In [1]:

```python
# importing all the required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
```

In [2]:

```python
# importing the required cdv file that has the data
student_exams_df = pd.read_csv('/Users/saiyer33/Downloads/Student_Exam_Performance.csv')
student_exams_df.head()
```

Out[2]:

	race/ethnicity	parental level of education	lunch	test preparation course	math percentage	reading score percentage	writing score percentage	sex
0	group B	bachelor's degree	standard	none	0.72	0.72	0.74	F
1	group C	some college	standard	completed	0.69	0.90	0.88	F
2	group B	master's degree	standard	none	0.90	0.95	0.93	F
3	group A	associate's degree	free/reduced	none	0.47	0.57	0.44	M
4	group C	some college	standard	none	0.76	0.78	0.75	M

In [3]:

```python
# formatting the position of the columns in the dataframe to group all the grades at the end
student_exams_df = student_exams_df[["race/ethnicity","parental level of education", "lunch","sex","test prepar
```

In [4]:

```python
#printing the dataframe
student_exams_df.head()
```

Out[4]:

	race/ethnicity	parental level of education	lunch	sex	test preparation course	math percentage	reading score percentage	writing score percentage
0	group B	bachelor's degree	standard	F	none	0.72	0.72	0.74
1	group C	some college	standard	F	completed	0.69	0.90	0.88
2	group B	master's degree	standard	F	none	0.90	0.95	0.93
3	group A	associate's degree	free/reduced	M	none	0.47	0.57	0.44
4	group C	some college	standard	M	none	0.76	0.78	0.75

In [5]:

```python
# pultiplting the percentage values by 100 to get score values between 0 and 100
student_exams_df["math percentage"] = 100*student_exams_df["math percentage"]
student_exams_df["reading score percentage"] = 100*student_exams_df["reading score percentage"]
student_exams_df["writing score percentage"] = 100*student_exams_df["writing score percentage"]
```

In [6]:

```python
#printing some relevant statistics from the dataframe
student_exams_df.describe()
```

Out[6]:

	math percentage	reading score percentage	writing score percentage
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

In [7]:

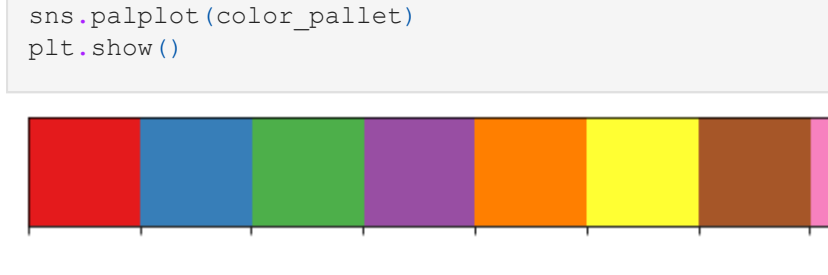
```python
# making a new column for the average score of a student and calculating it
average_score = np.sum(student_exams_df[['math percentage','reading score percentage','writing score percentage
average_score=average_score/3
student_exams_df['average score'] = average_score
#printing the dataframe
student_exams_df.head()
```

Out[7]:

	race/ethnicity	parental level of education	lunch	sex	test preparation course	math percentage	reading score percentage	writing score percentage	average score
0	group B	bachelor's degree	standard	F	none	72.0	72.0	74.0	72.666667
1	group C	some college	standard	F	completed	69.0	90.0	88.0	82.333333
2	group B	master's degree	standard	F	none	90.0	95.0	93.0	92.666667
3	group A	associate's degree	free/reduced	M	none	47.0	57.0	44.0	49.333333
4	group C	some college	standard	M	none	76.0	78.0	75.0	76.333333

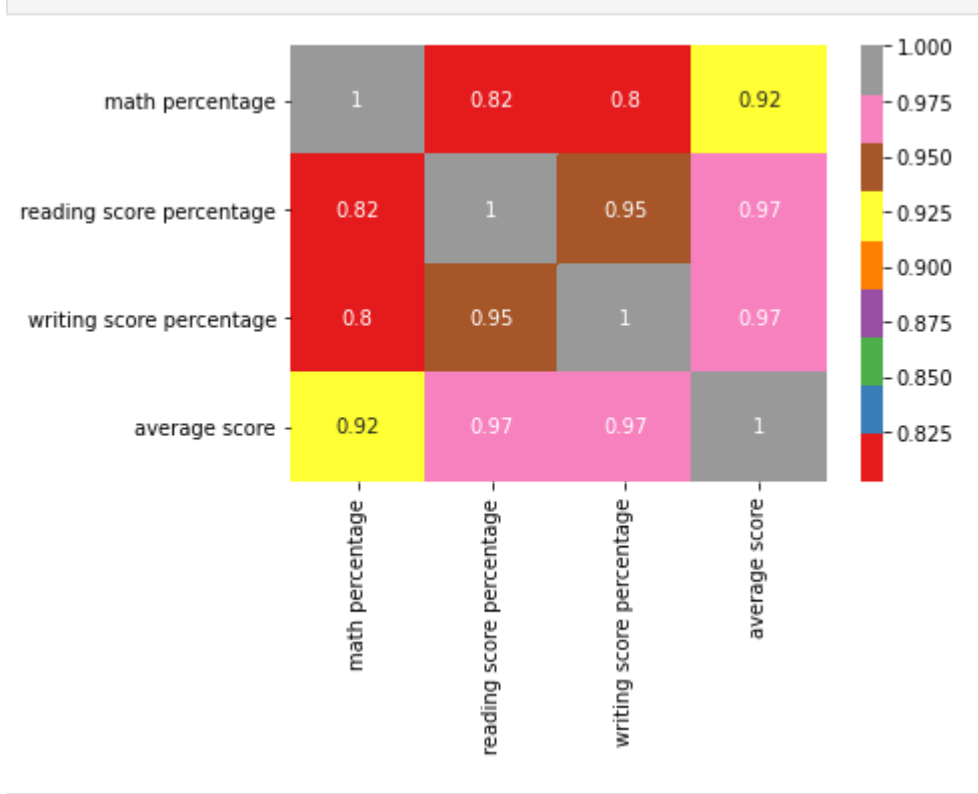
In [8]:

```python
# generating a relevent color pallet in order to see contrasts in a heatmap
color_pallet = sns.color_palette(palette='Set1')
sns.palplot(color_pallet)
plt.show()
```



