data602_final_project

December 17, 2023

Kelly Eng # DATA602 Final Project

0.1 Abstract

This project examines the role of level of parental education and type of lunch received as predictors of exam scores and whether there is a difference in those who took a test preparation course beforehand and those who do not. Visualization of the data, data analysis, and linear regression was performed on the dataset from Kaggle. This analysis showed there was a weak correlation between exam scores and level of parental education and type of lunch received. A difference in higher scores from those who have taken a test preparation course leads to the conclusion that students perform better when they have access to outside test preparatory courses.

0.2 Introduction

Exams are a huge part of people's life, students have to take them whether they like them or not. Not everyone have access to the same resources so this can have an effect on performance. In order to do well, some students are able to afford test preparation courses for exams such as the SAT, ACT, Regents, and the SHSAT. This is relevant to me because the high school I went to required the SHSAT to get into. Getting into it was judged solely by the exam. A high score can be life changing when it comes to college admissions. This can affect things such as scholarship and make someone appear competitive. I will be examining a dataset from Kaggle about student performance on exams.

Link to Dataset: https://www.kaggle.com/datasets/spscientist/students-performance-in-exams

0.2.1 Research Question

Which type of students perform best on exams? How does this differ among people who took a test preparation course and those who do not? Does parents' education level and the type of lunch a student receive affects this? This will be measured by the math, reading, and writing scores.

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Loads the dataset into a Pandas dataframe
df = pd.read_csv("StudentsPerformance.csv")
```

0.2.2 Summary Statistics

[ ]:
```
df.describe()
```

[ ]:
```
       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
```

0.3 Data Wrangling

[ ]:
```
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
```

[ ]:
```
# Drops all NA values
df = df.dropna(how = "any")
# Removes the gender & race/ethnicity column inplace
df.drop(columns=['gender', 'race/ethnicity'], inplace=True)

# HashMaps to map categorical values to represent them as numeric
lunch_mapping = {"free/reduced": 0, "standard": 1}
test_prep_mapping = {"none": 0, "completed": 1}
parental_level_of_education_mapping = {"some high school": 0, "high school": 1,␣
 ↪"some college": 2, "associate's degree": 3,
                                        "bachelor's degree": 4, "master's␣
 ↪degree": 5}
```

```
# Create new columns in the dataframe that maps categorical values to map them␣
 ↪to numerics
df['numeric_parental_level_of_education'] = df['parental level of education'].
 ↪map(parental_level_of_education_mapping)
df['lunch_type'] = df['lunch'].map(lunch_mapping)
df['test_prep'] = df['test preparation course'].map(test_prep_mapping)
df.sort_values(by=['numeric_parental_level_of_education', 'test_prep',␣
 ↪'lunch_type'], inplace=True)

# Prints the data type of each column in the Pandas dataframe
print(df.dtypes)
```

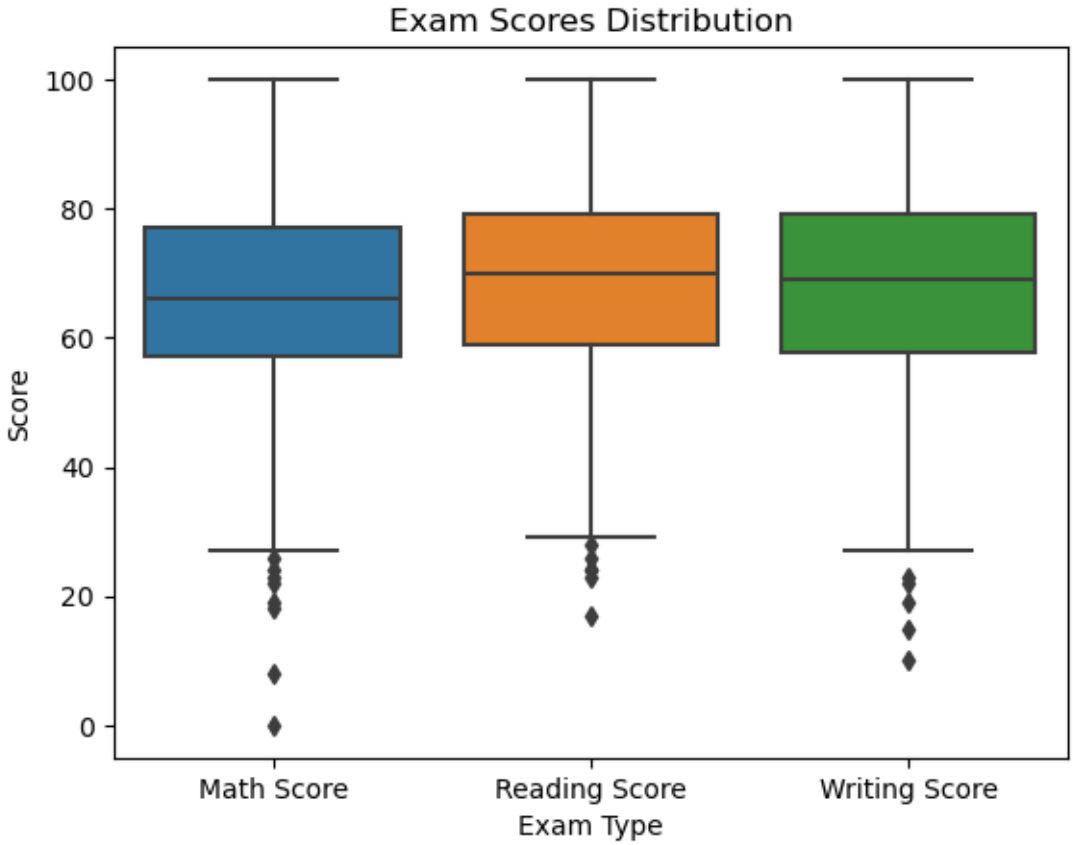
```
parental level of education           object
lunch                                 object
test preparation course               object
math score                             int64
reading score                          int64
writing score                          int64
numeric_parental_level_of_education    int64
lunch_type                             int64
test_prep                              int64
dtype: object
```

[ ]:
```
print(df.head())
```

```
   parental level of education         lunch test preparation course  \
17           some high school  free/reduced                     none
37           some high school  free/reduced                     none
59           some high school  free/reduced                     none
61           some high school  free/reduced                     none
66           some high school  free/reduced                     none

    math score  reading score  writing score  \
17          18             32             28
37          50             64             59
59           0             17             10
61          39             39             34
66          45             37             37

    numeric_parental_level_of_education  lunch_type  test_prep
17                                    0           0          0
37                                    0           0          0
59                                    0           0          0
61                                    0           0          0
66                                    0           0          0
```

0.4 Exploratory Data Analysis

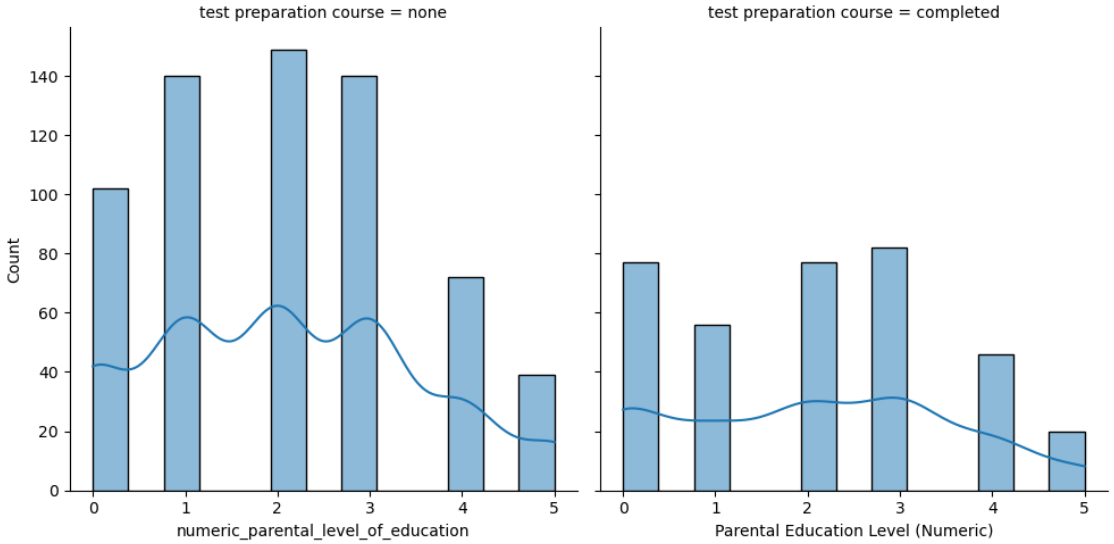
[ ]:
```
# Box plot showing the interquartile range for math, reading, and writing scores
sns.boxplot(data=df[['math score', 'reading score', 'writing score']])
plt.xticks(ticks=[0, 1, 2], labels=['Math Score', 'Reading Score', 'Writing␣
 ↪Score'])
plt.xlabel("Exam Type")
plt.ylabel("Score")
plt.title("Exam Scores Distribution")
plt.show()
```



0.4.1 Display plot of education levels with density curve for the two groups of test prep

[ ]:
```
sns.displot(data=df, x="numeric_parental_level_of_education", col="test␣
 ↪preparation course", kde=True)
plt.xlabel("Parental Education Level (Numeric)")

plt.show()
```

```
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

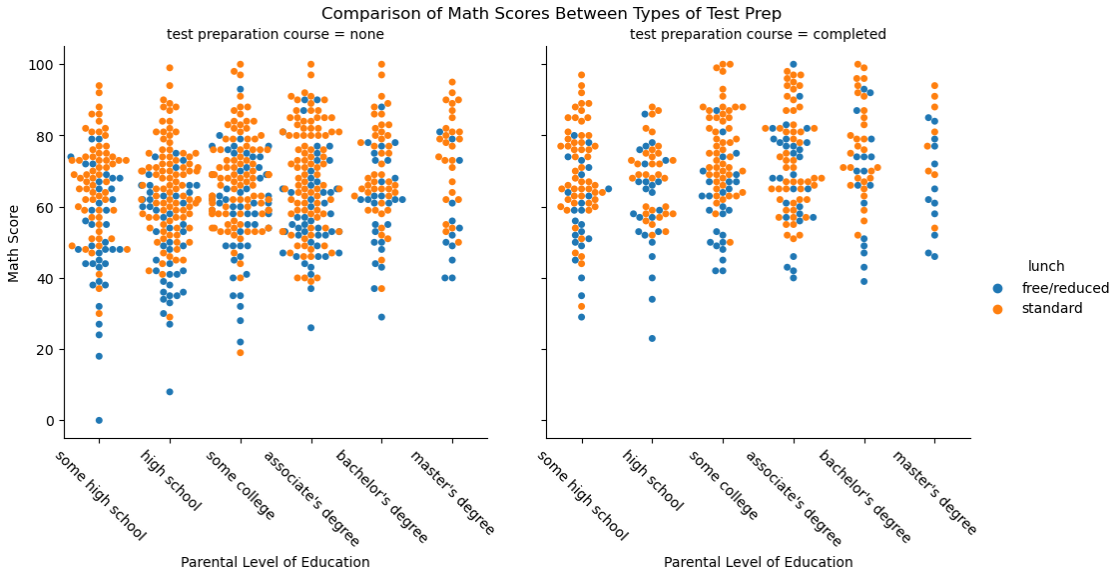


0.4.2 Swarm Plots to Compare Exam Scores

[ ]:
```
# Swarm Plot to compare categorical variables effects on math scores
math_compare = sns.catplot(x="parental level of education", y="math score",␣
 ↪data=df, hue="lunch", col="test preparation course", kind="swarm")
math_compare.set_xticklabels(rotation=-45)
math_compare.set_axis_labels("Parental Level of Education", "Math Score")
plt.suptitle("Comparison of Math Scores Between Types of Test Prep")
plt.subplots_adjust(top=0.9)

plt.show()

# Swarm Plot to compare categorical variables effects on reading scores
read_compare = sns.catplot(x="parental level of education", y="reading score",␣
 ↪data=df, hue="lunch", col="test preparation course", kind="swarm")
read_compare.set_xticklabels(rotation=-45)
read_compare.set_axis_labels("Parental Level of Education", "Reading Score")
plt.suptitle("Comparison of Reading Scores Between Types of Test Prep")
plt.subplots_adjust(top=0.9)

plt.show()

# Swarm Plot to compare categorical variables effects on writing scores
```

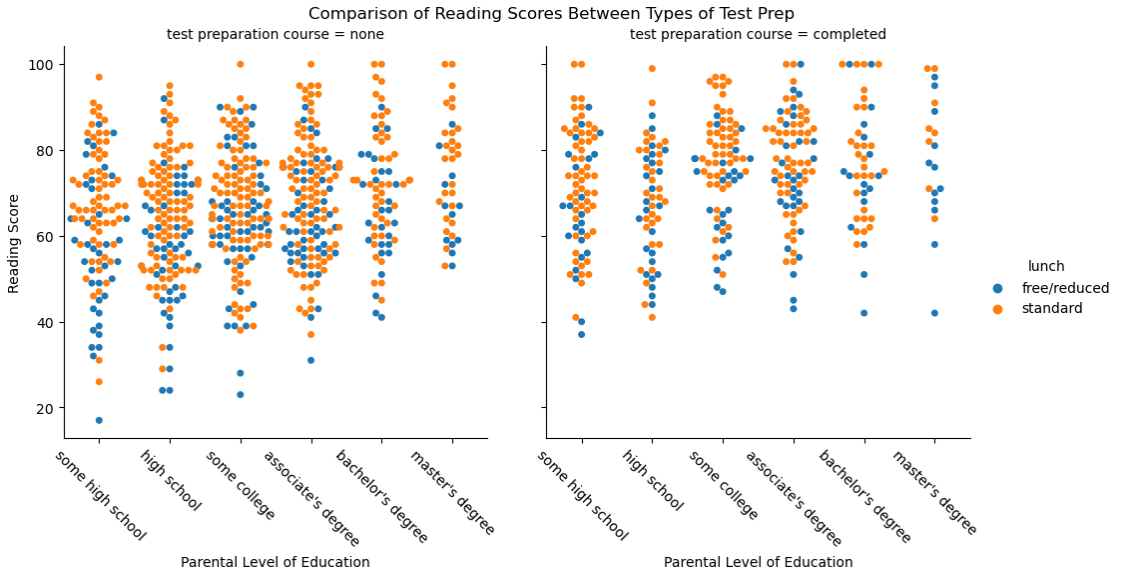
```
write_compare = sns.catplot(x="parental level of education", y="writing score",␣
 ↪data=df, hue="lunch", col="test preparation course", kind="swarm")
write_compare.set_xticklabels(rotation=-45)
write_compare.set_axis_labels("Parental Level of Education", "Writing Score")
plt.suptitle("Comparison of Writing Scores Between Types of Test Prep")
plt.subplots_adjust(top=0.9)

plt.show()
```

```
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\categorical.py:3544:
UserWarning: 14.1% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\categorical.py:3544:
UserWarning: 18.8% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\categorical.py:3544:
UserWarning: 7.9% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\categorical.py:3544:
UserWarning: 15.4% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

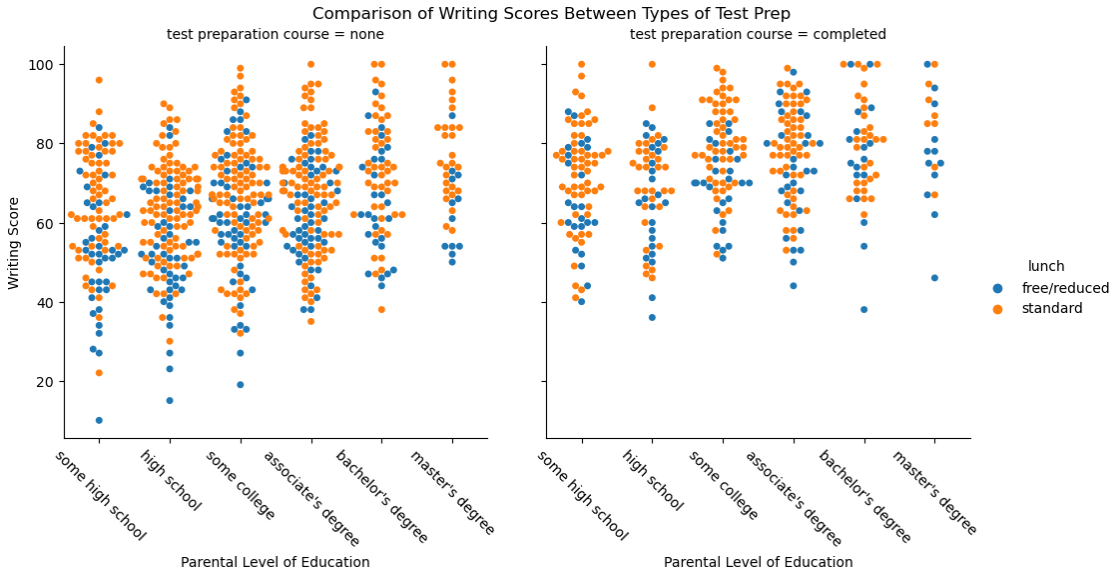



```
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\categorical.py:3544:
UserWarning: 5.4% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\categorical.py:3544:
UserWarning: 7.1% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\categorical.py:3544:
UserWarning: 7.4% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\categorical.py:3544:
UserWarning: 5.7% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\categorical.py:3544:
UserWarning: 7.1% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

```
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\categorical.py:3544:
UserWarning: 7.4% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\categorical.py:3544:
UserWarning: 5.7% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```




```
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\categorical.py:3544:
UserWarning: 5.7% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\Guestperson\anaconda3\Lib\site-packages\seaborn\categorical.py:3544:
UserWarning: 5.7% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



0.5 Data Analysis and Machine Learning

[ ]:
```
no_test_prep = df[df.test_prep == 0]
yes_test_prep = df[df.test_prep == 1]

no_test_prep.groupby(['lunch', 'parental level of education'])['math score'].
↪mean()
```

[ ]:
```
lunch         parental level of education
free/reduced  associate's degree             59.062500
              bachelor's degree              60.629630
              high school                    51.652174
              master's degree                56.666667
              some college                   59.830189
              some high school               51.552632
standard      associate's degree             68.967391
              bachelor's degree              70.666667
              high school                    65.563830
              master's degree                74.925926
              some college                   67.687500
              some high school               66.734375
Name: math score, dtype: float64
```

[ ]:
```
yes_test_prep.groupby(['lunch', 'parental level of education'])['math score'].
↪mean()
```

[ ]:
```
lunch         parental level of education
free/reduced  associate's degree             68.482759
```

```
                bachelor's degree              66.764706
                high school                    60.000000
                master's degree                65.666667
                some college                   60.730769
                some high school               57.869565
standard        associate's degree             73.660377
                bachelor's degree              77.103448
                high school                    68.750000
                master's degree                78.000000
                some college                   76.921569
                some high school               70.462963
Name: math score, dtype: float64
```

[ ]:
```
no_test_prep.groupby(['lunch', 'parental level of education'])['reading score'].
↪mean()
```

[ ]:
```
lunch         parental level of education
free/reduced  associate's degree             64.229167
              bachelor's degree              65.851852
              high school                    57.478261
              master's degree                66.416667
              some college                   63.169811
              some high school               56.552632
standard      associate's degree             69.750000
              bachelor's degree              73.466667
              high school                    66.372340
              master's degree                77.222222
              some college                   67.697917
              some high school               68.281250
Name: reading score, dtype: float64
```

[ ]:
```
yes_test_prep.groupby(['lunch', 'parental level of education'])['reading␣
↪score'].mean()
```

[ ]:
```
lunch         parental level of education
free/reduced  associate's degree             73.310345
              bachelor's degree              73.823529
              high school                    65.458333
              master's degree                74.166667
              some college                   68.961538
              some high school               66.000000
standard      associate's degree             77.735849
              bachelor's degree              78.448276
              high school                    69.625000
              master's degree                84.375000
              some college                   79.568627
              some high school               73.055556
```

```
Name: reading score, dtype: float64
```

[ ]:
```
no_test_prep.groupby(['lunch', 'parental level of education'])['writing score'].
↪mean()
```

[ ]:
```
lunch         parental level of education
free/reduced  associate's degree            61.625000
              bachelor's degree             65.148148
              high school                   53.152174
              master's degree               63.250000
              some college                  60.924528
              some high school              52.815789
standard      associate's degree            68.043478
              bachelor's degree             72.888889
              high school                   63.659574
              master's degree               77.925926
              some college                  67.052083
              some high school              65.468750
Name: writing score, dtype: float64
```

[ ]:
```
yes_test_prep.groupby(['lunch', 'parental level of education'])['writing␣
↪score'].mean()
```

[ ]:
```
lunch         parental level of education
free/reduced  associate's degree            73.310345
              bachelor's degree             75.705882
              high school                   65.541667
              master's degree               76.500000
              some college                  69.384615
              some high school              65.565217
standard      associate's degree            78.735849
              bachelor's degree             80.448276
              high school                   69.937500
              master's degree               85.500000
              some college                  80.156863
              some high school              72.407407
Name: writing score, dtype: float64
```

[ ]:
```
no_test_prep.describe()
```

[ ]:
```
       math score  reading score  writing score  \
count  642.000000     642.000000     642.000000
mean    64.077882      66.534268      64.504673
std     15.192376      14.463885      14.999661
min      0.000000      17.000000      10.000000
25%     54.000000      57.000000      54.000000
50%     64.000000      67.000000      65.000000
```

```
75%     74.750000      76.000000      74.000000
max    100.000000     100.000000     100.000000
```

```
       numeric_parental_level_of_education  lunch_type  test_prep
count                           642.000000  642.000000      642.0
mean                              2.088785    0.651090        0.0
std                               1.436074    0.476997        0.0
min                               0.000000    0.000000        0.0
25%                               1.000000    0.000000        0.0
50%                               2.000000    1.000000        0.0
75%                               3.000000    1.000000        0.0
max                               5.000000    1.000000        0.0
```

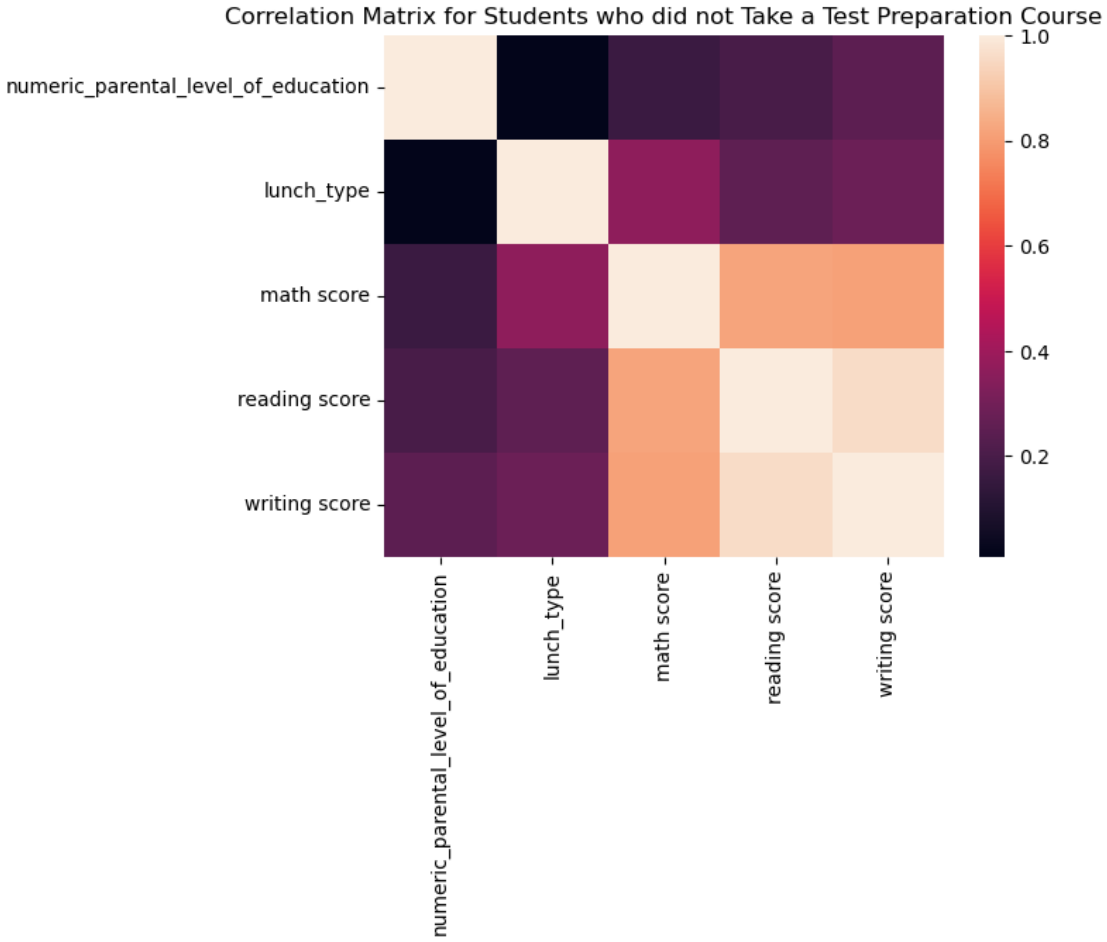
[ ]:
```
yes_test_prep.describe()
```

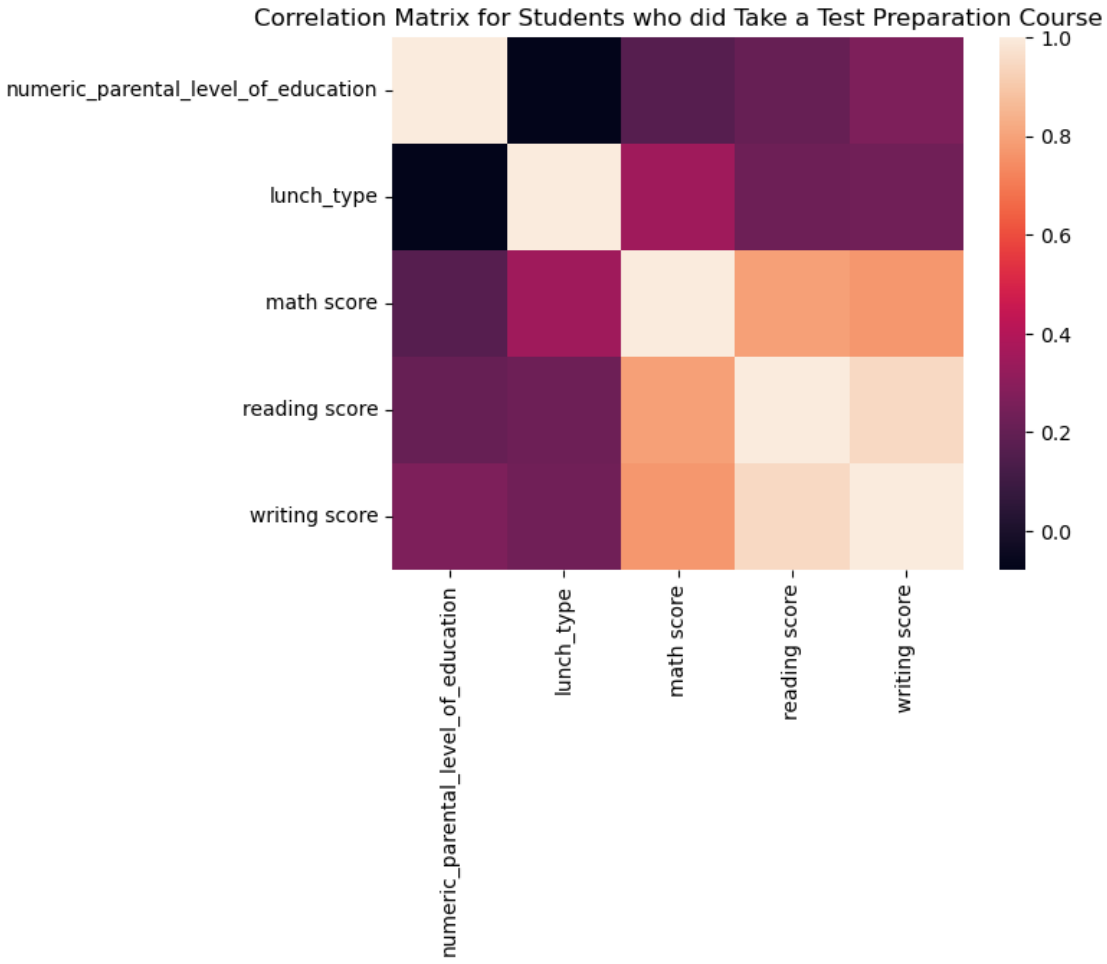
[ ]:
```
       math score  reading score  writing score  \
count  358.000000     358.000000     358.000000
mean    69.695531      73.893855      74.418994
std     14.444699      13.638384      13.375335
min     23.000000      37.000000      36.000000
25%     60.000000      65.000000      66.000000
50%     69.000000      75.000000      76.000000
75%     79.000000      84.000000      83.000000
max    100.000000     100.000000     100.000000

       numeric_parental_level_of_education  lunch_type  test_prep
count                           358.000000  358.000000      358.0
mean                              2.067039    0.634078        1.0
std                               1.504793    0.482362        0.0
min                               0.000000    0.000000        1.0
25%                               1.000000    0.000000        1.0
50%                               2.000000    1.000000        1.0
75%                               3.000000    1.000000        1.0
max                               5.000000    1.000000        1.0
```

0.5.1 Correlation

[ ]:
```
numeric_cols = ['numeric_parental_level_of_education', 'lunch_type', 'math␣
↪score', 'reading score', 'writing score']

# Subset to only get numeric values
no_test_prep = no_test_prep.loc[:, numeric_cols]
yes_test_prep = yes_test_prep.loc[:, numeric_cols]

no_test_prep_corr = no_test_prep.corr()
yes_test_prep_corr = yes_test_prep.corr()
```

```
sns.heatmap(no_test_prep_corr)
plt.title("Correlation Matrix for Students who did not Take a Test Preparation␣
  ↪Course")
plt.show()

sns.heatmap(yes_test_prep_corr)
plt.title("Correlation Matrix for Students who did Take a Test Preparation␣
  ↪Course")
plt.show()
```





0.5.2 Linear Regression

[ ]:
```
no_test_prep_numeric = no_test_prep[['numeric_parental_level_of_education',␣
 ↪'lunch_type']]
no_test_prep_math = no_test_prep['math score']
no_test_prep_reading = no_test_prep['reading score']
no_test_prep_writing = no_test_prep['writing score']
yes_test_prep_numeric = yes_test_prep[['numeric_parental_level_of_education',␣
 ↪'lunch_type']]
yes_test_prep_math = yes_test_prep['math score']
yes_test_prep_reading = yes_test_prep['reading score']
yes_test_prep_writing = yes_test_prep['writing score']

X = no_test_prep_numeric
y = no_test_prep_math

# General Format to train the model
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=2021)
no_test_prep_math_model = LinearRegression()
no_test_prep_math_model.fit(X_train, y_train)
predictions = no_test_prep_math_model.predict(X_test)
no_test_prep_math_score = no_test_prep_math_model.score(X_test, y_test)
no_test_prep_math_rmse = mean_squared_error(y_test, predictions, squared=False)

y = no_test_prep_reading

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=2021)
no_test_prep_reading_model = LinearRegression()
no_test_prep_reading_model.fit(X_train, y_train)
predictions = no_test_prep_reading_model.predict(X_test)
no_test_prep_reading_score = no_test_prep_reading_model.score(X_test, y_test)
no_test_prep_reading_rmse = mean_squared_error(y_test, predictions,␣
 ↪squared=False)

y = no_test_prep_writing

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=2021)
no_test_prep_writing_model = LinearRegression()
no_test_prep_writing_model.fit(X_train, y_train)
predictions = no_test_prep_writing_model.predict(X_test)
no_test_prep_writing_score = no_test_prep_writing_model.score(X_test, y_test)
no_test_prep_writing_rmse = mean_squared_error(y_test, predictions,␣
 ↪squared=False)

X = yes_test_prep_numeric
y = yes_test_prep_math

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=2021)
yes_test_prep_math_model = LinearRegression()
yes_test_prep_math_model.fit(X_train, y_train)
predictions = yes_test_prep_math_model.predict(X_test)
yes_test_prep_math_score = yes_test_prep_math_model.score(X_test, y_test)
yes_test_prep_math_rmse = mean_squared_error(y_test, predictions, squared=False)

y = yes_test_prep_reading

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=2021)
yes_test_prep_reading_model = LinearRegression()
```

```
yes_test_prep_reading_model.fit(X_train, y_train)
predictions = yes_test_prep_reading_model.predict(X_test)
yes_test_prep_reading_score = yes_test_prep_reading_model.score(X_test, y_test)
yes_test_prep_reading_rmse = mean_squared_error(y_test, predictions,␣
 ↪squared=False)

y = yes_test_prep_writing

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=2021)
yes_test_prep_writing_model = LinearRegression()
yes_test_prep_writing_model.fit(X_train, y_train)
predictions = yes_test_prep_writing_model.predict(X_test)
yes_test_prep_writing_score = yes_test_prep_writing_model.score(X_test, y_test)
yes_test_prep_writing_rmse = mean_squared_error(y_test, predictions,␣
 ↪squared=False)

print(f"No Test Prep - Math\n R^2: {no_test_prep_math_score}, Root Mean Squared␣
 ↪Error: {no_test_prep_math_rmse}")
print(f"No Test Prep - Reading\n R^2: {no_test_prep_reading_score}, Root Mean␣
 ↪Squared Error: {no_test_prep_reading_rmse}")
print(f"No Test Prep - Writing\n R^2: {no_test_prep_writing_score}, Root Mean␣
 ↪Squared Error: {no_test_prep_writing_rmse}")
print(f"Yes Test Prep - Math\n R^2: {yes_test_prep_math_score}, Root Mean␣
 ↪Squared Error: {yes_test_prep_math_rmse}")
print(f"Yes Test Prep - Reading\n R^2: {yes_test_prep_reading_score}, Root Mean␣
 ↪Squared Error: {yes_test_prep_reading_rmse}")
print(f"Yes Test Prep - Writing\n R^2: {yes_test_prep_writing_score}, Root Mean␣
 ↪Squared Error: {yes_test_prep_writing_rmse}")
```

```
No Test Prep - Math
 R^2: 0.16605703695010354, Root Mean Squared Error: 13.47256875683942
No Test Prep - Reading
 R^2: 0.10367569607741156, Root Mean Squared Error: 14.491694143528951
No Test Prep - Writing
 R^2: 0.15201011569866762, Root Mean Squared Error: 14.649774406919747
Yes Test Prep - Math
 R^2: 0.18952909562774378, Root Mean Squared Error: 13.4198353867764
Yes Test Prep - Reading
 R^2: 0.12250922414701149, Root Mean Squared Error: 13.552201699845652
Yes Test Prep - Writing
 R^2: 0.16430280996411262, Root Mean Squared Error: 13.765108686046368
```

0.6 Conclusions

There is a difference in exam scores for those who take a test prep course beforehand and those who do not. Those who took a test preparation course generally score higher than those who

don't. However a student's level of parental education and type of lunch they receive does not correlate strongly with performance on exams so there may be other factors that affect performance. Limitations of this analysis is that there may be other factors that influences student performance that was not included in the dataset. More insights can be gained if there were more variables to account for such as income, amount of time spent on studying, etc.