Data Narrative – Assignment 3

Kaveri Visavadiya (22110114)
Electrical Engineering
IIT Gandhinagar

I. Overview of the dataset

The given dataset is a collection of 8 files containing match statistics for both men and women at 4 major tennis tournaments of 2013. Each file has 42 columns and a minimum of 76 rows. Here, all null values are indicated by NaN. The attribute information of the dataset is as follows:

Player 1 Name of Player 1
Player 2 Name of Player 2
Result Result of the match (0/1) - Referenced on Player 1 is Result = 1 if Player 1 wins (FNL.1>FNL.2)
FSP.1 First Serve Percentage for player 1 (Real Number)
FSW.1 First Serve Won by player 1 (Real Number)
SSP.1 Second Serve Percentage for player 1 (Real Number)
SSW.1 Second Serve Won by player 1 (Real Number)
ACE.1 Aces won by player 1 (Numeric-Integer)
DBF.1 Double Faults committed by player 1 (Numeric-Integer)
WNR.1 Winners earned by player 1 (Numeric)
UFE.1 Unforced Errors committed by player 1 (Numeric)
BPC.1 Break Points Created by player 1 (Numeric)
BPW.1 Break Points Won by player 1 (Numeric)
NPA.1 Net Points Attempted by player 1 (Numeric)
NPW.1 Net Points Won by player 1 (Numeric)
TPW.1 Total Points Won by player 1 (Numeric)
ST1.1 Set 1 result for Player 1 (Numeric-Integer)
ST2.1 Set 2 Result for Player 1 (Numeric-Integer)
ST3.1 Set 3 Result for Player 1 (Numeric-Integer)
ST4.1 Set 4 Result for Player 1 (Numeric-Integer)
ST5.1 Set 5 Result for Player 1 (Numeric-Integer)
FNL.1 Final Number of Games Won by Player 1 (Numeric-Integer)
FSP.2 First Serve Percentage for player 2 (Real Number)
FSW.2 First Serve Won by player 2 (Real Number)
SSP.2 Second Serve Percentage for player 2 (Real Number)
SSW.2 Second Serve Won by player 2 (Real Number)
ACE.2 Aces won by player 2 (Numeric-Integer)
DBF.2 Double Faults committed by player 2 (Numeric-Integer)
WNR.2 Winners earned by player 2 (Numeric)
UFE.2 Unforced Errors committed by player 2 (Numeric)
BPC.2 Break Points Created by player 2 (Numeric)
BPW.2 Break Points Won by player 2 (Numeric)
NPA.2 Net Points Attempted by player 2 (Numeric)
NPW.2 Net Points Won by player 2 (Numeric)
TPW.2 Total Points Won by player 2 (Numeric)

ST1.2 Set 1 result for Player 2 (Numeric-Integer)
ST2.2 Set 2 Result for Player 2 (Numeric-Integer)
ST3.2 Set 3 Result for Player 2 (Numeric-Integer)
ST4.2 Set 4 Result for Player 2 (Numeric-Integer)
ST5.2 Set 5 Result for Player 2 (Numeric-Integer)
FNL.2 Final Number of Games Won by Player 2 (Numeric-Integer)
Round Round of the tournament at which game is played (Numeric-Integer)

II. Scientific questions/Hypotheses

1. Does the ratio of first serve won (FSW) to second serve won (SSW) affect the outcome of a match?
2. Is there a correlation between the number of breakpoints created (BPC) and the final number of games won (BPC) by a player in a round?
3. Is there a correlation between the total points won and the first serve percentage of the winning and losing players?
4. Assuming that higher first serve percentage (FSP) can contribute to higher chances of winning the match, is there a significant difference in FSP for winners and losers?
5. Is there a significant relationship between a player's first serve percentage (FSP) and their final number of games (FNL)?
6. Is there a correlation between the number of aces (ACE) served by a player and their probability of winning the match (Result)?
7. With what accuracy can we predict the outcome of any match based on the match statistics?
8. Compare the match statistics (ACE, DBF, WNR, UFE, BPW, TPW, FNL, Result) of the winner and loser in the final round (Round 7).