In [9]:

```python
# heatmap to find any connection between the various subject scores and the average score
sns.heatmap(student_exams_df.corr(),annot=True,cmap=color_pallet)
plt.show()
```



In [10]:

```python
# clubbing the different score and subject columns into one column called score and subject respectively
student_exams_df_clubbed = student_exams_df.melt(id_vars=["race/ethnicity","parental level of education","lunch
```

In [11]:

```python
# printing the clubbed columns along with the lunch column
student_exams_df_clubbed[["lunch", "subject", "score"]]
```

Out[11]:

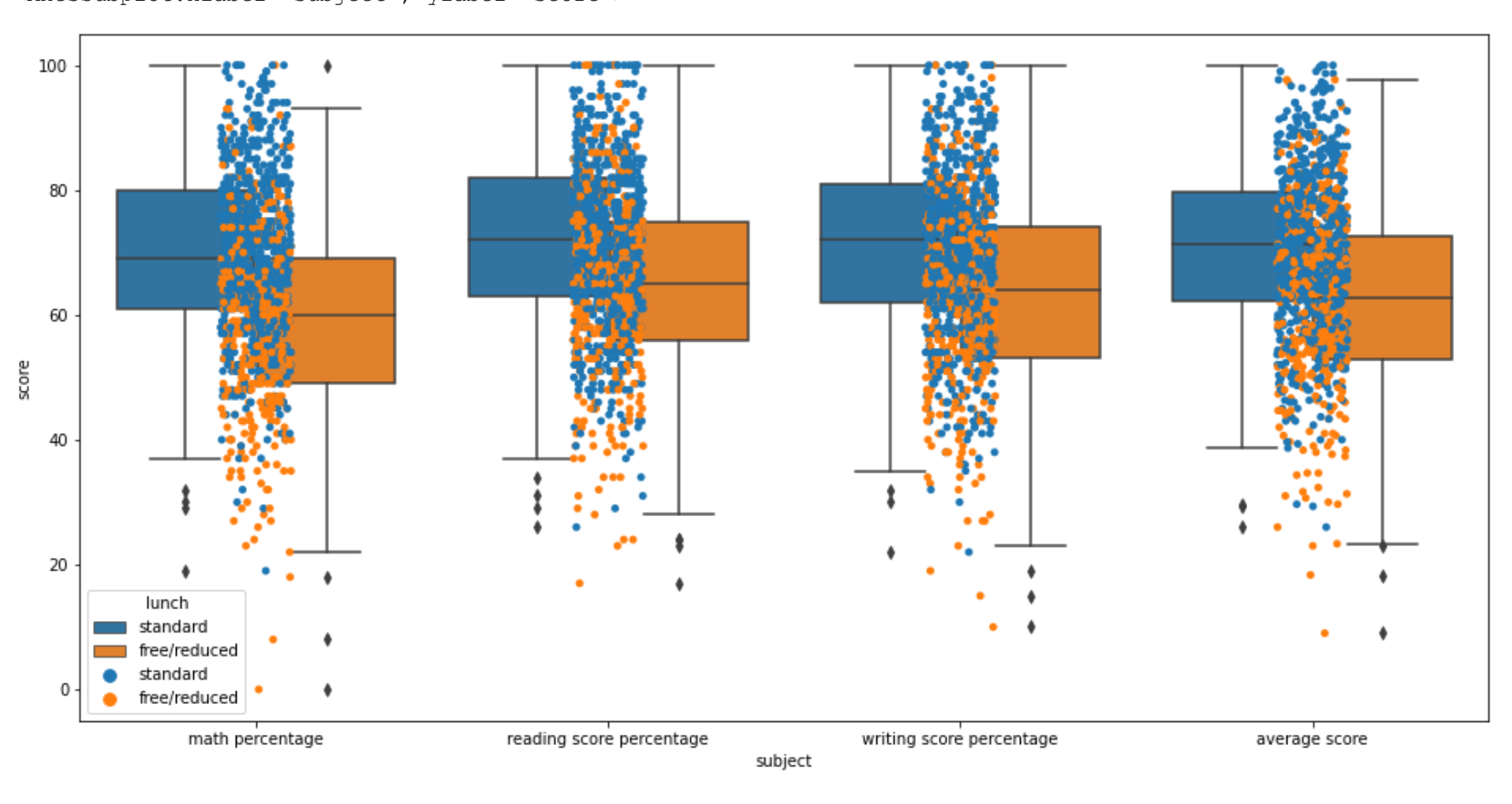
	lunch	subject	score
0	standard	math percentage	72.000000
1	standard	math percentage	69.000000
2	standard	math percentage	90.000000
3	free/reduced	math percentage	47.000000
4	standard	math percentage	76.000000
...	...	...	...
3995	standard	average score	94.000000
3996	free/reduced	average score	57.333333
3997	free/reduced	average score	65.000000
3998	standard	average score	74.333333
3999	free/reduced	average score	83.000000

