Student Assessment Performance Analysis

June 30, 2023

1 Abstract

A group of students at a charter school recently completed a round of diagnostic testing, and would like to analyze the data and create a data-driven instruction plan to improve scores for the next assessment.

2 Variables

- **Student**
- **Class**: Class grouped by the name of the teacher
- **Period**: Class period
- **Raw Score**: The total number of correct questions answered by each student
- **% Correct**: Percentage of questions correct

3 Objective

The analysis will be guided by the following questions:

1. How many students were tested?
2. What was the overall average and median % Correct in the dataset?
3. What was average and median score of % Correct for each class?

[4]:
```
#import packages

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statistics as s
```

[5]:
```
# import csv
df = pd.read_csv(r"/Users/scipio/Downloads/Part 1 - School Report_ 10th Grade␣
↪ELA - Student Performance by Individual_ 10th Grade ELA.csv", index_col = 0)
```

[6]:
```
# Preview of df
df.head()
```

[6]:
```
                  Student   Teacher      Period  Raw Score % Validation  \
Student Index
```

```
1              Student 1  Reynolds  5ELAHR-309         12          63%
2              Student 2   Kennedy  5ELAHR-308         11          58%
3              Student 3    Toomey  5ELAHR-307         10          53%
4              Student 4    Toomey  5ELAHR-307         10          53%
5              Student 5    Toomey  5ELAHR-307         13          68%

               Question 1 Question 2 Question 3 Question 4 Question 5  …  \
Student Index                                                           …
1                       D          B          A          C          B  …
2                       C          B          A          A          B  …
3                       D          B          A          A          A  …
4                       D          C          A          C          A  …
5                       D          B          A          C          B  …

               Question 10 Question 11 Question 12 Question 13 Question 14  \
Student Index
1                        D           D           D           B           B
2                        C           D           C           B           D
3                        C           D           A           B           C
4                        C           D           C           B           C
5                        C           D           A           B           C

               Question 15 Question 16 Question 17 Question 18 Question 19
Student Index
1                        D           C           C           C           A
2                        B           B           B           C           A
3                        D           B           C           C           C
4                        A           A           C           C           B
5                      NaN           B           C           B           D

[5 rows x 24 columns]
```

[7]:
```
# Slicing df for relevant columns needed for the analysis
df_1 = df.loc[:,'Student':'% Validation']
```

[8]:
```
# Changing the column name
df_1 = df_1.rename(columns={'% Validation': '% Correct', 'Teacher': 'Class'})
```

[11]:
```
# Validating column change
df_1.columns
```

[11]:
```
Index(['Student', 'Class', 'Period', 'Raw Score', '% Correct'], dtype='object')
```

[12]:
```
# Changing '% Correct' column data type to float
df_1['% Correct'] = df_1['% Correct'].str.replace('%','')

df_1['% Correct'] = df_1['% Correct'].str.strip()
```

```
df_1['% Correct'] = df_1['% Correct'].astype(float)
```

[13]:
```
# Converting data in '% Correct' in a decimals
df_1['% Correct'] = df_1['% Correct'] * .01
```

[14]:
```
# Schematic overview of data
df_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 68 entries, 1 to 68
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Student    68 non-null     object
 1   Class      68 non-null     object
 2   Period     68 non-null     object
 3   Raw Score  68 non-null     int64
 4   % Correct  68 non-null     float64
dtypes: float64(1), int64(1), object(3)
memory usage: 3.2+ KB
```

[15]:
```
# Shape of data
df_1.shape
```

[15]:
```
(68, 5)
```

There are 68 rows and 5 columns in the dataset.

[16]:
```
# null values
df_1.isna().sum()
```

[16]:
```
Student      0
Class        0
Period       0
Raw Score    0
% Correct    0
dtype: int64
```

There are no null values in the dataset.

