Predicting Test Scores

July 3, 2021

1 Predicting Test Scores

[1]:
```
# Load all necessary libraries.

import pandas as pd
import numpy as np
```

1.0.1 Part 1: Load and inspect the dataset.

[2]:
```
test_scores = pd.read_csv(r'/Users/andressotelo/Documents/Datasets/test_scores.
 ↪csv')
print(test_scores.shape)
test_scores.head()
```

```
(2133, 11)
```

[2]:
```
  school school_setting school_type classroom teaching_method  n_student  \
0  ANKYI          Urban  Non-public       6OL        Standard       20.0
1  ANKYI          Urban  Non-public       6OL        Standard       20.0
2  ANKYI          Urban  Non-public       6OL        Standard       20.0
3  ANKYI          Urban  Non-public       6OL        Standard       20.0
4  ANKYI          Urban  Non-public       6OL        Standard       20.0

  student_id  gender             lunch  pretest  posttest
0      2FHT3  Female  Does not qualify     62.0      72.0
1      3JIVH  Female  Does not qualify     66.0      79.0
2      3XOWE    Male  Does not qualify     64.0      76.0
3      556O0  Female  Does not qualify     61.0      77.0
4      74LOE    Male  Does not qualify     64.0      76.0
```

[3]:
```
test_scores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2133 entries, 0 to 2132
Data columns (total 11 columns):
school             2133 non-null object
school_setting     2133 non-null object
```

```
school_type        2133 non-null object
classroom          2133 non-null object
teaching_method    2133 non-null object
n_student          2133 non-null float64
student_id         2133 non-null object
gender             2133 non-null object
lunch              2133 non-null object
pretest            2133 non-null float64
posttest           2133 non-null float64
dtypes: float64(3), object(8)
memory usage: 183.4+ KB
```

[4]:

```
# Descriptive statistics of dataset.

test_scores.describe()
```

[4]:

```
         n_student      pretest     posttest
count  2133.000000  2133.000000  2133.000000
mean     22.796531    54.955931    67.102203
std       4.228893    13.563101    13.986789
min      14.000000    22.000000    32.000000
25%      20.000000    44.000000    56.000000
50%      22.000000    56.000000    68.000000
75%      27.000000    65.000000    77.000000
max      31.000000    93.000000   100.000000
```

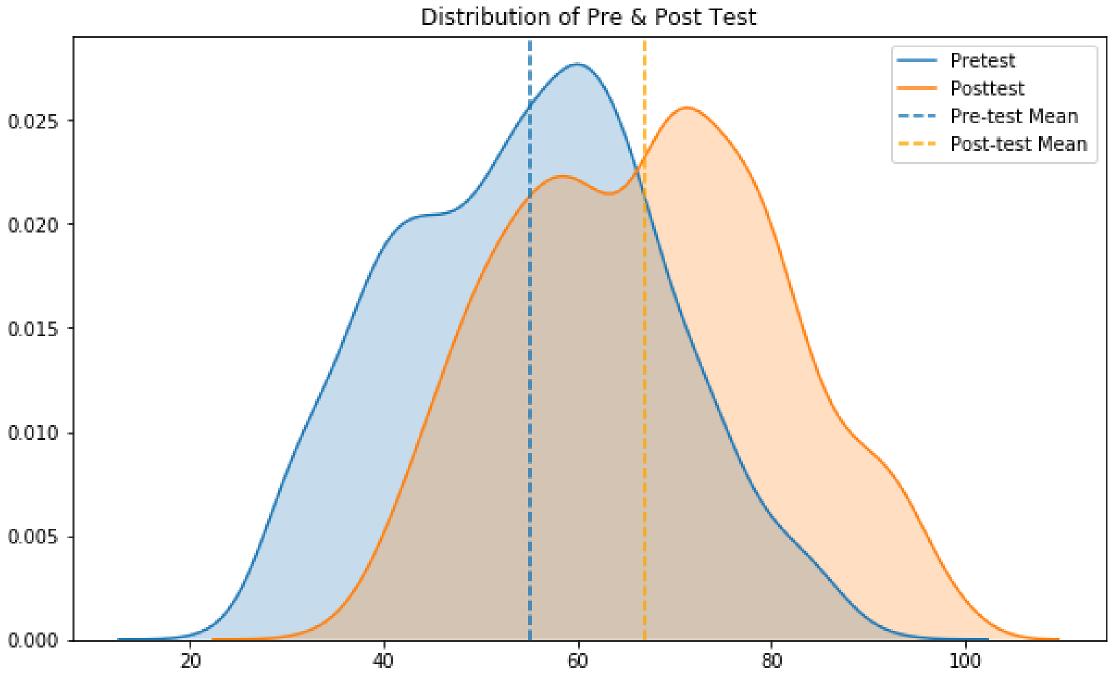
1.0.2 Part 2: Visualize the dataset.

