ES-114
Data Narrative Assignment - 3

Guntas Singh Saran
Computer Science and Engineering Dept.
Indian Institute of Technology, Gandhinagar
Roll No - 22110089
Prof. Shanmuganathan Raman
Computer Science and Engineering Dept.
(Jointly with Electrical Engineering)

***Abstract*—This Data Narrative analyzes the Tennis Major Tournaments dataset, which includes match statistics for tennis players from four Major Tennis tournaments of the world: Australia Open, French Open, US Open, and Wimbledon. The dataset is explored using various Python libraries such as Pandas, NumPy, Matplotlib, Seaborn, Plotly, and Scikit-Learn. The report poses several scientific questions and hypotheses about the data, ranging from basic descriptive statistics to more advanced probabilistic analysis and analysis using Machine Learning Models to test for Regression, Classification and Clustering. The results of the analysis are presented with tables, plots, and code snippets to support the answers. Overall, the analysis provides insights into the performance of tennis players in major tournaments and how various factors such as serves, net points, and break points can affect their success. The report concludes with a summary of observations, references, and acknowledgments.**

I. Overview of the Dataset

The Tennis Major Tournaments dataset contains match statistics for 4 major tennis tournaments of 2013. These tournaments include the Men's and Women's Australia Open, French Open, US Open, and The Wimbledon. The dataset includes a total of 951 matches, with almost 39 different variables for each match, including the names of the players, the number of games won by each player, the number of aces and double faults committed, and various other statistics. The dataset also includes the round of the tournament at which the match was played. This dataset provides an excellent opportunity for data analysis in terms of exploration in Regression, Classification, and Clustering and various other statistical techniques in the field of sports analytic.

II. Scientific Questions/Hypothesis on AAUP

Following are the questions proposed to be answered by the analysis of the **Tennis Major Tournaments dataset**.

The Men's Australia Open Tournament:

A. ***What distribution does the First Serve Percentage follow for Player1 and Player2? And do they compare to each other? Does it state out First Serve Percentage as good criteria for judging the Player1 and Player2?***

This question aims to explore the importance of first serve percentage as a good criteria for comparing the level of two players in a match. If both the distributions come out as highly identical then it won't count towards feature in analysis else it does.

B. ***Is there a correlation between the first serve percentage and first serve win for Player1 and Player2?***

It is expected that there is no good correlation between these and the result will reveal an important insight into the calculation of the first serve percentage.

The Women's Australia Open Tournament:

C. ***Which all columns/features have high positive or negative correlation?***

This question aims to explore the redundancy in the features given. For features having a strong correlation, only one of them is sufficient for modeling. Reducing the number of features, will allow for our ML model to have an efficient and less time consuming approach to training.

The Men's French Open Tournament:

D. ***How do the breaking points created and won by the loser and by winner vary?***

It is expected for the winner to have a strong side on this and hence we will explore the above using box plots.

The Women's French Open Tournament:

E. ***Can we train an AI model to predict the win results? What sort of model would it be?***

This question would require us to split the dataset into features and target segments. Since the result of the match is binary, hence a Classifier model would be best suited for its analysis.

The Men's US Open Tournament:

*F. **Can the above model be made more efficient by having lesser number of features? How would such a model work? How would the reduced features be decided?***

Since we wish to classify our data into two categories: Win or Loss, hence this question aims to explore the importance of Clustering in Unsupervised Learning scenario and how it still works with reduced number of features. The reduction in features will open up a new area of discussion called Principal Component Analysis.

The Women's US Open Tournament:

*G. **How does the performance of top player of the tournament vary in each round when compared to other players?***

This question aims to build a statistical comparison between a player and the rest based upon a single feature. Although it may be extend for other features but this poses a nice start towards a bigger exploration.

The Men's Wimbledon:

*H. **Considering the dataset's limitation, can we still create a way to arrange the players in on the basis of their number of wins?***

This question aims to explore the top players based upon their number of wins. The code snippet acts a supporter to the follow up question.

The Women's Wimbledon:

*I. **How does the performance of the winner vary in the Finals and in the rest of tournament rounds?***

Although it may be extended to any player and any match, the question aims to give a brief but detailed insight into the performance analysis of a player throughout the tournament. We will explore the above using a Series Line Plot.