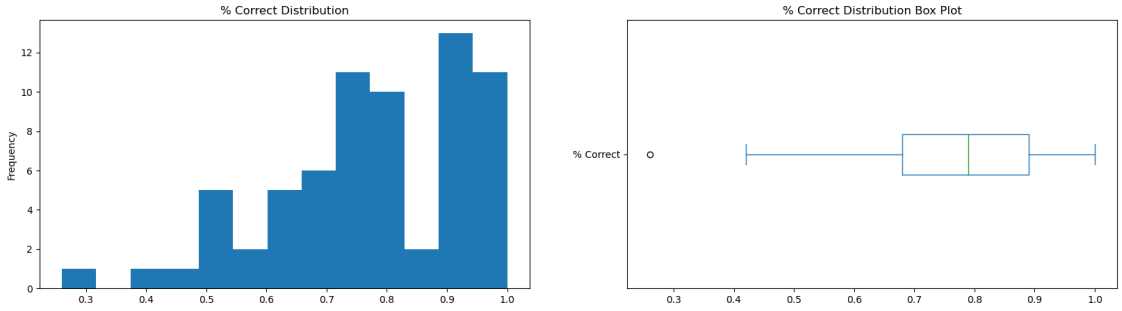
[109]:
```
# Statistical Overview of the dataset
round(df_1['% Correct'].describe(),2)
```

[109]:
```
count    68.00
mean      0.77
std       0.15
min       0.26
25%       0.68
```

```
50%       0.79
75%       0.89
max       1.00
Name: % Correct, dtype: float64
```

[18]:
```
# Histogram and boxplot of overall % Correct data
fig,axs = plt.subplots(nrows = 1, ncols = 2, figsize = (20,5))

df_1['% Correct'].plot(kind = 'hist', bins = 13, title = '% Correct␣
 ↪Distribution', ax = axs[0])
df_1['% Correct'].plot(kind = 'box', title = '% Correct Distribution Box Plot',␣
 ↪ax = axs[1], vert = False)
```

[18]:
```
<Axes: title={'center': '% Correct Distribution Box Plot'}>
```



3.1 Analysis

1. **How many students were tested?**

There were 68 students tested in total.

2. **What was the overall average and median % Correct in the dataset?**

The overall average of *'% Correct'* in the dataset was 77%. The overall median of *'% Correct'* was 79%. There is a multimodal distribution in the histogram based on the values of the '% Correct'. Based on the boxplot of the same values there is a negative skew in the data.

[130]:
```
# Mean % Correct of each class
class_mean = round(df_1.groupby('Class')['% Correct'].mean(),2) * 100

print(class_mean)
```

```
Class
Kennedy     80.0
Reynolds    80.0
Toomey      69.0
Name: % Correct, dtype: float64
```

[125]:
```
# Median '% Correct' of each class
class_median = round(df_1.groupby('Class')['% Correct'].median(),2) * 100

print(class_median)
```

```
Class
Kennedy     79.0
Reynolds    84.0
Toomey      74.0
Name: % Correct, dtype: float64
```