III. Details of the Libraries and Functions

The libraries matplotlib, pandas, seaborn and scikit-learn are required for answering the above posed scientific questions.

A. *Matplotlib*

Matplotlib is a scientific visualization library in Python for 2D plots of arrays. It allows us visual access to huge amounts of data in the form of plots such as line, bar, scatter, histogram, etc. Here we will be using the pyplot module which provides a MATLAB-like interface. Each pyplot function makes some change to a figure, i.e., creates a figure, plots some lines in the plotting area, inputs label, etc. It is imported as follows:

```
import matplotlib.pyplot as plt
```

- In the code, we use the functions title(), xlabel() and ylabel().
- The title() method in matplotlib module is used to specify the title of the plot. Similarly, xlabel() and

ylabel() specify the labels on the x and y axes.
- The legend() function is used to label different graphs on the same plot.

B. *Pandas*

Pandas works with tabular data such as data stored in spreadsheets and databases. pandas helps in cleaning, exploring and processing data. In pandas, a mapping from keys to values is called a Series and a mapping from a Series to column name, or equivalently a data table, is called a DataFrame.

We use several pandas functions in answering the scientific questions, the most important of which are mentioned below:

- read_csv(url, names = 'list') will import a data file into the namespace and store the contents of the file into a variable. This variable can be used to access the columns of the file. The columns of the file are named using names = [list containing names of the columns].
- DataFrame.groupby() groups a DataFrame according to the values in a column to create a GroupBy object. It can then be operated on using aggregate functions like mean, median, max, etc.
- DataFrame.dropna() is used to drop rows that have NaN values.
- DataFrame.fillna() is used to fill NaM values with a specified value.
- DataFrame.apply() is used to apply a function along an axis of the DataFrame.
- A new column can be added to a DataFrame using the syntax:

```
df['New Col'] = series
```

C. *Seaborn*

Seaborn is a Python library for making statistical graphics. It builds on top of matplotlib and integrates closely with pandas data structures. By convention, it is imported with the shorthand 'sns'.

```
import seaborn as sns
sns.set() #apply the default theme
```

The seaborn functions used in answering scientific questions are explained as follows:

- sns.heatmap(data, square = None, annot = None, cbar = None) plots rectangular data as a color-encoded matrix. If 'square' is True, each cell will be square-shaped. if 'annot' is True, each data value will be written in each cell. 'cbar' tells whether to draw a colorbar.
- sns.barplot(data, x = None, y = None) represents an estimate of central tendency for a numeric variable with the height of each rectangle. x and y are inputs for plotting data.
- sns.boxplot(data, x = None, y = None) shows distributions of quantitative data in a way that facilitates comparisons between variables or

across levels of a categorical variable. x and y are inputs for plotting data.
- sns.histplot(data, x = None, y = None, kde = False) represents the distribution of one or more variables by counting the number of observations that fall within discrete bins. It can add a smooth curve obtained using a kernel density estimate (kde).

D. Scikit-Learn

Scikit-Learn is a module for machine learning built on top of SciPy, NumPy and matplotlib. It has simple and efficient tools for predictive data analysis. We use several sklearn functions, the most important of which are mentioned below.

- sklearn.model_selection.train_test_split(X, y) splits arrays or matrices into random train and test subsets. The train subset is used to fit the model whereas the test subset is used to predict the labels given the features to the model.
- sklearn.linear_model.LogisticRegression is used to predict binary outcomes using one-vs-rest scheme or cross-entropy loss.
- sklearn.metrics.accuracy_score accepts values for model's predicted labels and true labels and computes the accuracy score
- sklearn.metrics.confusion_matrix is a table used to assess where the model made errors in predicting labels. The rows represent predicted labels and the columns represent actual labels.