[5]:

```
# Load the necessary libraries to visualize the data.

import matplotlib.pyplot as plt
import seaborn as sns
```

[6]:

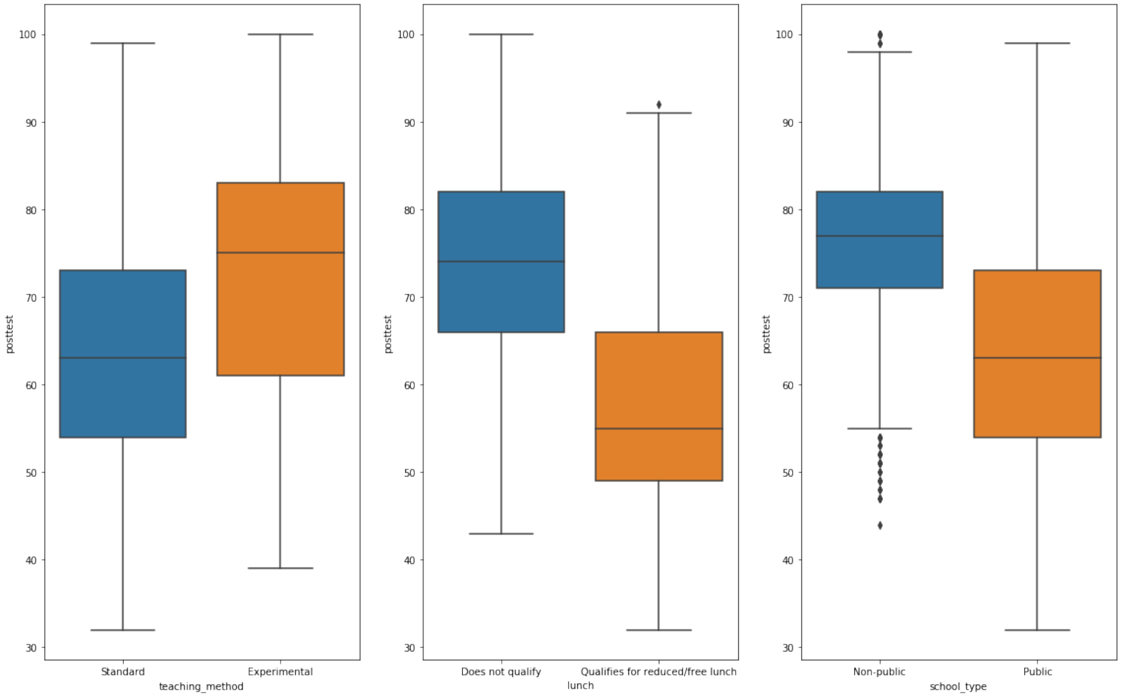
```
# Distribution of Pre and Post Test

plt.figure(figsize = (10, 6))
sns.kdeplot(data = test_scores['pretest'], shade = True, label = 'Pretest')
sns.kdeplot(data = test_scores['posttest'], shade = True, label = 'Posttest')
plt.title('Distribution of Pre & Post Test')
plt.axvline(x = 55, linestyle = '--', label = 'Pre-test Mean')
plt.axvline(x = 67, linestyle = '--', color = 'orange', label = 'Post-test␣
 ↪Mean')
plt.legend()
plt.show()
```



[7]:
```
pd.pivot_table(test_scores,
               values = 'pretest',
               index = ['school_type'],
               columns = ['gender'],
               aggfunc = np.mean)
```

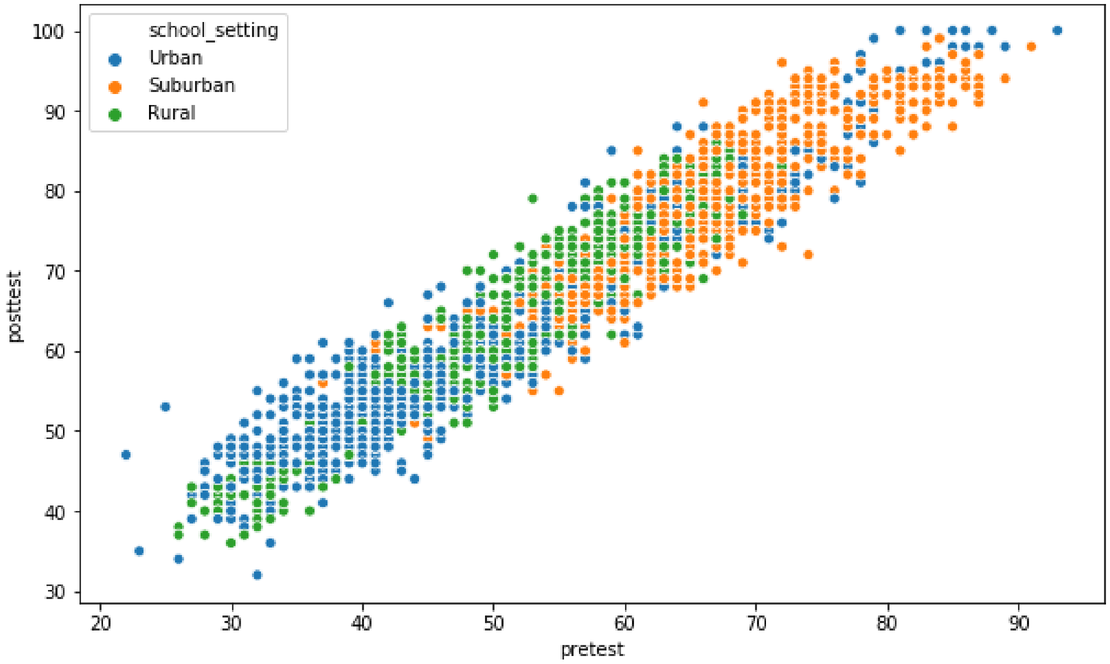
[7]:
```
gender          Female       Male
school_type
Non-public   63.263345  63.244444
Public       51.830968  52.291202
```

[8]:
```
f, axes = plt.subplots(1, 3, figsize = (16, 10))
sns.boxplot(data = test_scores, x = 'teaching_method', y = 'posttest', ax =␣
 ↪axes[0])
sns.boxplot(data = test_scores, x = 'lunch', y = 'posttest', ax = axes[1])
sns.boxplot(data = test_scores, x = 'school_type', y = 'posttest', ax = axes[2])
plt.tight_layout()
plt.show()
```