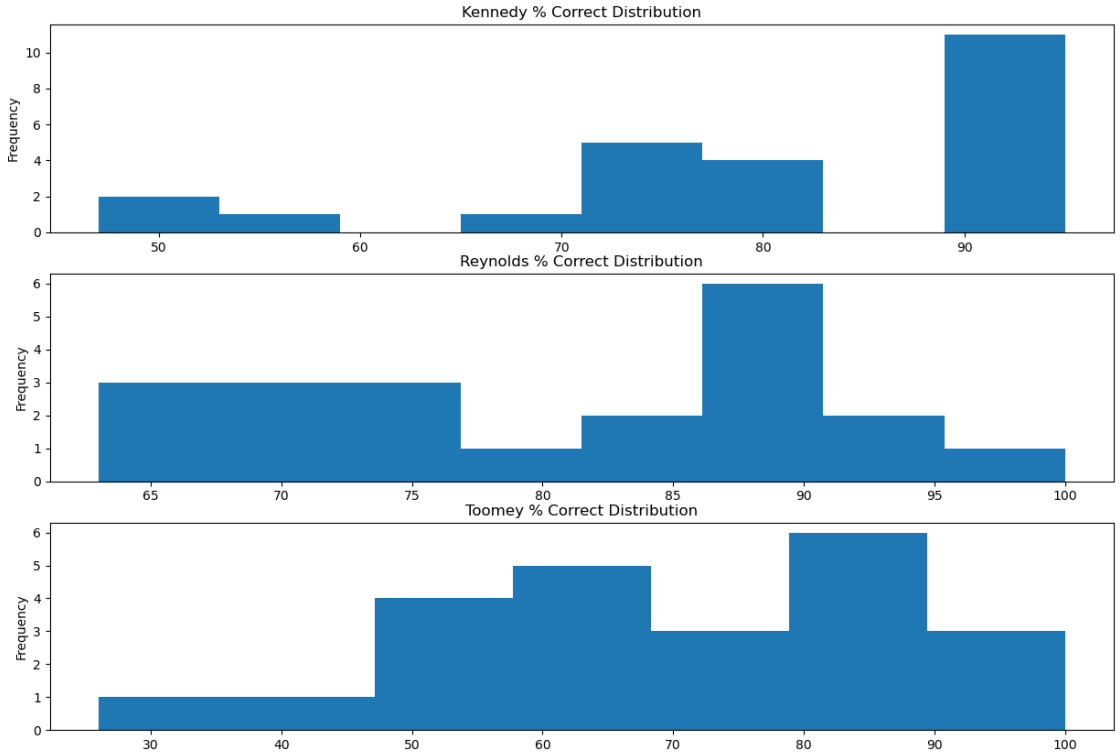
3.2 Analysis

3. **What was average and median score of % Correct for each class?**

Kennedy and Reynolds' classes have the same '% Correct' mean, 80%. Toomey's class has the lowest mean of '% Correct', 69%. Kennedy's class has a median *'% Correct* of 79%, Reynolds' class has a median* '% Correct* of 84%, and Toomey's class has a median *'% Correct'* of 74%.

[184]:
```
fig, axs = plt.subplots(nrows = 3, ncols = 1, figsize = (15,10))

df_1.query('Class == "Kennedy"')['% Correct'].plot(kind = 'hist', bins = 8,␣
 ↪title = 'Kennedy % Correct Distribution', ax = axs[0])
df_1.query('Class == "Reynolds"')['% Correct'].plot(kind = 'hist', bins = 8,␣
 ↪title = 'Reynolds % Correct Distribution', ax = axs [1])
df_1.query('Class == "Toomey"')['% Correct'].plot(kind = 'hist', bins = 7,␣
 ↪title = 'Toomey % Correct Distribution', ax = axs [2])
```

[184]: <Axes: title={'center': 'Toomey % Correct Distribution'}, ylabel='Frequency'>



All histograms representing the '% Correct' in each class have multimodal distributions and a left skew in the data.

3.3 Recommendations

Recommendations will be based on *Tiers* created based on students' *% Correct*. The Tier conditions are the following:

- % Correct <= 69 : Tier 3
- % Correct >= 79 AND <=89: Tier 2
- % Correct >= 90" Tier 1

[177]:
```
# Applying Tier conditions to the whole dataset
df_1['Tier'] = df_1['% Correct'].apply(lambda x: 'Tier 3' if x <= 69 else␣
 ↪('Tier 2' if  x <= 89 else  'Tier 1'))

df_1.head()
```

[177]:

Student Index	Student	Class	Period	Raw Score	% Correct	Tier
1	Student 1	Reynolds	5ELAHR-309	12	63.0	Tier 3
2	Student 2	Kennedy	5ELAHR-308	11	58.0	Tier 3
3	Student 3	Toomey	5ELAHR-307	10	53.0	Tier 3
4	Student 4	Toomey	5ELAHR-307	10	53.0	Tier 3

```
5              Student 5    Toomey  5ELAHR-307         13       68.0  Tier 3
```

[178]:
```
# Tier Count for all students
df_1['Tier'].value_counts()
```

[178]:
```
Tier 1    24
Tier 2    23
Tier 3    21
Name: Tier, dtype: int64
```

[176]:
```
# Tier Percentage for all students
round(df_1['Tier'].value_counts(normalize = True),2) * 100
```

[176]:
```
Tier 1    35.0
Tier 2    34.0
Tier 3    31.0
Name: Tier, dtype: float64
```

[134]:
```
# Creating a df for Kennedy's Class
Ken_Class = df_1.query('Class == "Kennedy"')['% Correct']

# Putting Kennedy's Class Results in a df
Ken_Class_df = pd.DataFrame (Ken_Class)

# Converting floats to integers
Ken_Class_df['% Correct'] = Ken_Class_df['% Correct'] * 100

#Creating a new column 'Tiers' based on '% Correct'
Ken_Class_df['Tier'] = Ken_Class_df['% Correct'].apply(lambda x: 'Tier 3' if x␣
 ↪<= 69 else ('Tier 2' if  x <= 89 else  'Tier 1'))
```

[79]:
```
# Tier Count Kennedy's Class
Ken_Class_df['Tier'].value_counts()
```

[79]:
```
Tier 2    15
Tier 1     5
Tier 3     4
Name: Tier, dtype: int64
```

[135]:
```
# Tier Percentage Kennedy's Class
round(Ken_Class_df['Tier'].value_counts(normalize = True),2) * 100
```

[135]:
```
Tier 2    62.0
Tier 1    21.0
Tier 3    17.0
Name: Tier, dtype: float64
```

[137]:
```
# Creating a df for Reynolds' Class
Rey_Class = df_1.query('Class == "Reynolds"')['% Correct']

# Putting Reynold's Class Results in a df
Rey_Class_df = pd.DataFrame (Rey_Class)

# Converting floats to integers
Rey_Class_df['% Correct'] = Rey_Class_df['% Correct'] * 100

#Creating a new column 'Tiers' based on '% Correct'
Rey_Class_df['Tier'] = Rey_Class_df['% Correct'].apply(lambda x: 'Tier 3' if x␣
 ↪<= 69 else ('Tier 2' if  x <= 89 else  'Tier 1'))
```

[90]:
```
# Tier Count Reynolds' Class
Rey_Class_df['Tier'].value_counts()
```

[90]:
```
Tier 2    12
Tier 3     6
Tier 1     3
Name: Tier, dtype: int64
```

[138]:
```
# Tier Percentage Reynolds' Class
round(Rey_Class_df['Tier'].value_counts(normalize = True),2) * 100
```

[138]:
```
Tier 2    57.0
Tier 3    29.0
Tier 1    14.0
Name: Tier, dtype: float64
```

[141]:
```
# Creating a df for Toomey's Class
Tom_Class = df_1.query('Class == "Toomey"')['% Correct']

# Putting Kennedy Class Results in a df
Tom_Class_df = pd.DataFrame (Tom_Class)

# Converting floats to integers
Tom_Class_df['% Correct'] = Tom_Class_df['% Correct'] * 100

#Creating a new column 'Tiers' based on '% Correct'
Tom_Class_df['Tier'] = Tom_Class_df['% Correct'].apply(lambda x: 'Tier 3' if x␣
 ↪<= 69 else ('Tier 2' if  x <= 89 else  'Tier 1'))
```

[42]:
```
# Tier Count Toomey's Class
Tom_Class_df['Tier'].value_counts()
```

[42]:
```
Tier 3    11
Tier 2     9
Tier 1     3
```

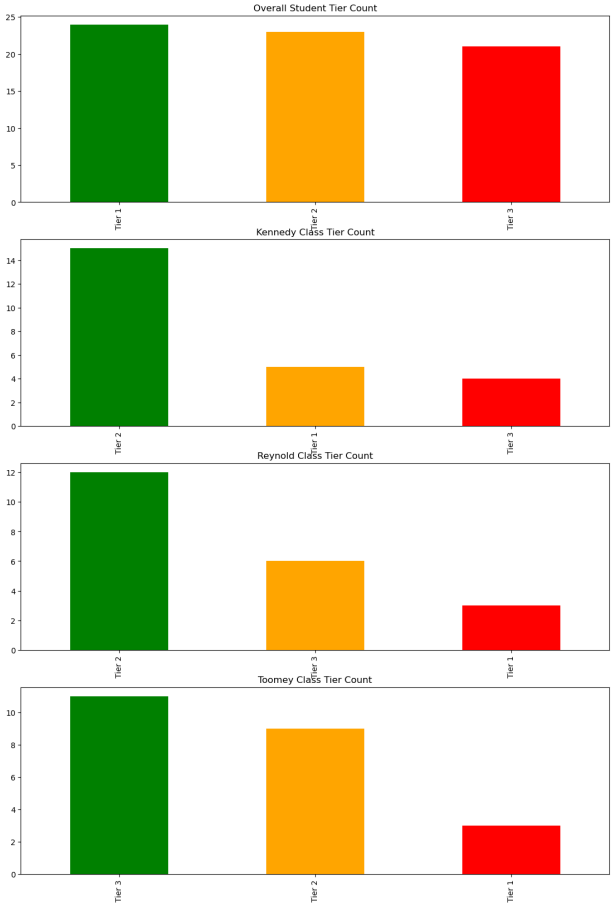
```
Name: Tier, dtype: int64
```

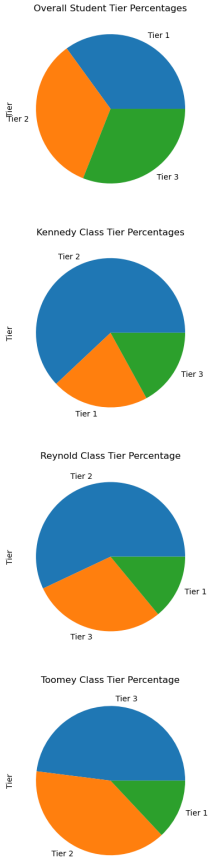
[44]:
```
# Tier Percentage Toomey's Class
round(Tom_Class_df['Tier'].value_counts(normalize = True),2) * 100
```

[44]:
```
Tier 3    48.0
Tier 2    39.0
Tier 1    13.0
Name: Tier, dtype: float64
```

[183]:
```
# Subplot to visualize metrics for each class
fig, axs = plt.subplots(nrows = 4, ncols = 2, figsize = (30,20))

# Bar and pie chart for all students
df_1['Tier'].value_counts().plot(kind = 'bar', ax = axs[0,0], color = ['green',␣
 ↪'orange', 'red'], title = 'Overall Student Tier Count')
(round(df_1['Tier'].value_counts(normalize = True),2) * 100).plot(kind = 'pie',␣
 ↪ax = axs[0,1], title = 'Overall Student Tier Percentages')

# Bar and pie chart for Kennedy's class
Ken_Class_df['Tier'].value_counts().plot(kind = 'bar', ax = axs[1,0], color =␣
 ↪['green', 'orange', 'red'], title = 'Kennedy Class Tier Count',)
(round(Ken_Class_df['Tier'].value_counts(normalize = True),2) * 100).plot(kind␣
 ↪= 'pie', ax = axs[1,1], title = 'Kennedy Class Tier Percentages')

# Bar and pie chart for Reynolds' class
Rey_Class_df['Tier'].value_counts().plot(kind = 'bar', ax = axs[2,0], color =␣
 ↪['green', 'orange', 'red'], title = 'Reynold Class Tier Count')
(round(Rey_Class_df['Tier'].value_counts(normalize = True),2) * 100).plot(kind␣
 ↪= 'pie', ax = axs[2,1], title = 'Reynold Class Tier Percentage')

# Bar and pie chart for Toomey's class
Tom_Class_df['Tier'].value_counts().plot(kind = 'bar', ax = axs[3,0], title =␣
 ↪'Toomey Class Tier Count', color = ['green','orange','red'])
(round(Tom_Class_df['Tier'].value_counts(normalize = True),2) * 100).plot(kind␣
 ↪= 'pie', ax = axs [3,1],title = 'Toomey Class Tier Percentage')
```

[183]:
```
<Axes: title={'center': 'Toomey Class Tier Percentage'}, ylabel='Tier'>
```





3.4 Recommendations

- Overall there is an even distribution of students in the three tiers. However, I recommend focusing on moving students into the next Tier for the next assessment, e.g. Tier 3 students into Tier 2 and Tier 2 studens into Tier 1.
- There are 15 students, 62%, in Kennedy's class that are in Tier 2. The instructur should focus on moving those students into Tier 1 for the next assessment
- There are 12 students, 57%, in Reynolds' class that are in Tier 2. The instructor should focus on moving students in Tier 2 to Tier 1 for the next assessment.
- There are 11 students, 48%, in that are in Tier 2 and 9 students, 39%, that are in Tier 3 in Toomey's class . The instructor should focus on moving students in Tier 3 to Tier 2 and students in Tier 2 to Tier 1 for the next assessment.