IV. Answers to the Questions (With Appropriate Illustrations)

1. Does the ratio of first serve won (FSW) to second serve won (SSW) affect the outcome of a match?

This question is based on the dataset AusOpen-men-2013.csv. We import the modules pandas, seaborn and matplotlib. We import the csv file, use pd.read_csv to create a variable data and replace NaN values with 0.

```
import pandas as pd
import seaborn as sns; sns.set()
import matplotlib.pyplot as plt
df =
pd.read_csv('AusOpen-men-2013.csv')
df.fillna(0, inplace = True)
```

Since we want the ratio of the FSW to the SSW for each player, we update the DataFrame to include 2 new columns, Ratio.1 and Ratio.2. They contain the ratio of the FSW to the SSW for the winner and loser.

```
df['Ratio.1'] = df['FSW.1'] /
df['SSW.1']
df['Ratio.2'] = df['FSW.2'] /
df['SSW.2']
```

Then, we plot the distribution of the FSW ratio for each player, grouped by the match outcome

```
sns.histplot(data=df,
x='Ratio.1', hue='Result',
alpha=0.5)
```

```
sns.histplot(data=df,
x='Ratio.2', hue='Result',
alpha=0.5)
```

The output is as follows.

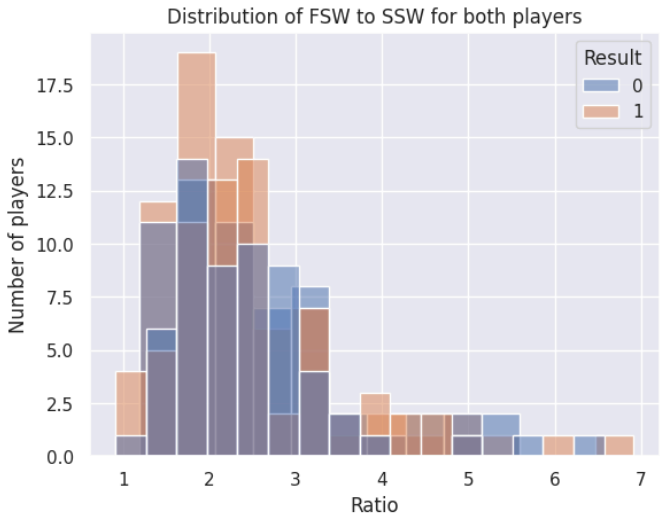

Fig. 1. Distribution of FSW to SSW for both players

As we can see, winners have a slightly higher FWS/SSW ratio compared to losers (considering that 0 = loser and 1 = winner).

2. Is there a correlation between the number of breakpoints created (BPC) and the final number of games won (BPC) by a player in a round?

This question is based on the dataset AusOpen-women-2013.csv. The modules required are pandas, seaborn and matplotlib.pyplot. After importing the dataset, we group it according to values in the column FNL.1 and take the median of the column BPC.1. This will give the median number of breakpoints created for a player who won 0, 1, or 2 games in a round.

```
df = df.groupby('FNL.1',
as_index =
False)['BPC.1'].median()
```

Then, we plot a barplot using seaborn that shows the no. of games won in a round on the x-axis and the no. of breakpoints created on the y-axis.

```
df = pd.DataFrame([exp,
grad]).T.dropna()
df = df[df['Grad. rate'] < 101]
sns.barplot(data = df, x =
'FNL.1', y = 'BPC.1')
```

The output in the form of a bar-plot is shown below.

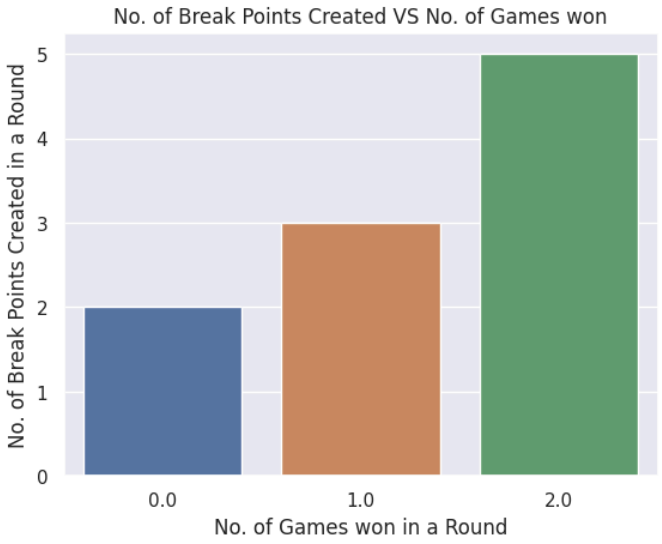

Fig. 2. No. of Breakpoints Created vs No. of Games Won

As we infer, there is a definite relationship between both factors. If a player has won a higher number of games in a round, then it is more likely that she has created more breakpoints.

3. **Is there a correlation between the total points won and the first serve percentage of the winning and losing players?**

This question is based on the dataset FrenchOpen-men-2013.csv. It can be answered by using the modules pandas, seaborn and matplotlib. After importing the modules and the dataset, we filter the relevant columns and create separate DataFrames for winning and losing players based on the result of the match.

```
relevant_columns = ['FSP.1', 'FSP.2', 'TPW.1', 'TPW.2', 'Result']

relevant_data = data[relevant_columns]

winning_players = relevant_data[relevant_data['Result'] == 1]

losing_players = relevant_data[relevant_data['Result'] == 0]
```

Then we use seaborn.histplot to view the distribution of TPW vs FSP for winning and losing players.

```
sns.histplot(data=winning_players, x='FSP.1', kde=True, label='Winning Player')

sns.histplot(data=losing_players, x='FSP.1', kde=True, label='Losing Player')
```

The output is as follows.

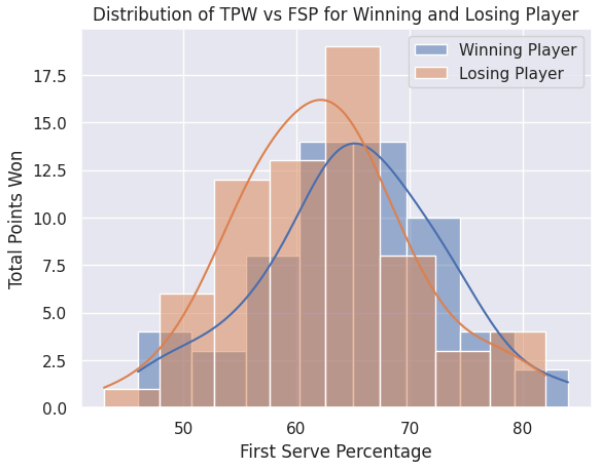

Fig. 3. Distribution of TPW vs FSP for Winning and Losing Players

As we can infer, the winners have only a slightly higher FSP than losers. Moreover, the distribution is not linear, implying that with increase in FSP, TPW does not increase. In fact, TPW reaches a maximum at around 65% and then decreases. This is due to the fact that average FSP for any player is around 50%, so it is more likely that a higher number of points will be gained at around this FSP.

4. **Assuming that higher first serve percentage (FSP) can contribute to higher chances of winning the match, is there a significant difference in FSP for winners and losers?**

This question is based on the dataset FrenchOpen-men-2013.csv. We require the modules pandas, seaborn and matplotlib. Since we wish to compare the First Serve Percent of winners and losers, we use a special function of seaborn. We create a boxplot, one for winning players and one for losing players to compare the medians, interquartile ranges and outliers for the FSPs of the two groups to see if there is a difference in their distribution.

```
sns.boxplot(data = df, x = 'Result', y = 'FSP.1')
```

The output is displayed below.

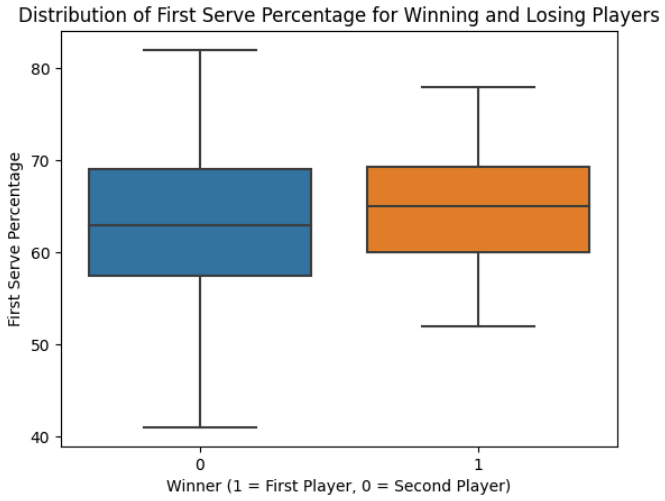

Fig. 4. Distribution of First Serve Percent of Winning and Losing Players

Contrary to expectation, there is no significant difference in their distributions. This is because a coin toss is equally likely to give heads or tails. Surely winners have a slightly higher FSP, but the result is dependent more on the skills of the player than on luck.

5. **Is there a significant relationship between a player's first serve percentage (FSP) and their final number of games (FNL)?**

This question is based on the dataset USOpen-men-2013.csv. We require the modules pandas, seaborn and matplotlib. After importing them, we group the dataset according to values in FNL.1 and take the median of values in FSP.1. Thus, we get FSP for every FNL (0, 1, 2, 3).

```
data = pd.read_csv('USOpen-men-2013.csv').groupby('FNL.1', as_index = False)['FSP.1].median()
```

Then, we use the barplot function of seaborn so that we can view the distribution in graphical form.

```
sns.barplot(data = data, x = 'FNL.1', y = 'FSP.1')
```

The output is:

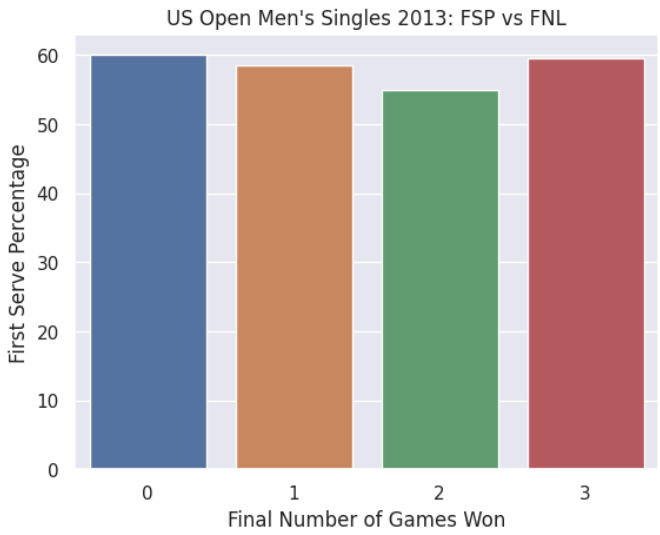

Fig. 5. Distribution of FSP vs FNL

Disappointingly, there is not much difference between FSP and FNL. As we inferred from the last question, the skills of the player matter more than luck.

6. **Is there a correlation between the number of aces (ACE) served by a player and their probability of winning the match (Result)?**

This question is based on the dataset USOpen-women-2013.csv. We require the modules pandas, seaborn and matplotlib.pyplot. After importing the modules and the dataset, we select the required columns (‘ACE.1, ‘ACE.2’ and ‘Result’) from the DataFrame.

```
df = df[[‘ACE.1’, ‘ACE.2’,
‘Result’]]
```

Then, we create a new column in the DataFrame to indicate whether Player 1 or Player 2 had more aces. We use the lambda keyword which is a placeholder that will be used to hold the value we want to pass (‘Player 1’ or ‘Player 2’) into the function expression (apply).

```
df[‘More Aces’] =
df.apply(lambda row: ‘Player 1’
if row[‘ACE.1’] > row[‘ACE.2’]
else ‘Player 2’, axis = 1)
```

Similarly, we create another column to indicate whether the player with more aces won the match.

```
df[‘Aces_Winner’] =
df.apply(lambda row:
(row[‘More_Aces’] == ‘Player 1’
and row[‘Result’] == 1) or
(row["More_Aces"] == "Player 2"
and row["Result"] == 0) else 0,
axis=1)
```

Then, we calculate the correlation between the number of aces and the probability of winning the match.

```
correlation = df[["ACE.1",
"Result"]].corr().iloc[0,1]
```

Finally, we create a relationship plot to visualize the relation between the number of aces and the probability of winning the match.

```
sns.relplot(x = ‘ACE.1’, y =
‘Result’, data = df, kind =
‘line’)

plt.title(‘No. of Aces vs
Result’)
```

The output is obtained as:

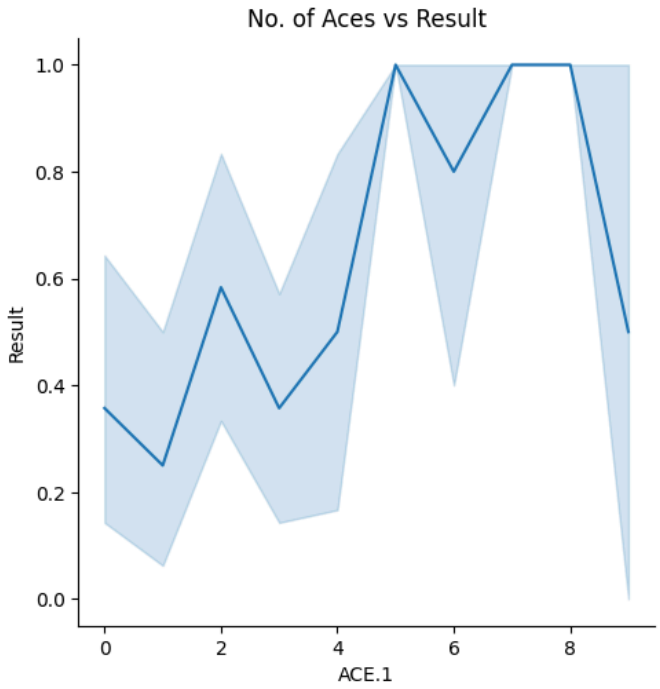

Fig. 6. No. of Aces vs Result

Disregarding the few outliers, there is a clear correlation between the number of aces and the result of the match.

7. With what accuracy can we predict the outcome of any match based on the match statistics?

This question is a continuation of the first question and is based on the dataset Wimbledon-men-2013.csv. We require the modules pandas, seaborn, matplotlib.pyplot, sklearn.model_selection.train_test_split(), sklearn.linear_model.LogisticRegression, sklearn.metrics.confusion_matrix, and sklearn.metrics.accuracy_score. We let the feature matrix be all the columns from the dataset except for 'Player1', 'Player2', 'Result' and let the target matrix be the column 'Result'.

```
g. ACT'X = df.drop(['Result',
'Player1', 'Player2'], axis=1)

y = df['Result']
```

After splitting X and y into train and test subsets, we train the model LogisticRegression (a model that estimates the probability of an event occurring) on X_train and y_train.

```
X_train, y_train, X_test, y_test
= train_test_split(X, y,
test_size = 0.3, random_state =
42)

model = LogisticRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

Then, we predict the labels for X_test and print the accuracy score and the confusion matrix.

```
print(f'Accuracy with which
result can be predicted:
{accuracy_score(y_test,
y_pred)*100}%')


cm = confusion_matrix(y_test,
y_pred)
```

```
sns.heatmap(cm.T, square=True,
annot=True, fmt='d', cbar=False)

plt.xlabel('true result of
match')

plt.ylabel('predicted result of
match')
```

The result is obtained as follows:

```
Accuracy with which result can
be predicted: 94.28571428571428%
```

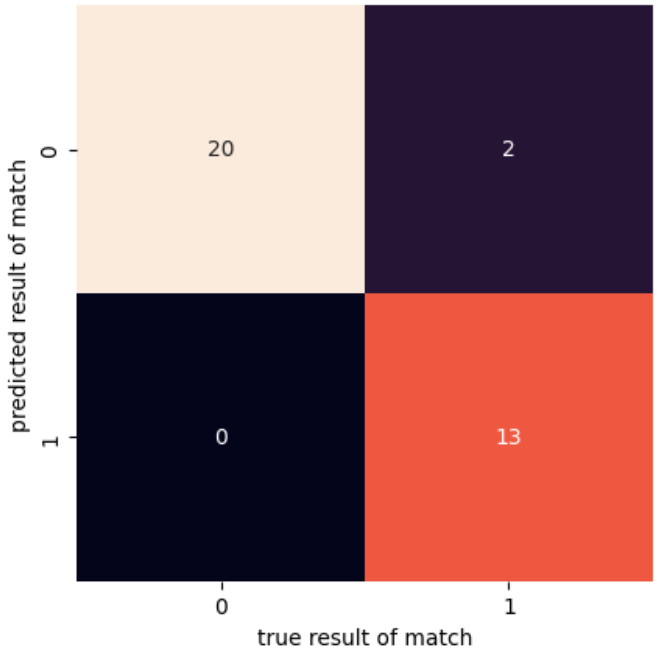

Fig. 7. Predicted Result of Match vs True Result of Match

As inferred from the confusion matrix, the model is good at predicting the result of any random subset of the dataset.

8. **Compare the match statistics (ACE, DBF, WNR, UFE, BPW, TPW, FNL, Result) of the winner and loser in the final round (Round 7).**

This question is based on the dataset Wimbledon-women-2013.csv. We require the modules pandas, seaborn and matplotlib.pyplot. After loading the modules and the dataset, we filter the data to include only Round 7 statistics and only the columns of interest.

```
round_7_data = df[df["Round"] ==
7]

loser_cols = ["ACE.1", "DBF.1",
"WNR.1", "UFE.1", "BPW.1",
"TPW.1", "FNL.1", "Result"]

winner_cols = ["ACE.2", "DBF.2",
"WNR.2", "UFE.2", "BPW.2",
"TPW.2", "FNL.2", "Result"]
```

Then, we rename the columns ‘ACE.1’, ‘DBF.1’, etc. and ‘ACE.2’, ‘DBF.2’, etc. to the common names ‘ACE’, ‘DBF’, etc. for easier graphical implementation.

```
loser_data =
round_7_data[loser_cols].rename(
columns={"ACE.1": "ACE",
"UFE.1": "UFE", "WNR.1": "WNR",
"DBF.1": "DBF", "BPW.1": "BPW",
"TPW.1": "TPW", "FNL.1": "FNL"})

winner_data =
round_7_data[winner_cols].rename
(columns={"ACE.2": "ACE",
"UFE.2": "UFE", "WNR.2": "WNR",
"DBF.2": "DBF", "BPW.2": "BPW",
"TPW.2": "TPW", "FNL.2": "FNL"})
```

Finally, we combine the data and use seaborn.barplot to plot 6 subplots.

```
combined_data = pd.concat([loser_data.assign(WinnerOrLoser="Loser"), winner_data.assign(WinnerOrLoser="Winner")])

fig, ax = plt.subplots(nrows = 3, ncols = 2, figsize=(8,6))

sns.barplot(x="WinnerOrLoser", y="ACE", data=combined_data, ax=ax[0, 0])

sns.barplot(x="WinnerOrLoser", y="WNR", data=combined_data, ax=ax[0, 1])

sns.barplot(x="WinnerOrLoser", y="UFE", data=combined_data, ax=ax[1, 0])

sns.barplot(x="WinnerOrLoser", y="BPW", data=combined_data, ax=ax[1, 1])

sns.barplot(x="WinnerOrLoser", y="DBF", data=combined_data, ax=ax[2, 0])

sns.barplot(x="WinnerOrLoser", y="FNL", data=combined_data, ax=ax[2, 1])

fig.suptitle("Performance Comparison of Winner and Loser in Round 7 of Wimbledon 2013 Women's Singles")
```

The output is obtained as follows:

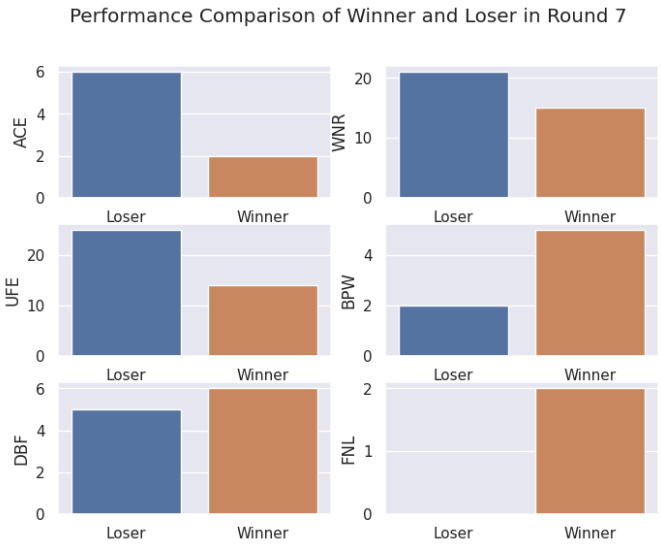

Fig. 8. Performance Comparison of Winner and Loser in Round 7

As we can see, although the loser put up a tough fight with higher number of aces and winners and lower number of double faults, the winner had a higher number of breakpoints and fewer unforced errors. The winner won the first two matches, ending the tournament.

V. Summary of the Observations

We can infer from the first answer that winners are likely to have a higher first serve won to second serve won ratio.

The second answer says that with an increase in the number of breakpoints created, the number of games won increases.

In the third answer, we see that total points won peaks at a first serve percentage of around 65%.

In the fifth question, our hypothesis was that higher first serve percentage (FSP) can contribute to higher chances of winning the match. However, contrary to our expectations, the answer reveals that there is no such correlation.

In the fifth question, we see a player who has won more games does not necessarily have a higher first serve percentage. This is probably because the outcome of the match is more dependent on the skills of the player than on probability.

The sixth answer shows the correlation between the number of aces of a player and the probability of winning the match.

The seventh answer quite accurately predicts the result of any random subset of the dataset by training the model on a random train subset.

The eighth answer compares the match statistics (ACE, DBF, WNR, UFE, BPW, TPW, FNL, Result) of the winner and loser in the final round (Round 7). We see that although the match statistics were close, the winner won the first 2 games, winning the tournament.

This was a fairly simple and straightforward way of gaining meaningful results from a dataset. The data narrative helped gain insight about how to frame relevant scientific questions and how to appropriately answer them using various Python libraries. Moreover, it helped explore various aspects of the dataset like how various match statistics like breakpoints won, number of aces, etc. may or may not vary with the outcome of the match.

VI. Unanswerable Questions

Questions such as 'How likely is it for a world top 10 player to win the tournament?' and 'Is there an increase in audience number as the match rounds progress?' remain unanswerable. This is because the required data to answer them is missing from the dataset, or there were certain discrepancies while collecting the data, or there is some subjectivity involved in answering the question.

References

[1] Chen, Daniel Y. *Pandas for everyone: Python data analysis*. Addison-Wesley Professional, 2017.

[2] VanderPlas, Jake. *Python data science handbook: Essential tools for working with data*. " O'Reilly Media, Inc.", 2016.

[3] Pandas. "User Guide." Accessed April 20, 2023. https://pandas.pydata.org/docs/user_guide/index.html

[4] Matplotlib. "Users Guide." Accessed April 20, 2023. https://matplotlib.org/stable/users/index.html#

[5] Seaborn. "Tutorial." Accessed April 20, 2023. https://seaborn.pydata.org/tutorial/introduction.html

[6] Scikit-Learn. "User Guide." Accessed April 20, 2023. https://scikit-learn.org/stable/user_guide.html

Acknowledgement

I would like to thank Prof. Shanmuga for giving me the opportunity to gain insight into real-life applications of pandas, numpy, matplotlib, and sklearn. I would also like to thank the providers of the data for the online dataset –Shruti Jauhari, Aniket Morankar and Ernest Fokoue. Finally, I thank creators of online resources on pandas, seaborn and matplotlib such as Jake VanderPlas, Michael Waskom and John Hunter; this data narrative would have been impossible without them.