III. Details of Libraries and Functions

The presented dataset was analysed using Python and several of its useful, built-in libraries like **Numpy, Pandas, MatPlotLib, Seaborn, Plotly** and **Scikit-Learn.** Some of the most commonly used functions from these libraries include:

- Pandas' **.read_csv()** function was used to read data onto a DataFrame from a CSV File.
- Pandas' **.value_counts()** function enabled to count the unique data-points in the dataset.
- Pandas' **.nlargest()** and **.nsmallest()** was used to filter out the N highest and lowest data-points in the DataFrame.
- MatPlotLib's **.pyplot()** was the primary function that enabled the plotting of the graphs.
- For interactive visualizations, Plotly's module was used, especially the library: **plotly.express** as **px**, **plotly.graph_objects** as **go**, and **plotly.figure_factory** as **ff**:
  - **ff.create_distplot()** was used to create distribution Plot
  - **px.scatter()** was used to create scatter plot.
  - **px.imshow()** was used to create confusion matrix of a 2D array as a heatmap.
  - **px.box()** was used to create box plots.
  - **px.3d_scatter()** was create scatter plots in 3D.
  - **go.Scatter3D()** was also used to create 3D scatter plot.
  - **px.bar()** to create bar plots.
- For performing regression, classification, and clustering, several of Scikit-Learn's libraries were used:
  - The **sklearn.linear_model** library had **LinearRegression** model which was used for performing Regression on the dataset.
  - From **sklearn.model_selection** library several modules were imported like **train_test_split**, **cross_val_score**, and **LeaveOneOut**. All of these are used in preparing the data for model fitting.
  - Several prediction accuracy functions like **mean_squared_error** and **accuracy_score** were imported from the **sklearn.metrics** library.
  - The **GaussainNB** classifier was imported from the **sklearn.naive_bayes** library.
  - To scale the data for fitting of PCA, **StandardScaler** was imported from **sklearn.preprocessing** library.
  - For performing Principal Component Analysis on the data, the **PCA** model was imported from the **sklearn.decomposition** library.
  - To perform clustering, **KMeans** model was imported from **sklearn.cluster** library.

IV. Answers to the Questions

*A. **What distribution does the First Serve Percentage follow for Player1 and Player2? And do they compare to each other? Does it state out First Serve Percentage as good criteria for judging the Player1 and Player2?***

ANS. As per Fig. 1, the distribution are normally distributed. The kernel density plots of these two are highly identical. The identical nature of these two graphs is an indicator of the fact the first serve percentage is not a good feature to be added when we wish to judge player 1 and player 2.

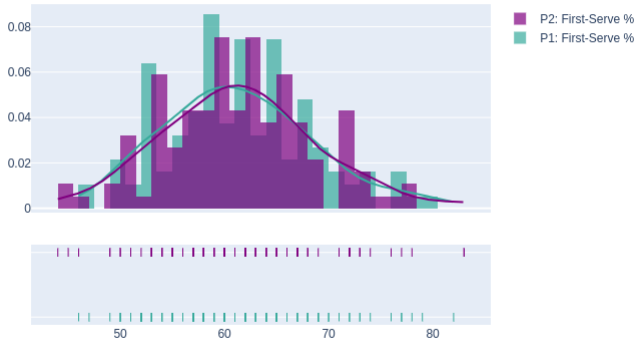

Fig. 1: Distribution plots of first serve % of P1 & P2

B. Is there a correlation between the first serve percentage and first serve win for Player1 and Player2?

ANS. As per Fig. 2, the first serve win and first serve percentage of player 1 seem to have a positive correlation but are actually not so related since the correlation factor came out to be **0.18869064127952748**.

The code snippet for the following is:

```
print("Correlation Factor:")
print(ausM["FSW.1"].corr(ausM["FSP.1"]))

>> Correlation Factor:
>> 0.18869064127952748
```

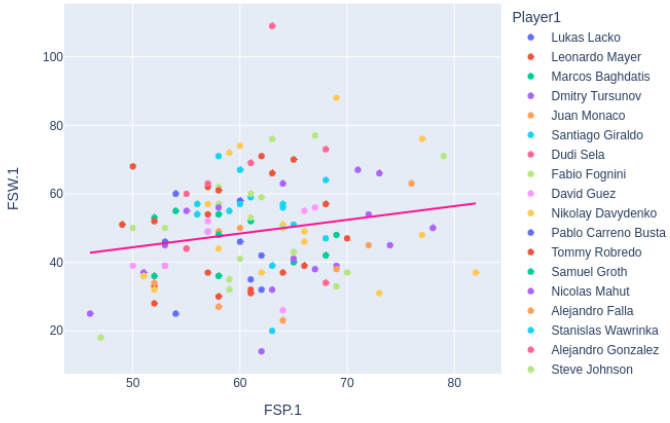

Fig. 2: Scatter plot for P1 between 1st serve % and 1st serve win.

And as per Fig. 3, again there isn't much strong positive correlation between FSP and FSW for player 2 too, hence the correlation factor came out to be **0.2671214679032754**.

The code snippet for the following is:

```
print("Correlation Factor:")
print(ausM["FSW.2"].corr(ausM["FSP.2"]))

>> Correlation Factor:
>> 0.2671214679032754
```

The correlation factor between two random variables is given by:

$$\rho_{A,B} = \frac{cov(A,B)}{\sigma_A \sigma_B} \tag{1}$$

where,

$$cov(A,B) = E[(A-\mu_A)(B-\mu_B)]$$

$$cov(A,B) = E[AB] - E[A]E[B]$$

$$\mu_X = E[X]$$

$$\sigma_X^2 = E[X^2] - (E[X])^2$$

Fig. 3: Scatter plot for P2 between 1st serve % and 1st serve win.

The reason for their weak correlation despite looking like doing a simple is as follows:

- The first serve percentage for a player is the percentage of number of first serves that they make during the match.
- The first serve won is the percentage that out of the first serves that the player makes, how many of them do they win.

For instance, in a match between Jannik Sinner and Casper Rudd, the stats are:

Sinner:

– First Serves Made = 56% (41/73)
– First Serves Won = 76% (31/41)

Rudd:

– First Serves Made = 65% (49/75)
– First Serves Won = 65% (32/49)

At first glance, Ruud made 65 per cent to Sinner's 56 per cent, so it was probably the Norwegian. But the Italian won 76 per cent to 65 per cent, which now muddies the water. The truth is that their first-serve performance was almost identical. You would never know it by looking at the two statistics separately, but it's immediately recognisable once you combine them.

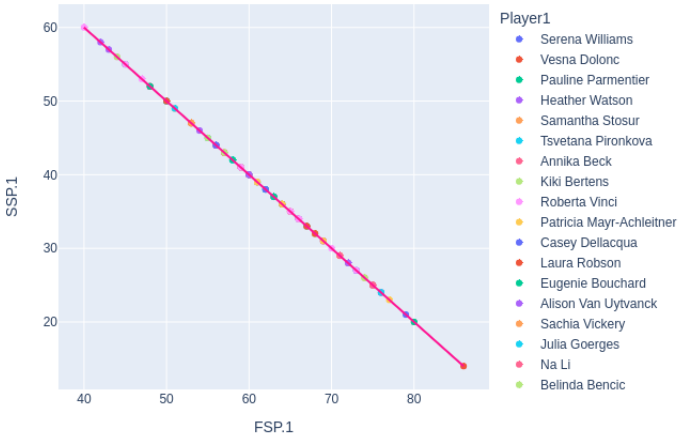

Fig. 5: Scatter plot showing the correlation b/w FSP and SSP

C. Which all columns/features have high positive or negative correlation?

ANS. As seen in the Fig. 4 and Fig. 5 it is evident that the Series FSP(first serve percentage) and SSP(second serve percentage) have high negative correlation of -1.0. Furthermore from the Fig. 6, the Series NPA(net points attempted) and NPW(net point won) also have a high positive correlation of 0.98.
The Code snippet is shown below:

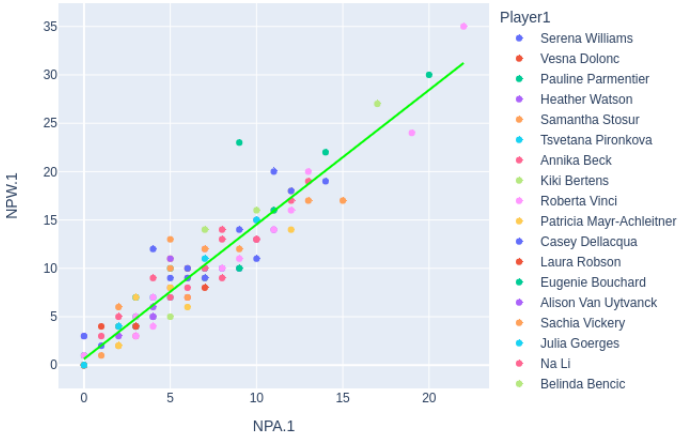

Fig. 6: Scatter plot showing the correlation b/w NPA and NPW

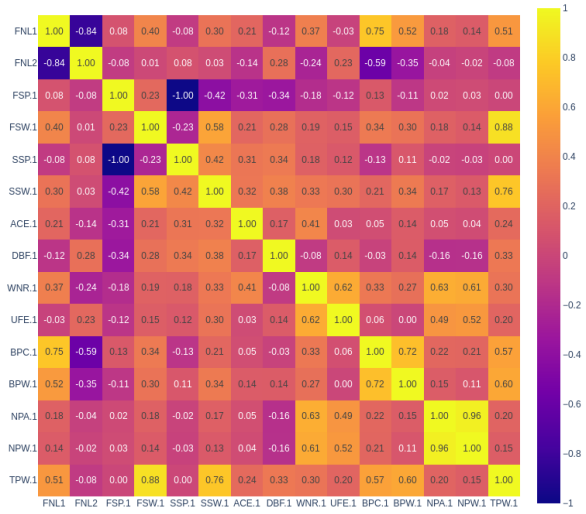

Fig. 4: Covariance Matrix showing the correlation between various features

Hence, whenever we go for model training of the features, we can safely remove one of these series while the other still acts the same.

D. How do the breaking points created and won by the loser and by winner vary?

ANS. In all the four figures, the mean of the winner(Result == 1 for P1 and Result == 0 for P2) is always greater than the loser. Hence, more number of breaking points and created and won by the winner of the match.

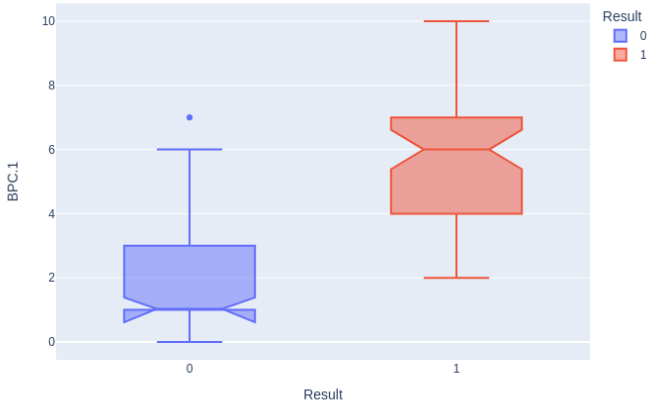

Fig. 7: Box plot of BPC.1 against the winner

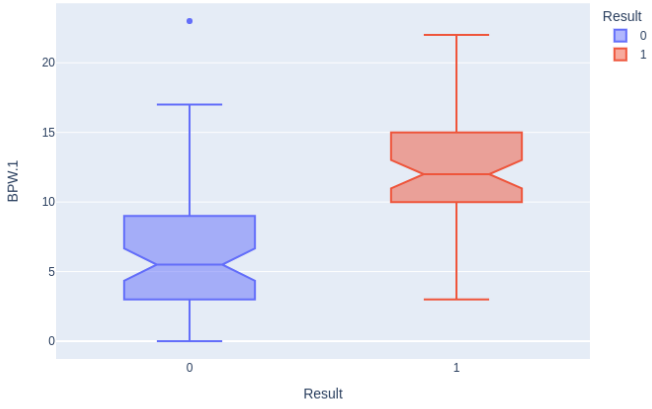

Fig. 8: Box plot of BPW.1 against the winner

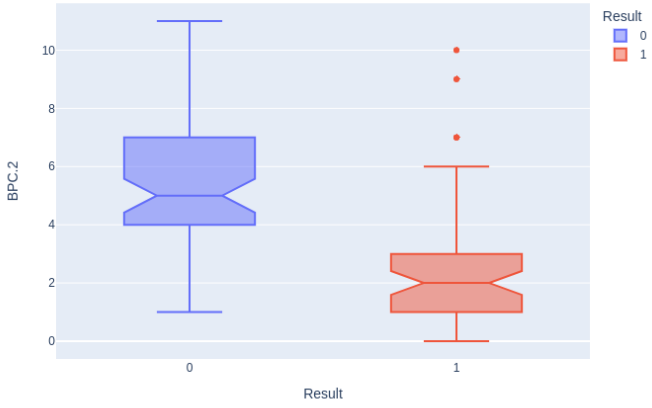

Fig. 9: Box plot of BPC.2 against the winner

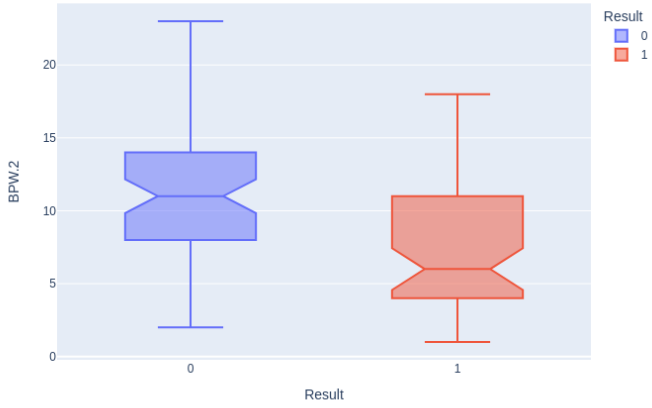

Fig. 10: Box plot of BPW.2 against the winner

E. Can we train an AI model to predict the win results? What sort of model would it be?

ANS. Notice that the Result Series has only two possible values 0 and 1. Hence the whole dataset can be seen to correspond towards two classes. So, in such scenario, a **Classifier** model would be best suited.
In the following analysis, the **GaussianNB: Gaussian Naive Bayes** Classifier was used.

The data was first split into two parts: **Features** and **Target**. No doubt that the Target array would the "Result" column, whereas the features 2D array was chosen to be all the 24 remaining columns.

The code snippet for the following is:

```
features = fraW[["FSP.1", "FSW.1", "SSP.1",
    "SSW.1", "ACE.1", "DBF.1", "WNR.1",
    "UFE.1", "BPC.1", "BPW.1", "NPA.1",
    "TPW.1", "FSP.2", "FSW.2", "SSP.2",
    "SSW.2", "ACE.2", "DBF.2", "WNR.2",
    "UFE.2", "BPC.2", "BPW.2", "NPA.2",
    "TPW.2"]].to_numpy()
target = fraW["Result"].to_numpy()
print(features)
print()
print(target)

>>
[[62. 18. 38. ... 6. 3. 57.]
 [57. 23. 43. ... 3. 4. 48.]
 [76. 30. 24. ... 4. 14. 56.]
 ...
 [72. 28. 28. ... 10. 2. 87.]
 [52. 14. 48. ... 0. 2. 16.]
 [69. 27. 31. ... 2. 3. 56.]]

>>
 [0 1 1 0 0 1 1 1 0 1 0 1 1 1 1 1 0 0 0 1 0 0
    0 1 0 1 1 0 0 0 1 1 0 0 1 1 0
 0 1 0 0 0 1 0 1 1 1 1 0 1 0 1 1 0 0 1 0 1 1
    1 0 0 0 1 0 0 0 0 0 1 0 1 0 1
```

```
 1 1 0 1 1 0 0 1 0 1 0 0 1 1 0 1 0 0 0 0 0 1
     0 0 1 0 0 0 0 1 0 0 0 1 0 0 0
 1 0 0 0 0 0 1 1 1 0 0 0 1 0 1 1]
```

Now for a rough estimate the features dataset was split into a 1:4 ratio, with 20% being the training data and the remaining 80% as the testing data. For this testing, an accuracy score of **0.7722772277227723** was found.

The code snippet to achieve the same is:

```
model = GaussianNB()
Xtrain, Ytrain = features[:26], target[:26] #
    20% Training Data
Xtest, Ytest = features[26:], target[26:] #
    80% Testing Data
model.fit(Xtrain, Ytrain)
y_model = model.predict(Xtest)
print(accuracy_score(y_model, Ytest))

>> 0.7722772277227723
```

But this accuracy score was based upon the training of 20% of the data. To account the size of the training data, we performed a **5-Fold Cross Validation** and obtained the 5 accuracy scores. The mean of these accuracy scores, will give the right accuracy score of the **GaussianNB** model on this dataset.

The code snippet to do the same:

```
model = GaussianNB()
scores = cross_val_score(model, features,
    target, cv = 5)
print(scores)
print(scores.mean())

>> [1. 0.88461538 0.92 0.84 0.88]
>> 0.9049230769230769
```

Hence the Classifier actually did a pretty good job by achieving a high accuracy score of almost **90.5%!**

F. ***Can the above model be made more efficient by having lesser number of features? How would such a model work? How would the reduced features be decided?***

ANS. Certainly Yes! There exists a special statistical technique called **Principal Component Analysis** aka **PCA**. **PCA** works in the ways of eigenvector decomposition. Considering only the top eigenvectors having the highest eigenvalues, captures most of the essence of the data. This is because eigenvectors with high eigenvalues are those dimensions in N-D space that have the highest mean squared error, and hence they pass close to the maximum number of points.
Mathematically, this is how it works:

- Consider a data matrix X of size N x M: has M data points as its columns and N dimensions in each data point
- Scale the matrix down by replacing each data point X_i by a_i and form a matrix A. This is the scaled version of X.

$$a_i = \frac{X_i - \mu}{\sigma} \tag{2}$$

where,

$$\mu = \frac{\sum_{i=1}^{M} X_i}{M} \tag{3}$$

- Find the covariance matrix of this matrix A, which has dimension N x N.

$$C = (\frac{1}{M-1})AA^T \tag{4}$$

- Calculate the eigenvalues and eigenvectors of this covariance matrix C. Select the top K eigenvectors based upon the top K eigenvalues. Add these K eigenvectors in a matrix W as columns. This matrix is of dimension N X K. V_i are the eigenvectors.

$$CV_i = \lambda_i V_i \tag{5}$$

- Finally project the scaled matrix using the eigenvector matrix as:

$$B = W^T A \tag{6}$$

- Since W^T has dimension K x N and A^T has dimension N X M, hence the B has dimension K X M. So we reduced our dimensions from N to K. This B is the final data matrix to be worked on.

The code snippet for PCA for dimensionality reduction to 3:

```
features = usM[["FSP.1", "FSW.1", "SSP.1",
    "SSW.1", "ACE.1", "DBF.1", "WNR.1",
    "UFE.1", "BPC.1", "BPW.1", "NPA.1",
    "TPW.1", "FSP.2", "FSW.2", "SSP.2",
    "SSW.2", "ACE.2", "DBF.2", "WNR.2",
    "UFE.2", "BPC.2", "BPW.2", "NPA.2",
    "TPW.2"]].to_numpy()
target = usM["Result"].to_numpy()
scaler = StandardScaler()
X_scaled = scaler.fit_transform(features)
model = PCA(n_components = 3)
model.fit(X_scaled)
X_3D = model.transform(X_scaled)
```

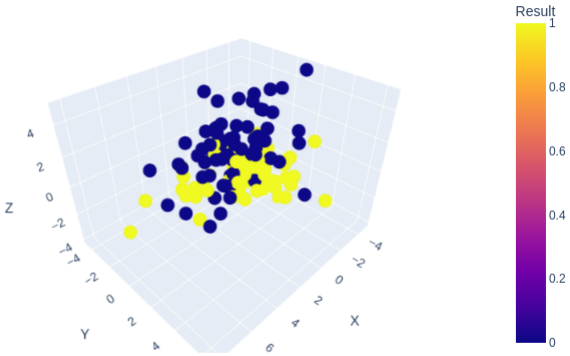

Fig. 11: The points as seen in 3D.

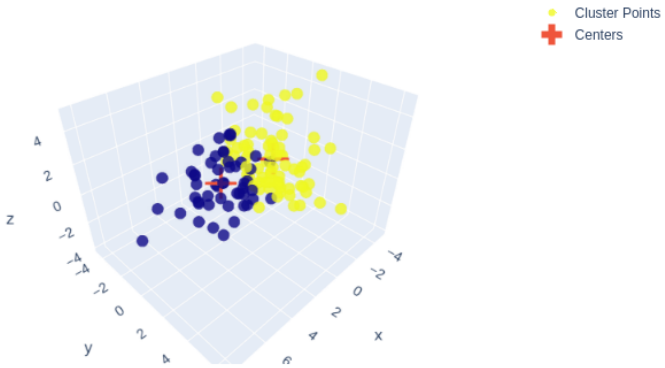

Fig. 12: KMeans Clustering in 3D, with their cluster centers.

Fig. 11 shows the original points in 24-D space in 3-D space. All these points correspond to each row of the dataframe, which are the datapoints in the dataset. These points were orginally in 24-dimensional space and hence by dimensionality reduction, we have brought these points into 3-dimensional space. This allows for us to visualize the datapoints in their entirety. The blue points all belong class of winners and yellow points belong to the class of losing side. Then clustering is performed using the built-in KMeans of scikit-learn.

```
from sklearn.cluster import KMeans
km = KMeans(n_clusters = 2, init = "random",
    n_init = "auto", max_iter = 300)
kmeans = km.fit(X_3D)
labelK = kmeans.labels_
center = kmeans.cluster_centers_
fig = go.Figure()
fig.add_trace(go.Scatter3d(x = X_3D[:,0], y =
    X_3D[:,1], z = X_3D[:,2], mode =
    "markers", marker = dict(size = 6, color
    = labelK, opacity = 0.8), showlegend =
    True, name = "Cluster Points"))
fig.add_trace(go.Scatter3d(x = center[:,0], y
    = center[:,1], z = center[:,2], mode =
    "markers", marker = dict(size = 15,
    opacity = 1, symbol = "cross"),
    showlegend = True, name = "Centers"))
fig.show()
```

Fig. 12 shows the two clusters formed corresponding to win/loss classes and their respective centers marked by red crosses.

G. How does the performance of top player of the tournament vary in each round when compared to other players?

ANS. Considering Serena Williams to the top player of the US Open, Fig. 13 shows the bar plot of number of breaking points created by her during each round.
Whereas 14 shows the same for rest of the players.

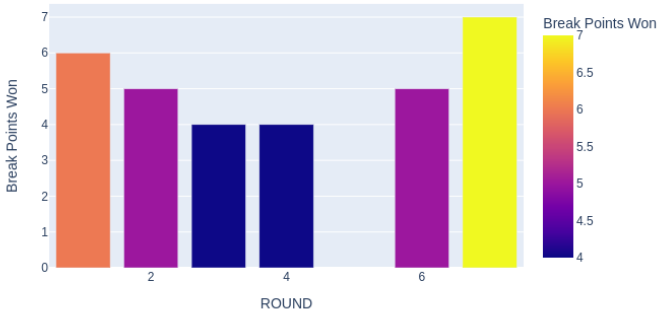

Fig. 13: Bar Plot for BPW in each round by S. Williams

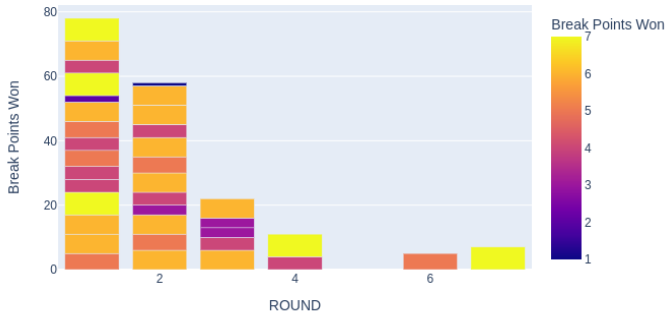

Fig. 14: Bar Plot for BPW in each round by other players

From the data plotted, it can be calculated that the average breaking point won by S. Williams is **5.166666666666667** and for the remaining players, it is **5.027777777777778**. Hence she certainly performed better in this respect.

The code snippet to achieve the same is:

```
df = usW[usW["Player 1"] == "S Williams"]
print(df["BPW.1"].mean())
print(usW[usW["Result"] == 1]["BPW.1"].mean())

>> 5.166666666666667
>> 5.027777777777778
```

H. Considering the dataset's limitation, can we still create a way to arrange the players in on the basis of their number of wins?

ANS. Yes, there's a tricky way to account for it. To analyse the data for number of wins, we need to create a list of distinct players.

The code snippet is:

```
players = set(wimM["Player1"].unique()) |
    set(wimM["Player2"].unique())
dfnew = pd.DataFrame({"Player" :
    list(players), "Wins" : 0})
for _, row in wimM.iterrows():
   if row["Result"] == 1:
      dfnew.loc[dfnew["Player"] ==
          row["Player1"], "Wins"] += 1
   else:
      dfnew.loc[dfnew["Player"] ==
          row["Player2"], "Wins"] += 1
dfnew = dfnew.sort_values("Wins", ascending =
    False)
dfnew.reset_index(inplace = True)
dfnew.drop("index", axis = 1, inplace = True)
```

TABLE I: Table of Matches won by each player

Player	Wins
A.Murray	7
N.Djokovic	6
J.Del Potro	5
T.Berdych	4
J.Janowicz	4
J.Melzer	3
I.Dodig	3
D.Ferrer	3
D.Brown	2
L.Kubot	2
M.Youzhny	2
I.Sijsling	2
T.Robredo	2
K.Nishikori	2
B.Paire	2
G.Zemlja	2
R.Gasquet	2
S.Stakhovsky	2

I. How does the performance of the winner vary in the Finals and in the rest of tournament rounds?

ANS. Continuing from the previous question, we can still create the list of top 10 players with most win.

```
players = set(wimW["Player1"].unique()) |
    set(wimW["Player2"].unique())
dfnew = pd.DataFrame({"Player" :
    list(players), "Wins" : 0})
for _, row in wimW.iterrows():
   if row["Result"] == 1:
      dfnew.loc[dfnew["Player"] ==
          row["Player1"], "Wins"] += 1
   else:
      dfnew.loc[dfnew["Player"] ==
          row["Player2"], "Wins"] += 1
dfnew = dfnew.sort_values("Wins", ascending =
    False)
dfnew.reset_index(inplace = True)
dfnew.drop("index", axis = 1, inplace = True)
```

TABLE II: Table of Matches won by each player

Player	Wins
M.Bartoli	7
S.Lisicki	6
A.Radwanska	5
K.Flipkens	5
N.Li	4
T.Pironkova	3
S.Williams	3
P.Cetkovska	3
S.Stephens	3
R.Vinci	3

Here I picked the winner of the tournament: M.Bartoli for the analysis. Then we create a list of stats for that given player for all their matches.

```
FNL, FSP, FSW, SSP, SSW, ACE, DBF, WNR, UFE,
    BPC, BPW, NPA, NPW = [], [], [], [], [],
    [], [], [], [], [], [], [], []
for i in range(len(wimW.index)):
  row = wimW.iloc[i]
  if ((newdf["Player"][0] == row["Player1"])
      and (row["Result"] == 1)):
    FNL.append(row["FNL.1"])
    FSP.append(row["FSP.1"])
    FSW.append(row["FSW.1"])
    SSP.append(row["SSP.1"])
    SSW.append(row["SSW.1"])
    ACE.append(row["ACE.1"])
    DBF.append(row["DBF.1"])
    WNR.append(row["WNR.1"])
    UFE.append(row["UFE.1"])
    BPC.append(row["BPC.1"])
    BPW.append(row["BPW.1"])
    NPA.append(row["NPA.1"])
    NPW.append(row["NPW.1"])

  if ((newdf["Player"][0] == row["Player2"])
      and (row["Result"] == 0)):
    FNL.append(row["FNL.2"])
    FSP.append(row["FSP.2"])
    FSW.append(row["FSW.2"])
    SSP.append(row["SSP.2"])
    SSW.append(row["SSW.2"])
    ACE.append(row["ACE.2"])
    DBF.append(row["DBF.2"])
    WNR.append(row["WNR.2"])
    UFE.append(row["UFE.2"])
    BPC.append(row["BPC.2"])
    BPW.append(row["BPW.2"])
    NPA.append(row["NPA.2"])
    NPW.append(row["NPW.2"])

>>
[2, 2, 2, 2, 2, 2, 2]
[58, 64, 62, 61, 58, 61, 67]
[29, 38, 26, 26, 26, 21, 31]
[42, 36, 38, 39, 42, 39, 33]
[16, 11, 10, 7, 15, 10, 7]
[4.0, 1.0, 1.0, 1.0, 0.0, 5.0, 2.0]
[3, 6, 5, 7, 3, 3, 6]
[30, 22, 20, 12, 6, 23, 15]
[25, 17, 12, 13, 12, 10, 14]
[7, 12, 8, 7, 12, 7, 13]
[3, 5, 6, 5, 6, 5, 5]
[18, 16, 8, 2, 10, 11, 11]
[13, 12, 7, 2, 4, 11, 9]
```

All we have to do is take the mean of all values in each column leaving the last entry to get the average stats in previous rounds. And the last values become the stats for the final match.

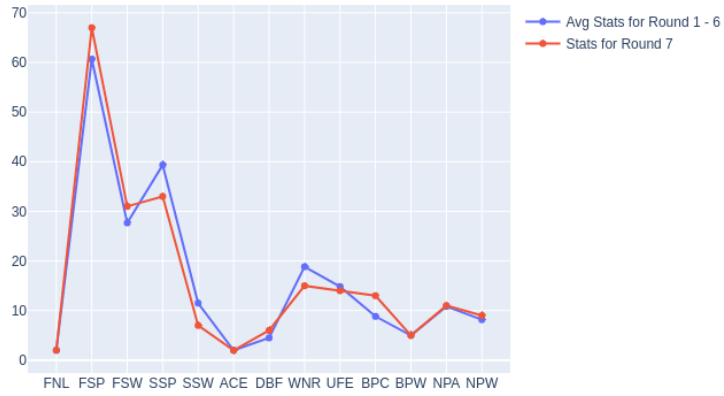

Fig. 15: Line Plot for stats in each round for M. Bartoli.

V. Summary of the Observations

1) The first serve percentage follows Gaussian distribution for both P1 and P2, and both distributions are highly identical. This implies that this feature does not qualify as for a good judging criteria among the two players.
2) Quite surprisingly, there does not seem to be any good correlation between FSW and FSP despite having a common name. The reason for it has been discussed in detail above.
3) The series **first serve percentage(FSP)** and **second serve percentage(SSP)**, and the series **net points attempted(NPA)** and **net points won(NPW)** have strong correlations. Hence in each pair, one of them is redundant.
4) In any scenario, the breaking points created and won are favourable towards the winner of the match.
5) We can train a Classifier model on this dataset. Gaussian Naive Bayes is the classifier used and it provided a high accuracy of **0.9049230769230769**.
6) The features can have their dimensionality reduced by applying **Principal Component Analysis(PCA)**. Then with the reduced features we can apply clustering using the **KMeans** model.
7) Based upon the breaking points won feature, the top performer of the tournament has a better performance than the rest. The top one having an average of **5.166666666666667** while the rest **5.027777777777778**.
8) A Table of content having list of players against their total number of wins can be created. The code can act as a replicator for many further analysis
9) Taking the average of all previous rounds, we can plot a line chart to see that the player has a better performance in terms of all features during the final when compared to rest of the matches.

REFERENCES

[1] Jauhari,Shruti, Morankar,Aniket Fokoue,Ernest. (2014). Tennis Major Tournament Match Statistics. UCI Machine Learning Repository. https://doi.org/10.24432/C54C7K.

[2] pandas documentation — pandas 1.5.3 documentation. "Pandas Documentation — Pandas 1.5.3 Documentation," n.d. https://pandas.pydata.org/docs/.

[3] NumPy Documentation. "NumPy Documentation," n.d. https://numpy.org/doc/.

[4] seaborn: statistical data visualization — seaborn 0.12.2 documentation. "Seaborn: Statistical Data Visualization — Seaborn 0.12.2 Documentation," n.d. https://seaborn.pydata.org/.

[5] Matplotlib documentation — Matplotlib 3.7.0 documentation. "Matplotlib Documentation — Matplotlib 3.7.0 Documentation," n.d. https://matplotlib.org/stable/index.html.

[6] scikit-learn: machine learning in Python mdash; scikit-learn 1.2.2 documentation. "Scikit-Learn: Machine Learning in Python mdash; Scikit-Learn 1.2.2 Documentation," n.d. https://scikit-learn.org/stable/.

[7] Plotly Python Graphing Library. "Plotly," n.d. https://plotly.com/python/.

ACKNOWLEDGMENT

I would like to express our sincere gratitude to the Shruti Jauhari, Aniket Morankar, and Ernest Fokouefor for providing the datasets used in this data narrative. Without their efforts in collecting and compiling this valuable information, this study would not have been possible.

I would also like to thank the UC Irvine Machine Learning Repository for making these datasets available to the public. Their commitment to promoting open access to data is greatly appreciated.

Finally, I would like to thank Prof. Shanmuganathan Raman, without the guidance and supervision of whom, this data narrative would not have achieved the set bar of data mining as required for such a useful data set.