4000 rows × 3 columns

In [12]:

```python
# using a boxplot and stripplot to check the correlation between lunch meals and student scores
plt.figure(figsize=(16,8))
sns.boxplot(x="subject", y="score", hue="lunch", data=student_exams_df_clubbed)
sns.stripplot(x="subject", y="score", hue="lunch", data=student_exams_df_clubbed)
```

Out[12]: <AxesSubplot:xlabel='subject', ylabel='score'>



In [13]:

```python
# printing the clubbed columns along with the parental level of education column
student_exams_df_clubbed[["parental level of education", "subject", "score"]]
```

Out[13]:

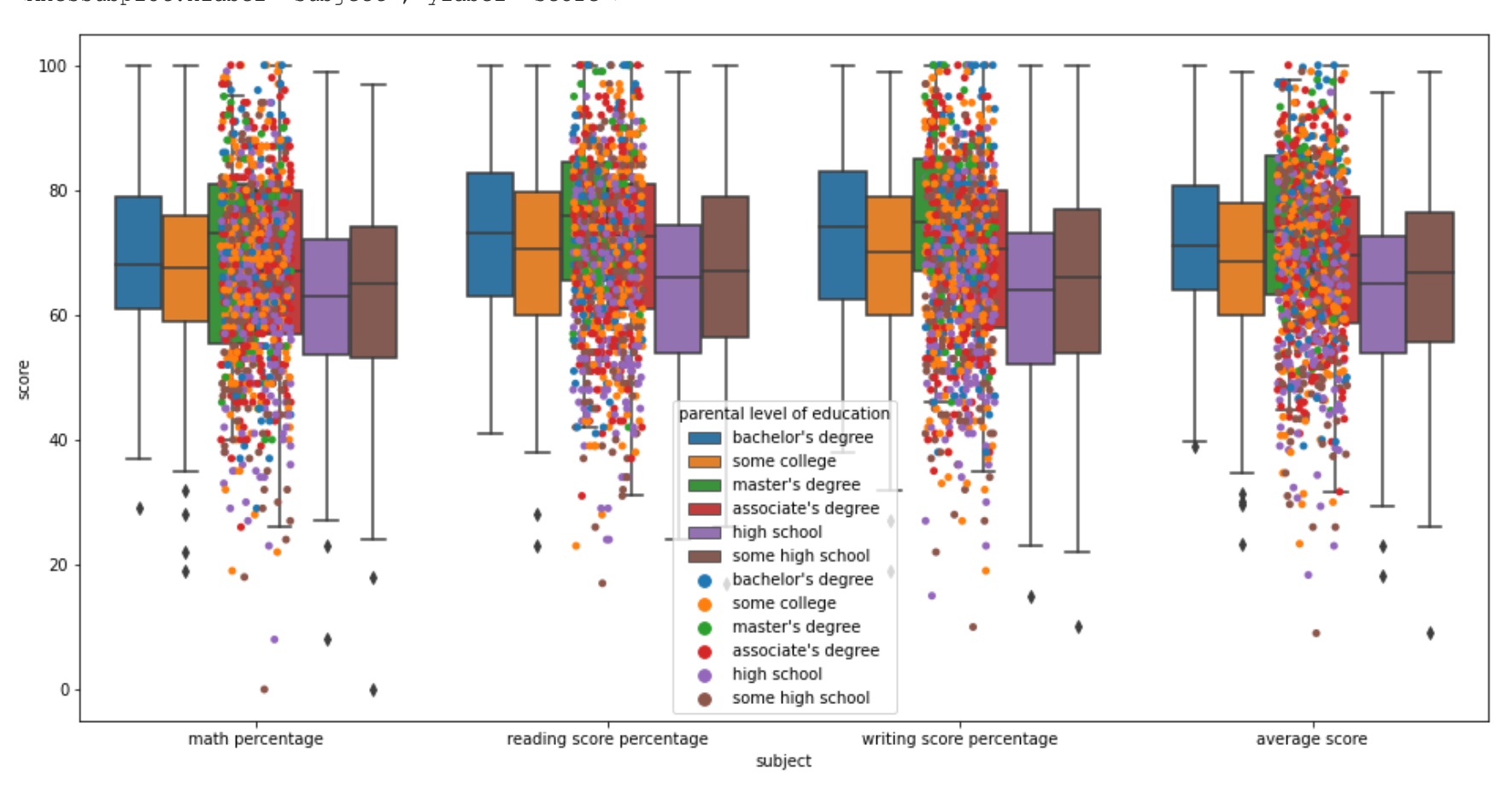
	parental level of education	subject	score
0	bachelor's degree	math percentage	72.000000
1	some college	math percentage	69.000000
2	master's degree	math percentage	90.000000
3	associate's degree	math percentage	47.000000
4	some college	math percentage	76.000000
...	...	...	...
3995	master's degree	average score	94.000000
3996	high school	average score	57.333333
3997	high school	average score	65.000000
3998	some college	average score	74.333333
3999	some college	average score	83.000000