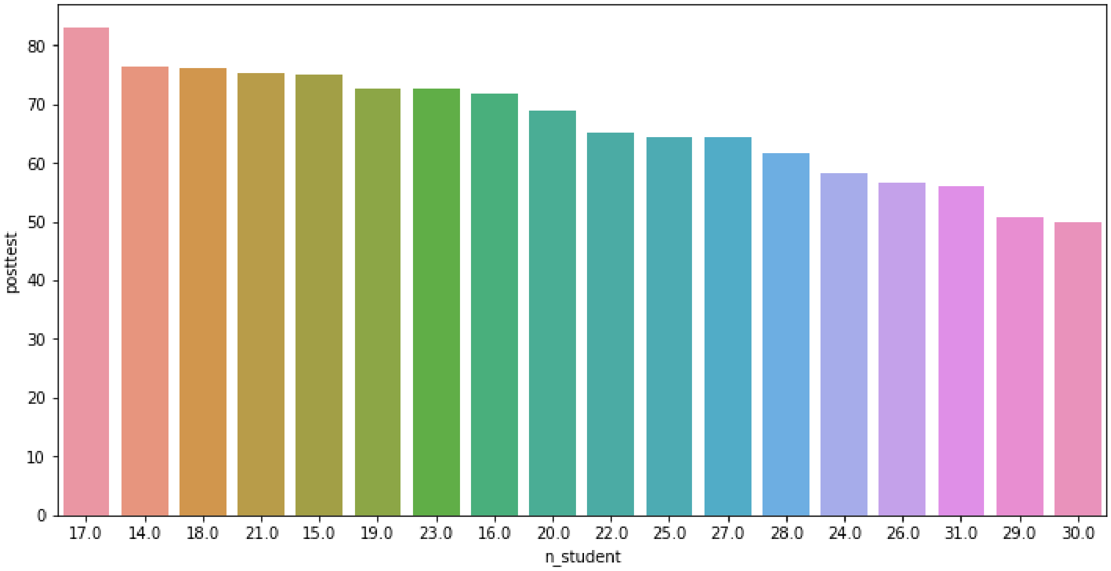
```
[9]: plt.figure(figsize = (10, 6))
     sns.scatterplot(data = test_scores, x = 'pretest', y = 'posttest', hue =␣
      ↪'school_setting')
```

```
[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8316fe8630>
```



[10]:

```
# Assemble a bar chart that shows the average test score of class based on its␣
↪class size.

class_size = test_scores.groupby(['n_student'])['posttest'].mean().reset_index()
class_size = pd.DataFrame(class_size)
plt.figure(figsize = (12, 6))
plt.xlabel('Class Size')
plt.ylabel('Average Test Score')
sns.barplot(x = 'n_student',
            y = 'posttest',
            data = class_size,
            order = class_size.sort_values('posttest', ascending =␣
↪False)['n_student'])
plt.show()
```



1.0.3 Part 3: Machine Learning

[11]:

```
# Import necessary libraries to implement ML tools.

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

[12]:
```
# Choose variables that may have an impact on predicting test score.

x = test_scores[['pretest', 'n_student', 'school_setting', 'school_type',␣
 ↪'teaching_method', 'lunch']]
y = test_scores[['posttest']]

x = pd.get_dummies(x)
x.head()
```

[12]:
```
   pretest  n_student  school_setting_Rural  school_setting_Suburban  \
0     62.0       20.0                    0                        0
1     66.0       20.0                    0                        0
2     64.0       20.0                    0                        0
3     61.0       20.0                    0                        0
4     64.0       20.0                    0                        0

   school_setting_Urban  school_type_Non-public  school_type_Public  \
0                     1                       1                   0
1                     1                       1                   0
2                     1                       1                   0
3                     1                       1                   0
4                     1                       1                   0

   teaching_method_Experimental  teaching_method_Standard  \
0                             0                         1
1                             0                         1
2                             0                         1
3                             0                         1
4                             0                         1

   lunch_Does not qualify  lunch_Qualifies for reduced/free lunch
0                       1                                       0
1                       1                                       0
2                       1                                       0
3                       1                                       0
4                       1                                       0
```

[30]:
```
# Multiple Linear Regression Model

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,␣
 ↪random_state = 42)
mlr = LinearRegression()
mlr.fit(x_train, y_train)
y_pred = mlr.predict(x_test)

r_square = r2_score(y_pred, y_test)
print(r_square)
print(round(mean_squared_error(y_test, y_pred), 2))
```

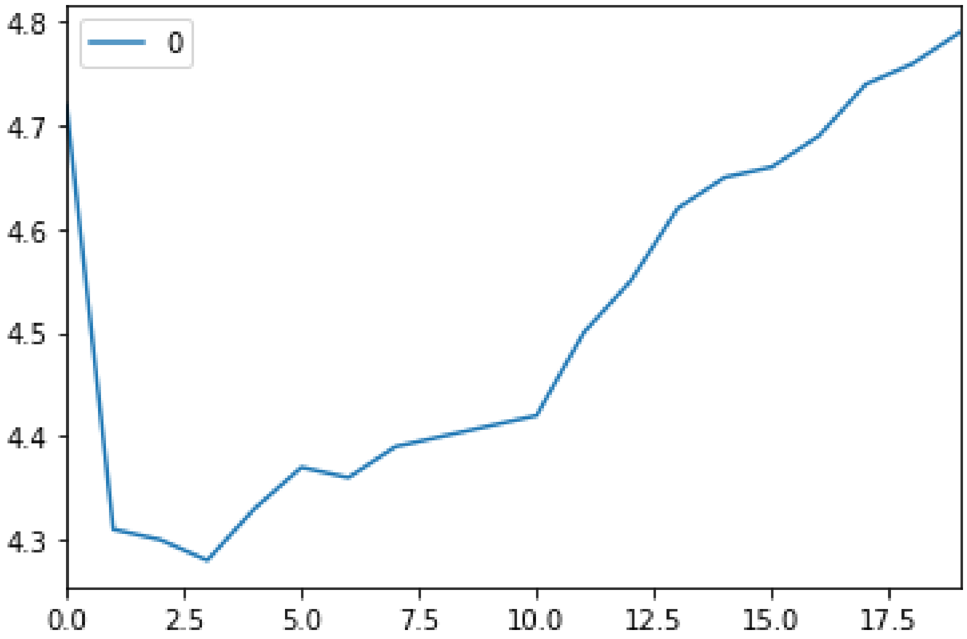
```
0.9443020680439963
10.18
```

[23]:

```
# KNN Regression
from sklearn import neighbors
from math import sqrt

# Preprocessing the data (normalize the dataset)
from sklearn import preprocessing
x_train_norm = preprocessing.normalize(x_train)
x_train_norm = pd.DataFrame(x_train_norm)
x_test_norm = preprocessing.normalize(x_test)
x_test_norm = pd.DataFrame(x_test_norm)

# Build a "for" loop to create RMSE values
rmse_val = []
for k in range(20):
    k = k + 1
    knn = neighbors.KNeighborsRegressor(n_neighbors = k)
    knn.fit(x_train_norm, y_train)
    pred = knn.predict(x_test_norm)
    error = round(sqrt(mean_squared_error(y_test, pred)), 2)
    rmse_val.append(error)
    print('RMSE Value for k = ', k, 'is:', error)

# Plot RMSE values
rmse_val = pd.DataFrame(rmse_val)
rmse_val.plot()
```

```
RMSE Value for k =  1 is: 4.72
RMSE Value for k =  2 is: 4.31
RMSE Value for k =  3 is: 4.3
RMSE Value for k =  4 is: 4.28
RMSE Value for k =  5 is: 4.33
RMSE Value for k =  6 is: 4.37
RMSE Value for k =  7 is: 4.36
RMSE Value for k =  8 is: 4.39
RMSE Value for k =  9 is: 4.4
RMSE Value for k =  10 is: 4.41
RMSE Value for k =  11 is: 4.42
RMSE Value for k =  12 is: 4.5
RMSE Value for k =  13 is: 4.55
RMSE Value for k =  14 is: 4.62
RMSE Value for k =  15 is: 4.65
RMSE Value for k =  16 is: 4.66
RMSE Value for k =  17 is: 4.69
RMSE Value for k =  18 is: 4.74
RMSE Value for k =  19 is: 4.76
RMSE Value for k =  20 is: 4.79
```

[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f83181d2a90>
```



[ ]: