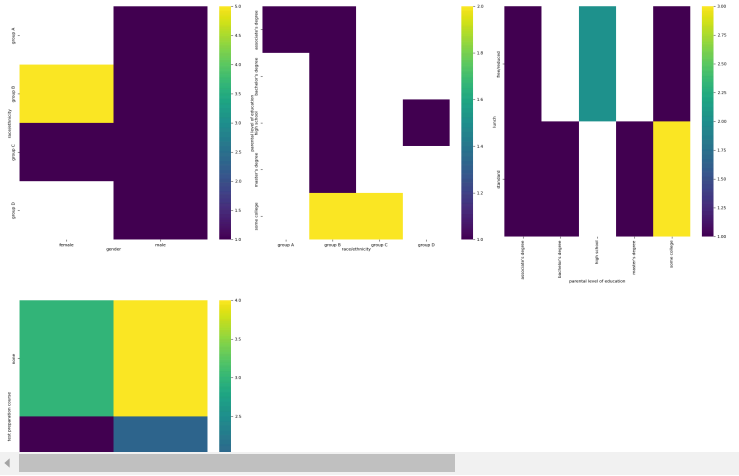
2-d distributions



Values



2-d categorical distributions



shape is used to retrieve the dimensions of a data structure (rows & columns)

```
df.shape
```

```
(1000, 9)
```

.columns is used to retrieve the column labels of a DataFrame

```
df.columns
```

```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
       'test preparation course', 'math score', 'reading score',
       'writing score', 'average_score'],
      dtype='object')
```

.dtypes is used to retrieve the data types of each column in a DataFrame

```
df.dtypes
```

```
gender                         object
race/ethnicity                 object
parental level of education    object
lunch                          object
test preparation course        object
math score                      int64
reading score                   int64
writing score                   int64
average_score                 float64
dtype: object
```

.unique is used to get the unique values in a Series, which is typically a column in a DataFrame.

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

```
array(['group B', 'group C', 'group A', 'group D', 'group E'],
      dtype=object)
```

.nunique is used to count the number of unique values in a Series

```
df["race/ethnicity"].nunique()
```

```
5
```

.describe is used to generate descriptive statistics of a DataFrame

```
df.describe()
```

	math score	reading score	writing score	average_score
count	1000.00000	1000.000000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000	67.770667
std	15.16308	14.600192	15.195657	14.257326
min	0.00000	17.000000	10.000000	9.000000
25%	57.00000	59.000000	57.750000	58.333333
50%	66.00000	70.000000	69.000000	68.333333
75%	77.00000	79.000000	79.000000	77.666667
max	100.00000	100.000000	100.000000	100.000000

.value_counts is used to count the occurrences of unique values in a Series

```
df["race/ethnicity"].value_counts
```

```
pandas.core.base.IndexOpsMixin.value_counts
def value_counts(normalize: bool=False, sort: bool=True, ascending: bool=False, bins=None, dropna:
bool=True) -> Series

Return a Series containing counts of unique values.

The resulting object will be in descending order so that the
first element is the most frequently-occurring element.
Excludes NA values by default.
```

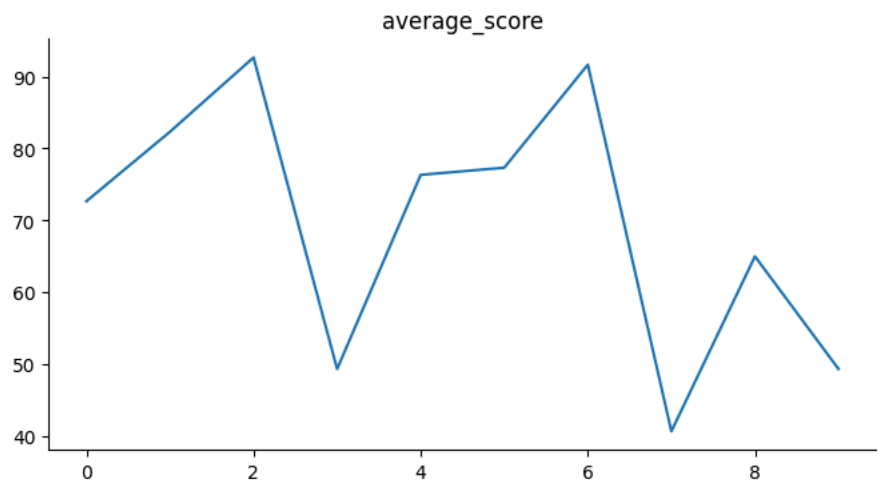
.isnull is used to identify missing or NA values in a Series or DataFrame

```
df.isnull
```

```
pandas.core.frame.DataFrame.isnull
def isnull() -> DataFrame

DataFrame.isnull is an alias for DataFrame.isna.

Detect missing values.

Return a boolean same-sized object indicating if the values are NA.
NA values, such as None or :attr:`numpy.NaN`, gets mapped to True
```

Visualizations

```
from matplotlib import pyplot as plt
_df_14['average_score'].plot(kind='line', figsize=(8, 4), title='average_score')
plt.gca().spines[['top', 'right']].set_visible(False)
```



Item Analysis

Calculate Descriptive Statistics

descriptive_stats use to calculate descriptive statistics for each subject, such as mean, median, and standard deviation. This will an overview of the distribution of scores in each subject.

```
descriptive_stats = df[['math score', 'reading score', 'writing score']].describe()

print(descriptive_stats)
```

```
       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
```

Analyze Difficulty Level

Calculate the difficulty level of each subject by examining the percentage of students who scored below a certain threshold (e.g., below 70).

```
df['average_score'] = df[['math score', 'reading score', 'writing score']].mean(axis=1)
```

Analyze Discriminatory Power

Calculate the discriminatory power of each subject by analyzing the correlation between scores in that subject and the overall average score.

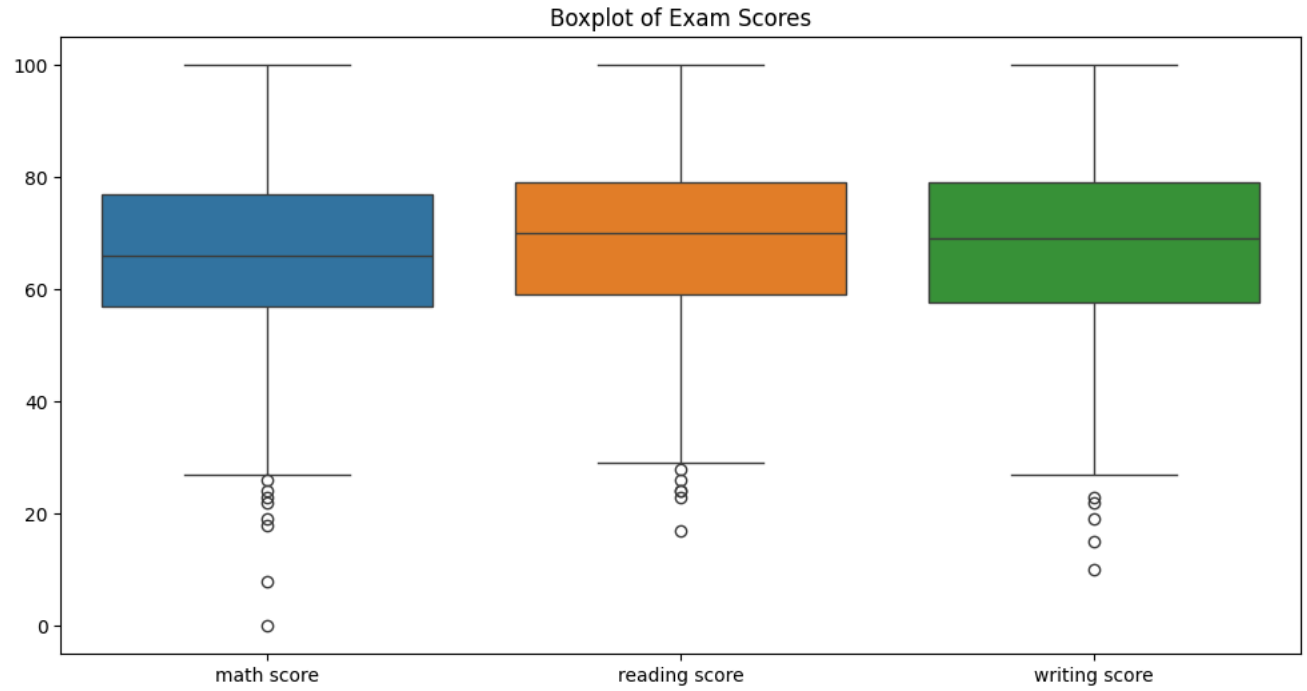
```
descriptive_stats = df[['math score', 'reading score', 'writing score']].describe()
```

Visualizations

Descriptive Statistics

```
plt.figure(figsize=(12, 6))
sns.boxplot(data=df[['math score', 'reading score', 'writing score']])
plt.title('Boxplot of Exam Scores')

print("Descriptive Statistics:")
print(descriptive_stats)
plt.show()
```

```
Descriptive Statistics:
       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
```

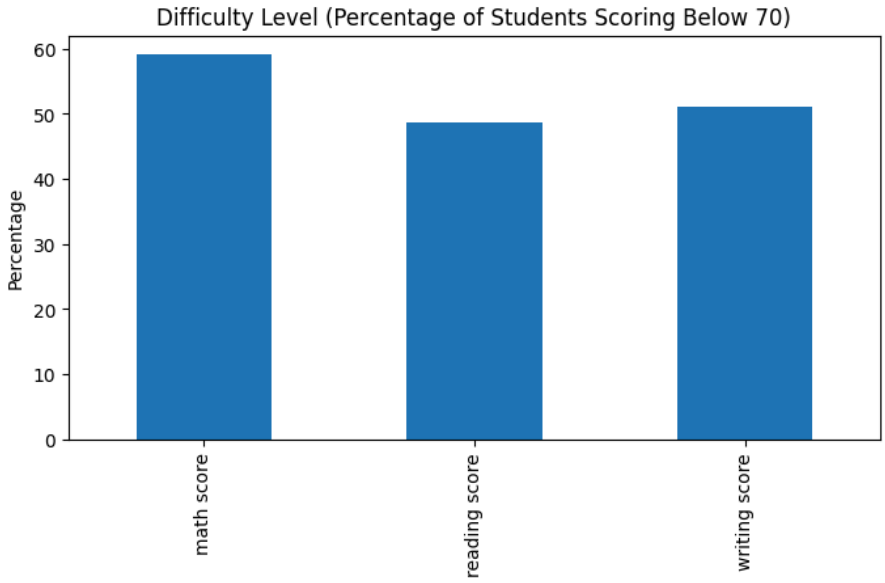


This visualization displays three boxplots representing the distribution of scores in math, reading, and writing. Each boxplot shows the minimum, first quartile, median, third quartile, and maximum scores for the respective subject. From the boxplot, it's evident that the median math score is higher than the median scores for reading and writing. Additionally, the spread of scores, as indicated by the length of the box and whiskers, is wider for math compared to reading and writing, suggesting more variability in math scores.

Difficulty Level Analysis

```
threshold = 70
difficulty_level = df[['math score', 'reading score', 'writing score']].apply(lambda x: (x < threshold).mean() * 100)

plt.figure(figsize=(8, 4))
difficulty_level.plot(kind='bar')
plt.title('Difficulty Level (Percentage of Students Scoring Below 70)')
plt.ylabel('Percentage')

print("\nDifficulty Level (Percentage of Students Scoring Below 70):")
print(difficulty_level)
plt.show()
```

```
Difficulty Level (Percentage of Students Scoring Below 70):
math score       59.1
reading score    48.7
writing score    51.1
dtype: float64
```

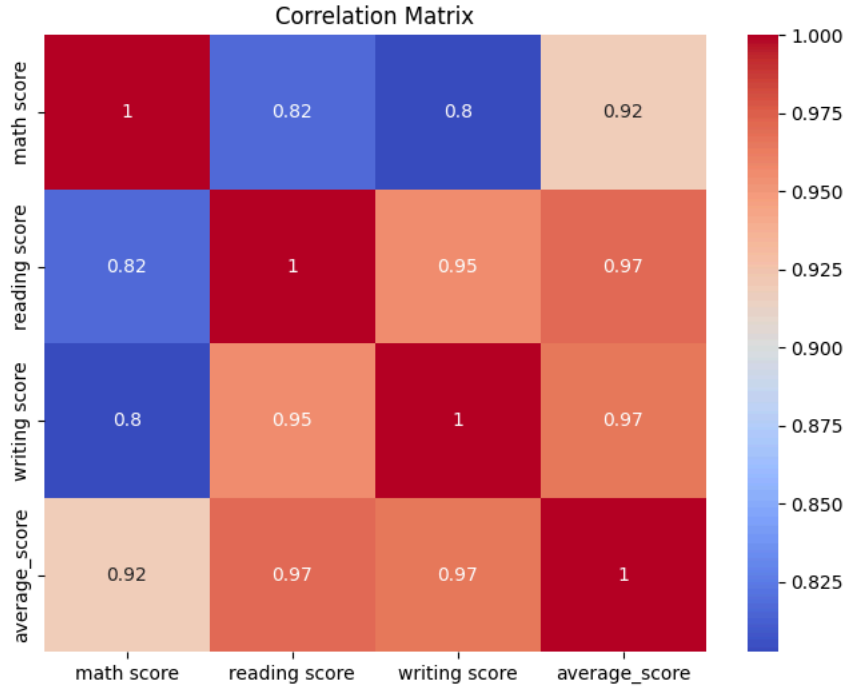


This bar chart illustrates the percentage of students scoring below 70 in math, reading, and writing. Each bar represents one of the subjects, with the height of the bar indicating the percentage of students who scored below the threshold. From the chart, it's apparent that a higher percentage of students scored below 70 in math compared to reading and writing, indicating that math may be relatively more challenging for students in this dataset.

Discriminatory Power Analysis

```
correlation = df[['math score', 'reading score', 'writing score', 'average_score']].corr()['average_score'].drop('average_score')

plt.figure(figsize=(8, 6))
sns.heatmap(df[['math score', 'reading score', 'writing score', 'average_score']].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')

print("\nDiscriminatory Power (Correlation with Overall Average Score):")
print(correlation)
plt.show()
```

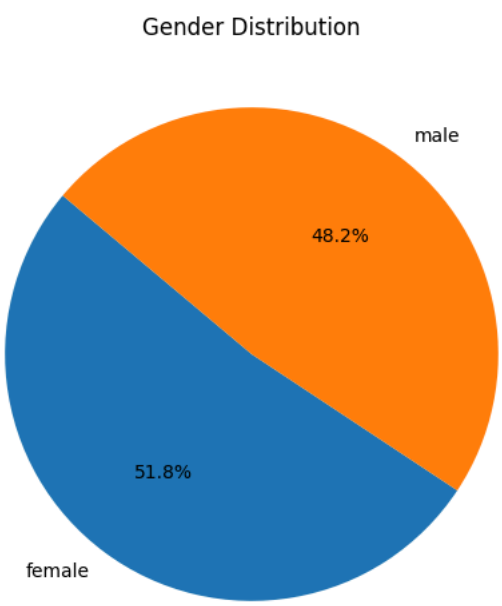
```
Discriminatory Power (Correlation with Overall Average Score):
math score       0.918746
reading score    0.970331
writing score    0.965667
Name: average_score, dtype: float64
```



The heatmap displays a correlation matrix between math, reading, writing scores, and the average score. Each cell in the heatmap represents the correlation coefficient between two variables, with warmer colors indicating stronger positive correlations and cooler colors indicating stronger negative correlations. From the heatmap, we can observe the correlation between different subject scores and the average score, identifying which subjects are more strongly correlated with overall performance.

Gender Distribution

```
gender_counts = df['gender'].value_counts()
plt.figure(figsize=(6, 6))
gender_counts.plot(kind='pie', autopct='%1.1f%%', startangle=140)
plt.title('Gender Distribution')
plt.ylabel('')
plt.show()
```

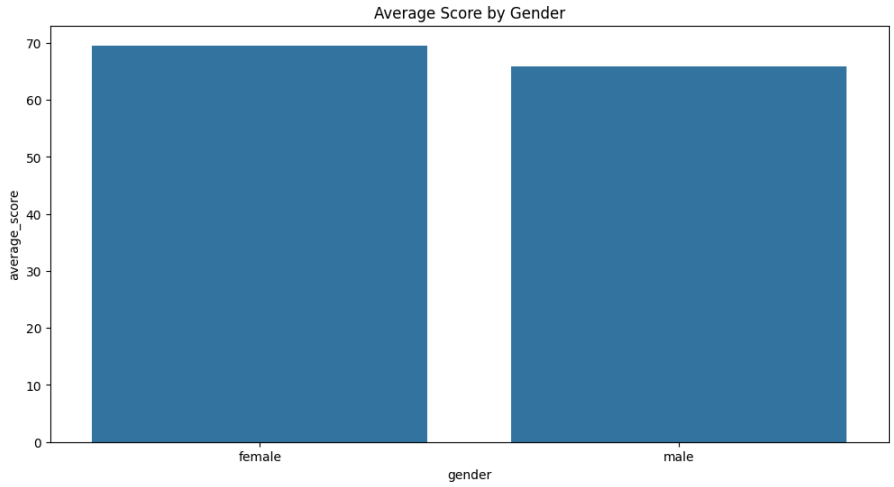


This pie chart illustrates the distribution of genders within the dataset. Each slice of the pie represents a gender category, with the size of the slice corresponding to the proportion of students in that category. The pie chart visually demonstrates the gender balance in the dataset, showing the relative frequency of male and female students.

Score Distribution by Gender

```
plt.figure(figsize=(12, 6))
sns.barplot(data=df, x='gender', y='average_score', ci=None)
plt.title('Average Score by Gender')
plt.show()
```

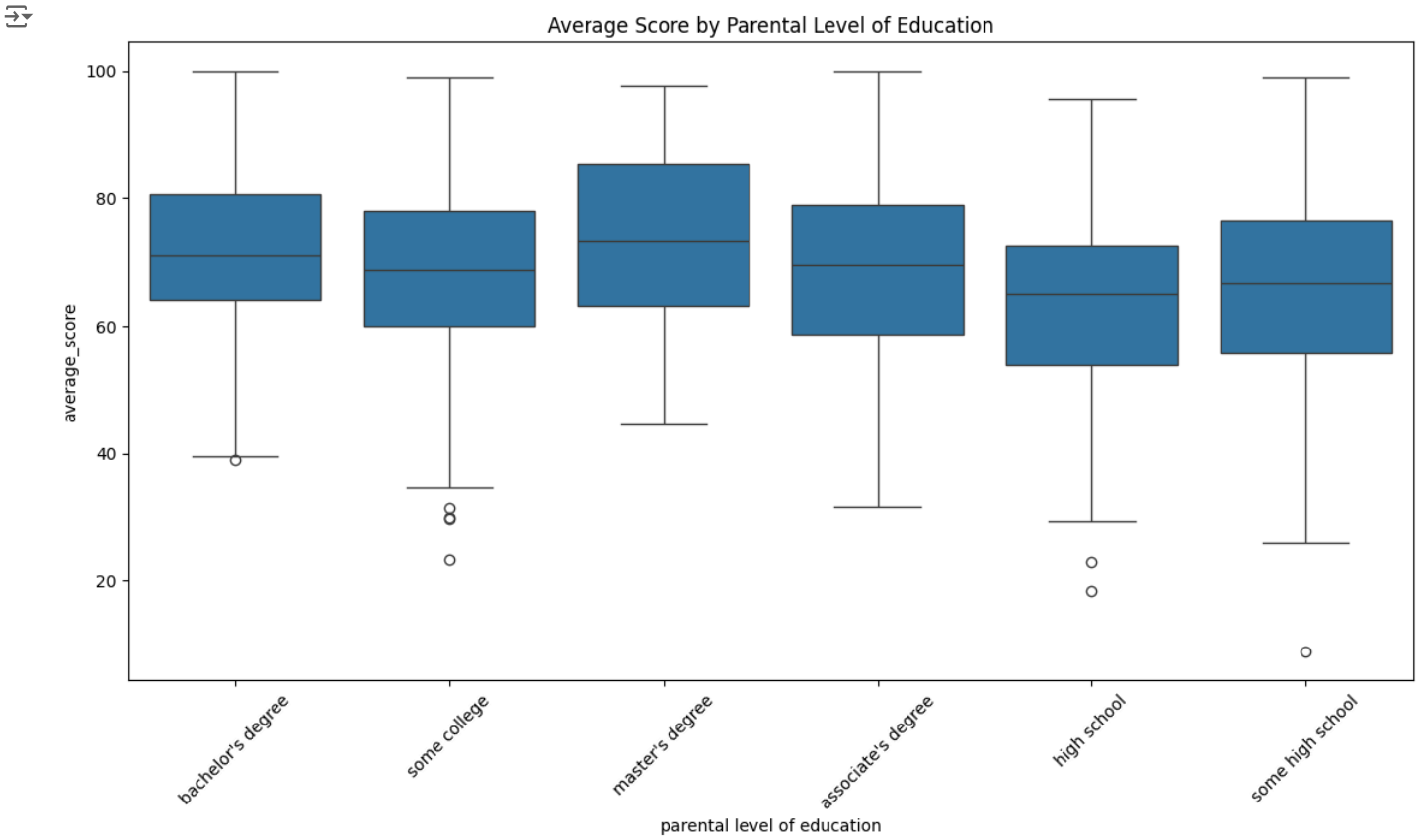
```
<ipython-input-27-3ff8631333be>:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=df, x='gender', y='average_score', ci=None)
```



This bar chart compares the average scores between male and female students. Two bars represent the average score for each gender, allowing for a visual comparison of academic performance. The chart provides insights into any disparities in performance between male and female students.

Distribution of Scores by Parental Education Level

```
plt.figure(figsize=(14, 7))
sns.boxplot(data=df, x='parental level of education', y='average_score')
plt.title('Average Score by Parental Level of Education')
plt.xticks(rotation=45)
plt.show()
```

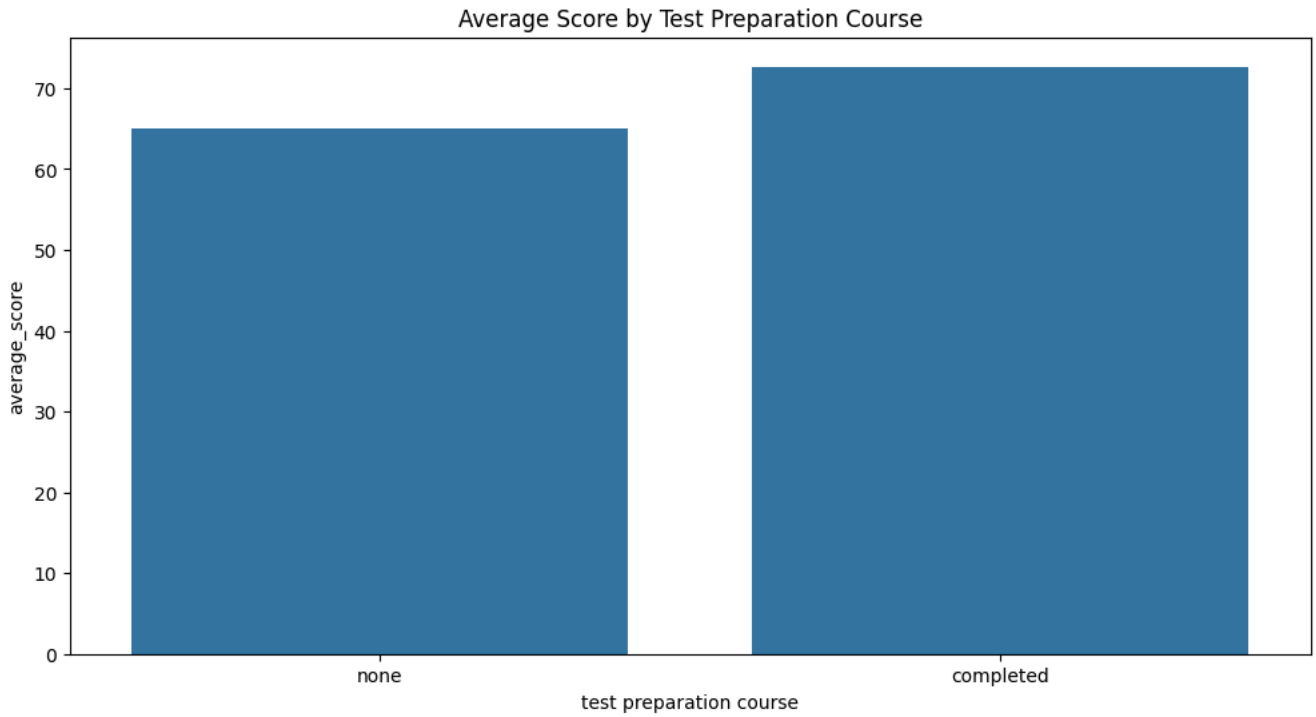


This box plot visualizes the distribution of average scores across different levels of parental education. Each box represents a parental education category, showing the spread of average scores within that category. The box plot helps identify any differences in academic performance based on parental education levels.

Score Distribution by Test Preparation

```
plt.figure(figsize=(12, 6))
sns.barplot(data=df, x='test preparation course', y='average_score', ci=None)
plt.title('Average Score by Test Preparation Course')
plt.show()
```

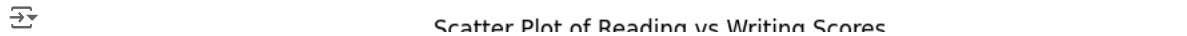
```
<ipython-input-29-498772b6f029>:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=df, x='test preparation course', y='average_score', ci=None)
```



This bar chart compares the average scores of students who completed a test preparation course with those who did not. Two bars represent the average score for each group, enabling a comparison of performance between students who received test preparation and those who did not. The chart offers insights into the effectiveness of test preparation courses in improving academic performance.

Reading vs Writing Scores

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='reading score', y='writing score', hue='gender')
plt.title('Scatter Plot of Reading vs Writing Scores')
plt.show()
```



This scatter plot visualizes the relationship between reading and writing scores for each student, with points colored by gender for differentiation. Each point represents an individual student, with their reading score plotted on the x-axis and their writing score plotted on the y-axis. The scatter plot helps identify any patterns or trends in the relationship between reading and writing scores, as well as any gender-based differences in performance.

Machine Learning Implementation

Preprocess the Data

We will preprocess the data by encoding categorical variables and handling any missing values.

```
label_encoders = {}
for column in ['gender', 'race/ethnicity', 'parental level of education', 'lunch', 'test preparation course']:
    label_encoders[column] = LabelEncoder()
    df[column] = label_encoders[column].fit_transform(df[column])

df.head()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	average_score
0	0	1	1	1	1	72	72	74	72.666667
1	0	2	4	1	0	69	90	88	82.333333
2	0	1	3	1	1	90	95	93	92.666667
3	1	0	0	0	1	47	57	44	49.333333
4	1	2	4	1	1	76	78	75	76.333333

Define Features and Labels

We will define our features (X) and labels (y). For simplicity, we will predict if a student scores above 70 in the average of their math, reading, and writing scores.

```
df['average_score'] = df[['math score', 'reading score', 'writing score']].mean(axis=1)
```