4000 rows × 3 columns

In [14]:

```python
# using a boxplot and stripplot to check the correlation between parental level of education and student scores
plt.figure(figsize=(16,8))
sns.boxplot(data=student_exams_df_clubbed, x="subject", y="score", hue="parental level of education")
sns.stripplot(data=student_exams_df_clubbed, x="subject", y="score", hue="parental level of education")
```

Out[14]: <AxesSubplot:xlabel='subject', ylabel='score'>



In [15]:

```python
# printing the clubbed columns along with the test preparation course
student_exams_df_clubbed[["test preparation course", "subject", "score"]]
```

Out[15]:

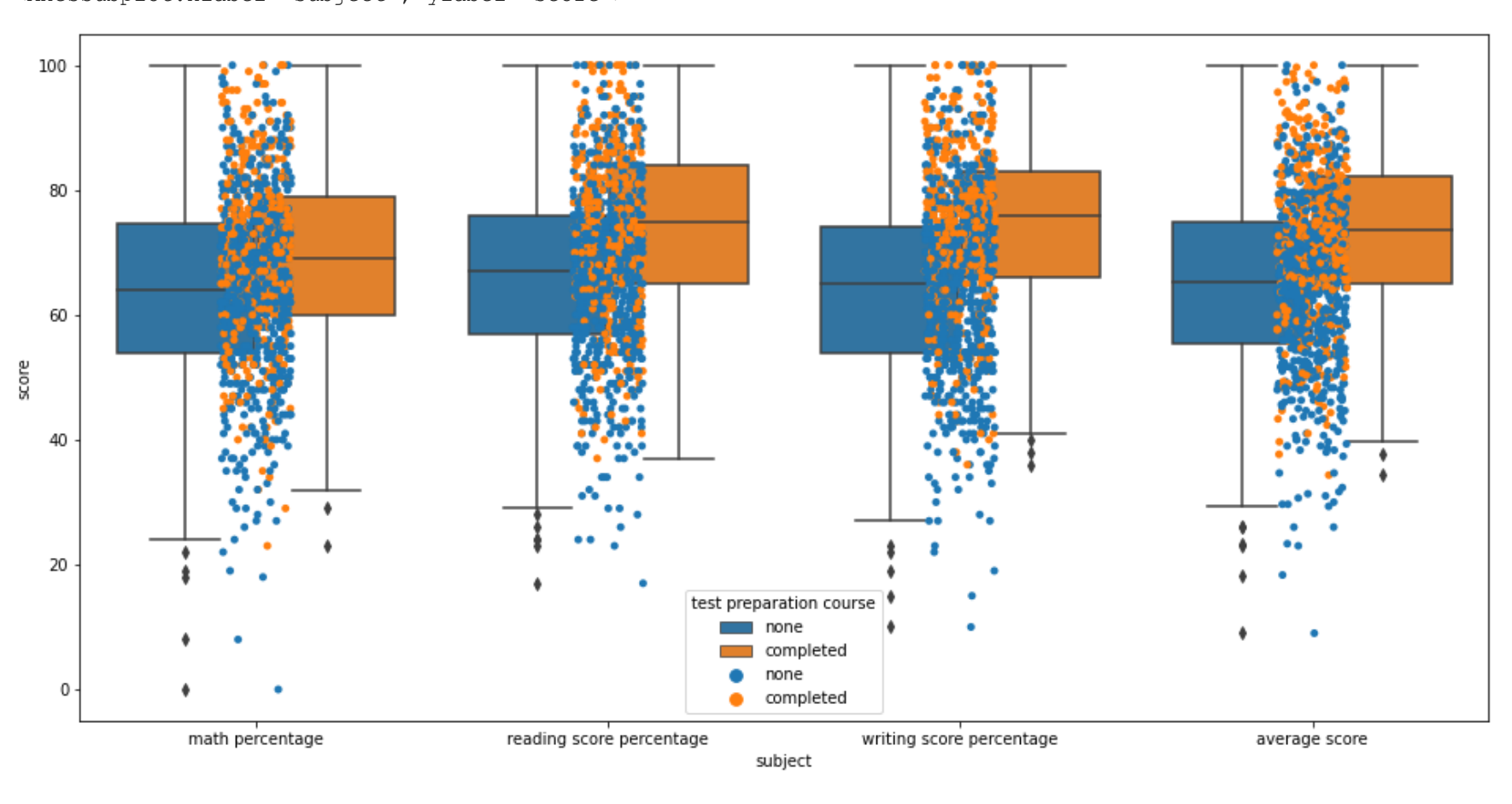
	test preparation course	subject	score
0	none	math percentage	72.000000
1	completed	math percentage	69.000000
2	none	math percentage	90.000000
3	none	math percentage	47.000000
4	none	math percentage	76.000000
...	...	...	...
3995	completed	average score	94.000000
3996	none	average score	57.333333
3997	completed	average score	65.000000
3998	completed	average score	74.333333
3999	none	average score	83.000000

4000 rows × 3 columns

In [16]:

```python
# using a boxplot and stripplot to check the correlation between test preparation course and student scores
plt.figure(figsize=(16,8))
sns.boxplot(data=student_exams_df_clubbed, x="subject", y="score", hue="test preparation course")
sns.stripplot(data=student_exams_df_clubbed, x="subject", y="score", hue="test preparation course")
```

Out[16]: <AxesSubplot:xlabel='subject', ylabel='score'>



In [17]:

```python
# printing the clubbed columns along with the race/ethnicity column
student_exams_df_clubbed[["race/ethnicity", "subject", "score"]]
```

Out[17]:

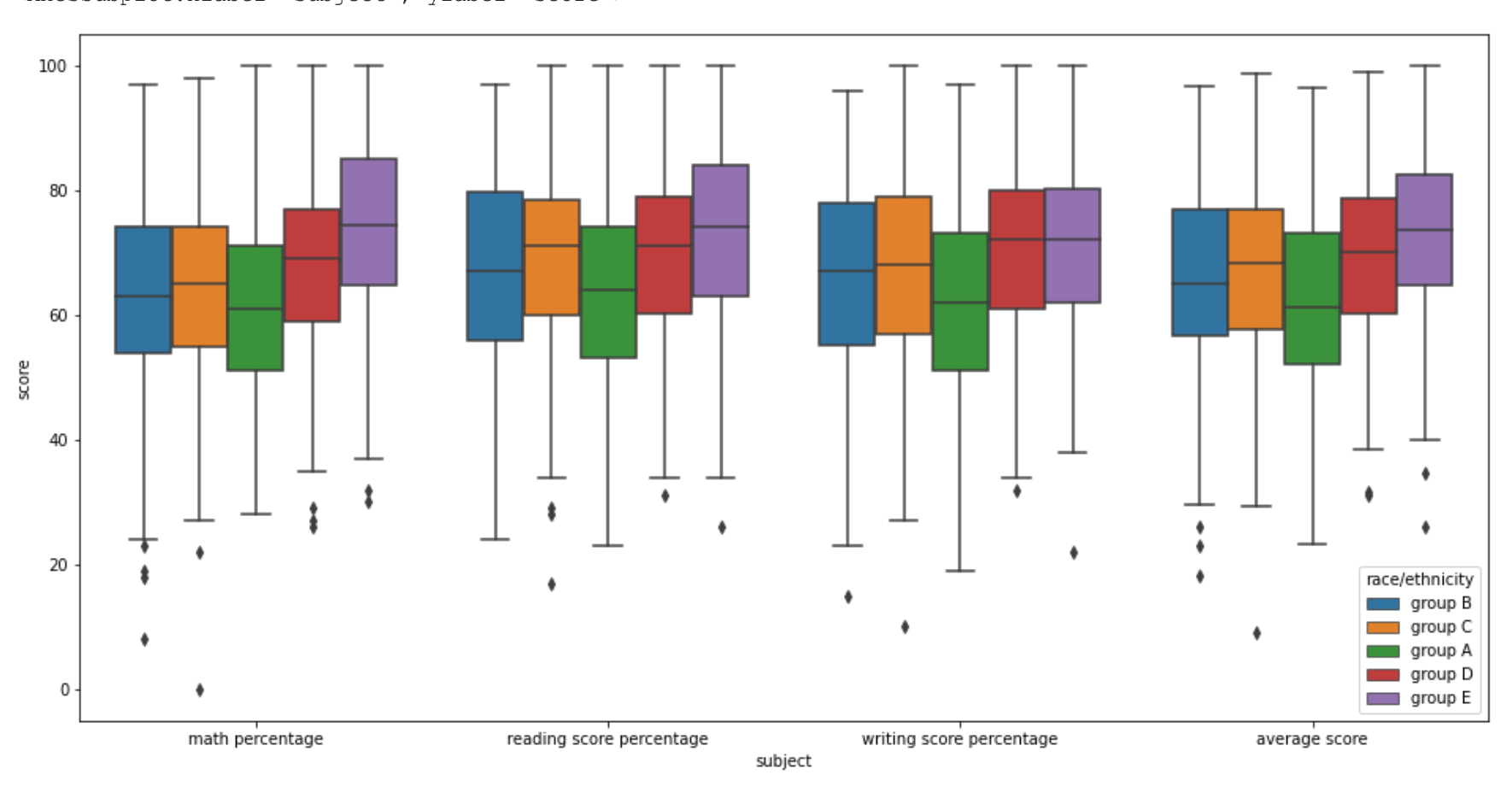
	race/ethnicity	subject	score
0	group B	math percentage	72.000000
1	group C	math percentage	69.000000
2	group B	math percentage	90.000000
3	group A	math percentage	47.000000
4	group C	math percentage	76.000000
...	...	...	...
3995	group E	average score	94.000000
3996	group C	average score	57.333333
3997	group C	average score	65.000000
3998	group D	average score	74.333333
3999	group D	average score	83.000000

4000 rows × 3 columns

In [18]:

```python
# using a boxplot and sripplot to check the correlation between race/ethnicity and student scores
plt.figure(figsize=(16,8))
sns.boxplot(data=student_exams_df_clubbed, x="subject", y="score", hue="race/ethnicity")
```

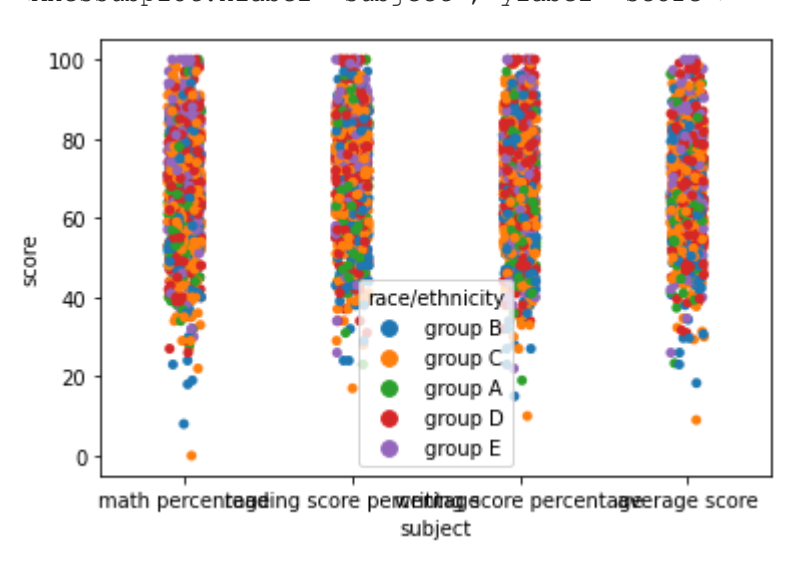
Out[18]: <AxesSubplot:xlabel='subject', ylabel='score'>



In [19]:

```python
sns.stripplot(data=student_exams_df_clubbed, x="subject", y="score", hue="race/ethnicity")
```

Out[19]: <AxesSubplot:xlabel='subject', ylabel='score'>



In [20]:

```python
# printing the clubbed columns along with the sex column
student_exams_df_clubbed[["sex", "subject", "score"]]
```

Out[20]:

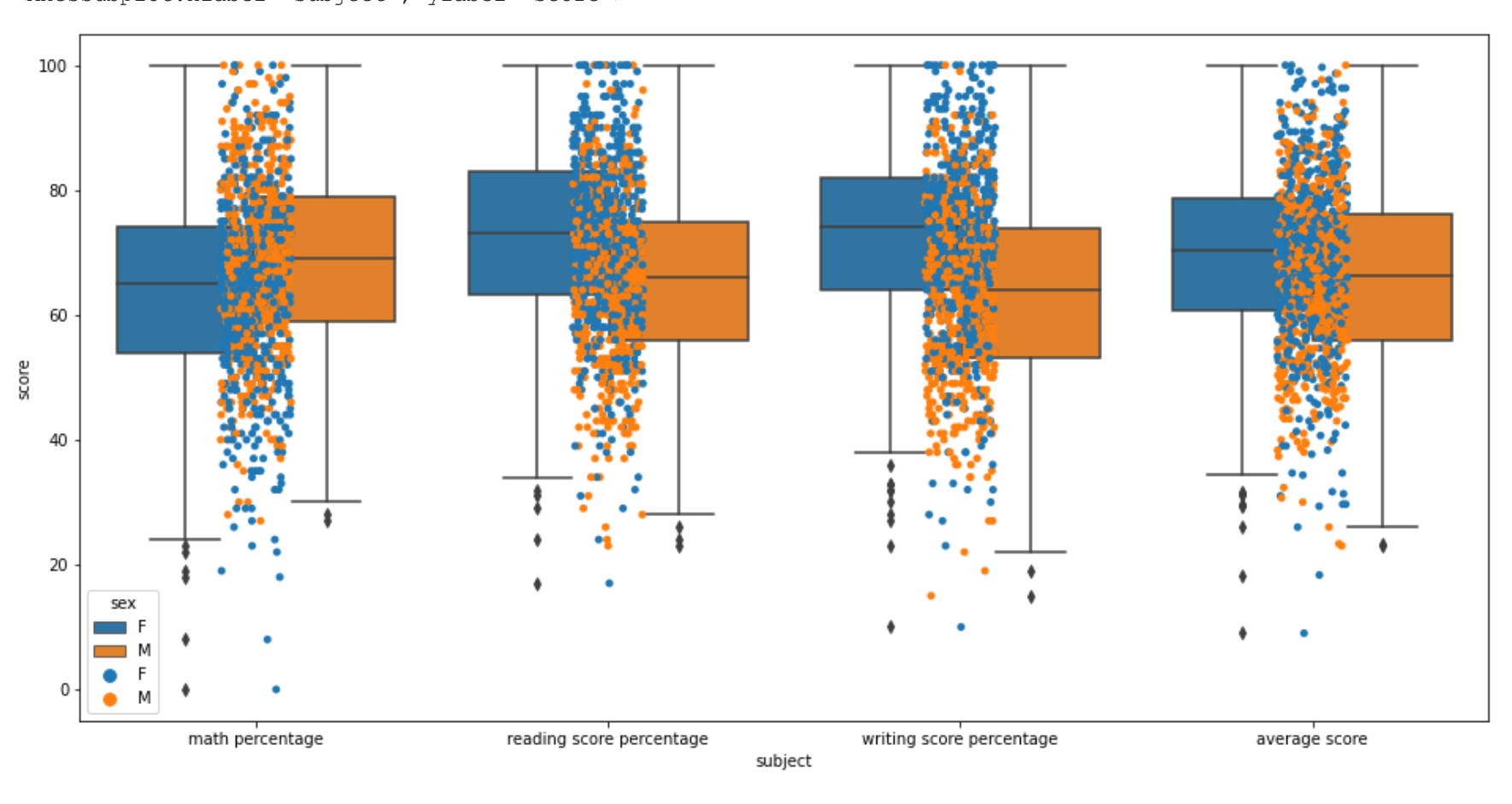
	sex	subject	score
0	F	math percentage	72.000000
1	F	math percentage	69.000000
2	F	math percentage	90.000000
3	M	math percentage	47.000000
4	M	math percentage	76.000000
...	...	...	...
3995	F	average score	94.000000
3996	M	average score	57.333333
3997	F	average score	65.000000
3998	F	average score	74.333333
3999	F	average score	83.000000

4000 rows × 3 columns

In [21]:

```python
# using a boxplot and stripplot to check the correlation between sex and student scores
plt.figure(figsize=(16,8))
sns.boxplot(data=student_exams_df_clubbed, x="subject", y="score", hue="sex")
sns.stripplot(data=student_exams_df_clubbed, x="subject", y="score", hue="sex")
```

Out[21]: <AxesSubplot:xlabel='subject', ylabel='score'>



In [22]:

```python
#printing the dataframe
student_exams_df.head()
```

Out[22]:

	race/ethnicity	parental level of education	lunch	sex	test preparation course	math percentage	reading score percentage	writing score percentage	average score
0	group B	bachelor's degree	standard	F	none	72.0	72.0	74.0	72.666667
1	group C	some college	standard	F	completed	69.0	90.0	88.0	82.333333
2	group B	master's degree	standard	F	none	90.0	95.0	93.0	92.666667
3	group A	associate's degree	free/reduced	M	none	47.0	57.0	44.0	49.333333
4	group C	some college	standard	M	none	76.0	78.0	75.0	76.333333

In [23]:

```python
# dropping the score columns to check feature importance as these columns arent categorical columns
student_exams_df_rf = student_exams_df.drop(['math percentage','reading score percentage','writing score percer
```

In [24]:

```python
# storing the average score values into a variable in order to use later and then dropping the average score co
y = student_exams_df_rf['average score']
student_exams_df_rf = student_exams_df_rf.drop('average score',axis=1)
```

In [25]:

```python
# changing all categorical variables into dummies in order to prosses using random forest
student_exams_df_rf = pd.get_dummies(student_exams_df_rf)
student_exams_df_rf
```

Out[25]:

	race/ethnicity_group A	race/ethnicity_group B	race/ethnicity_group C	race/ethnicity_group D	race/ethnicity_group E	parental level of education_associate's degree
0	0	1	0	0	0	0
1	0	0	1	0	0	0
2	0	1	0	0	0	0
3	1	0	0	0	0	1
4	0	0	1	0	0	0
...	...	...	...	...	...	...
995	0	0	0	0	1	0
996	0	0	1	0	0	0
997	0	0	1	0	0	0
998	0	0	0	1	0	0
999	0	0	0	1	0	0

1000 rows × 17 columns

In [26]:

```python
# creating a test and train set with a 30/70 split as the dataset is relatively small
x_train,y_train,x_test,y_test = train_test_split(student_exams_df_rf,y,test_size=0.3,random_state=42)
```

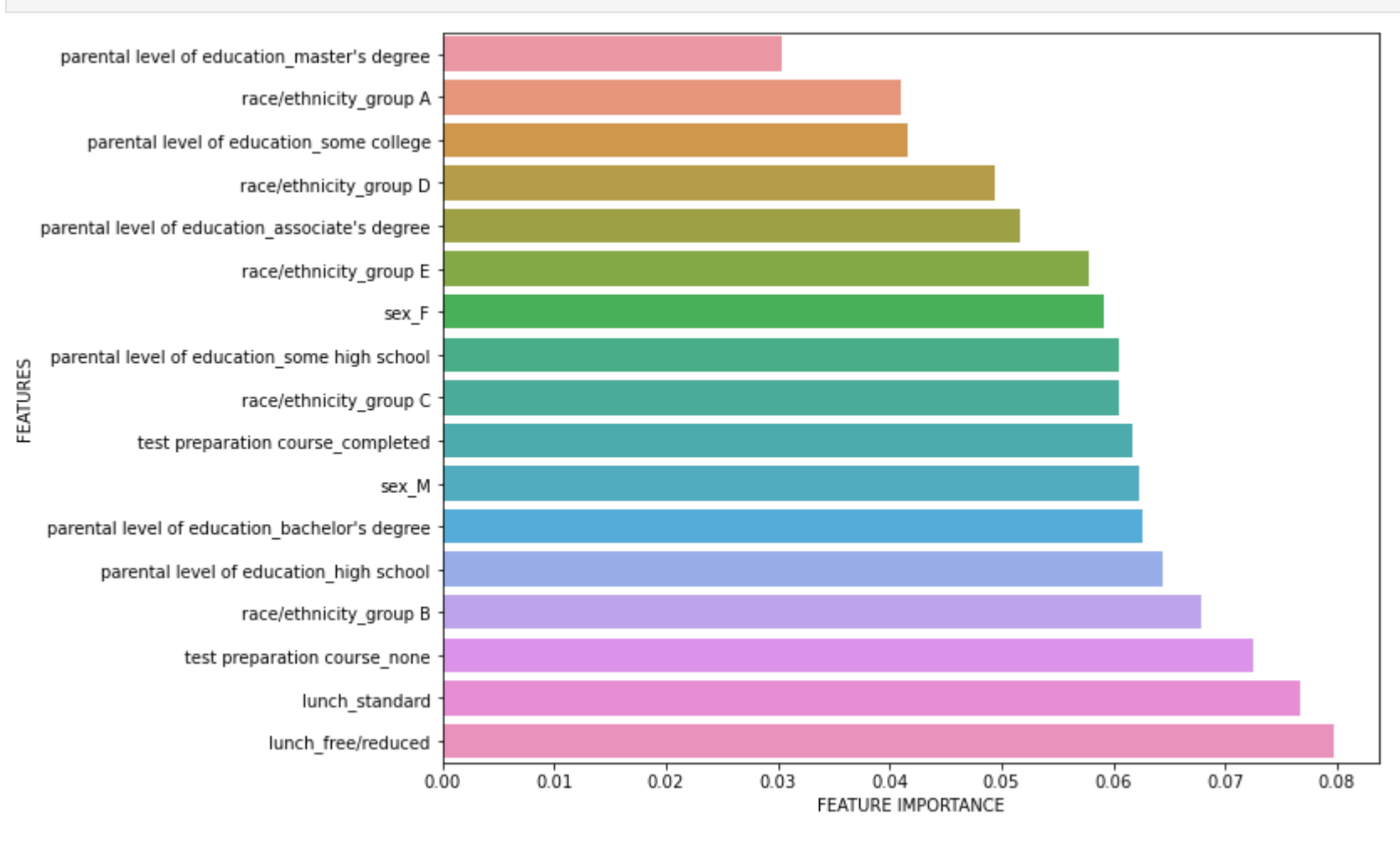
In [27]:

```python
# using a ridge regressor model as we have multicollinerity present in the dataframe
model = Ridge()
# fitting the model with a train and test sets
model.fit(x_train,x_test)
hyp = model.predict(y_train)
train_hyp = model.predict(x_train)
#MSE is performed to asses the average difference between observed and predicted values
score =  mean_squared_error(y_test,hyp,squared=False)
score
```

Out[27]: 13.27512125555686

In [28]:

```python
#make a random forest regressor model and use its feature importance to help in our classification task
model = RandomForestRegressor()
model.fit(x_train,x_test)
feature_importance = np.array(model.feature_importances_)
feature_names = np.array(x_train.columns)
data={'feature_names':feature_names,'feature_importance':feature_importance}
#plotting a barplot in order to visualize the difference in various feature importances
df_plt = pd.DataFrame(data)
df_plt.sort_values(by=['feature_importance'], ascending=True,inplace=True)
plt.figure(figsize=(10,8))
sns.barplot(x=df_plt['feature_importance'], y=df_plt['feature_names'])
#labeling the axis of the barplot
plt.xlabel('FEATURE IMPORTANCE')
plt.ylabel('FEATURES')
#printing the plot
plt.show()
```



In [ ]: