notebook

April 4, 2025

File to import, clean, and analyze data. Can then move the code to a different file for proper documentation purposes.

```
[88]: # Import libraries
      import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
      import seaborn as sns
      import statsmodels.api as sm
[89]: # URL for the dataset
      url = "https://github.com/JeffSackmann/tennis_atp/blob/master"
      # Load the dataset, going from 2004 to 2024 (picking only atp_matches_years.csv)
      years = list(range(2004, 2025))
      matches = []
      for year in years:
          file_url = f"{url}/atp_matches_{year}.csv?raw=true"
          df = pd.read_csv(file_url)
          matches.append(df)
      # Concatenate all dataframes into one
      matches_df = pd.concat(matches, ignore_index=True)
      # Drop duplicates
      # matches_df.drop_duplicates(inplace=True)
      # Reset index
      matches_df.reset_index(drop=True, inplace=True)
      # # Display the first few rows of the dataframe
      # print(matches_df.head())
      # # Display the shape of the dataframe
      # print(matches_df.shape)
      # # Display the columns of the dataframe
      # print(matches_df.columns)
      # # Display the data types of the columns
      # print(matches_df.dtypes)
      # # Display the summary statistics of the dataframe
      # print(matches_df.describe())
```

```
[90]: print(matches_df["surface"].unique())
```

['Hard' 'Clay' 'Grass' 'Carpet' nan]

G is Grand Slam F is Finals M is Masters A is Other D is Davis Cup O is Null?

```
[91]: # Quantify the data types of the columns
      # matches_df["surface"] = matches_df["surface"].map(
            {
                "Clay": 1,
      #
                "Grass": 2,
                "Hard": 3,
      #
      #
                "Carpet": 4,
      # ) # Convert surface to numeric
      matches_df["tourney_level"] = matches_df["tourney_level"].map(
          {
              "G": 1,
              "M": 2,
              "F": 3,
              "A": 4,
              "C": 5,
              "D": 6,
              "S": 7,
              "T": 8,
              "X": 9,
          }
      ) # Convert tourney_level to numeric
      matches_df["round"] = matches_df[
          "round"
      ].map( # Convert round to numeric (num of players left)
          {
              "R64": 64,
              "R32": 32,
              "R16": 16,
              "QF": 8,
              "SF": 4,
              "F": 2,
              "R128": 128,
              "RR": 1, # Round Robin
              "BR": 3, # Best
              "ER": 0,
         }
      ) # Early Round
      matches_df["winner_hand"] = matches_df["winner_hand"].map(
          {
```

```
"L": 1,
        "R": 2,
        "U": 3,
        "A": 4,
    }
) # Convert winner_hand to numeric
matches_df["loser_hand"] = matches_df["loser_hand"].map(
    {
        "L": 1,
        "R": 2,
        "U": 3,
        "A": 4,
    }
) # Convert loser_hand to numeric
# Drop all NA values for the surface column
matches_df.dropna(subset=["surface"], inplace=True)
# Convert the categorical data into numerical data (Ex. Male/Female to 0/1)
matches_df = pd.get_dummies(matches_df, columns=["surface"])
# Turn true/false into 1/0
\# matches_df = matches_df[
    ["surface_Hard", "surface_Clay", "surface_Carpet", "surface_Grass"]
# ].astype(int)
```

[92]: print(matches_df.info())

<class 'pandas.core.frame.DataFrame'>
Index: 61737 entries, 0 to 61789
Data columns (total 52 columns):

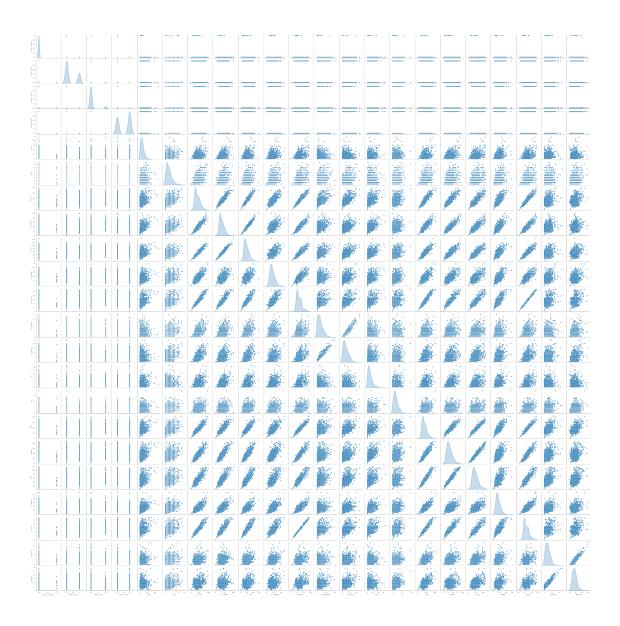
#	Column	Non-Null Count	Dtype
0	tourney_id	61737 non-null	object
1	tourney_name	61737 non-null	object
2	draw_size	61737 non-null	int64
3	tourney_level	61673 non-null	float64
4	tourney_date	61737 non-null	int64
5	match_num	61737 non-null	int64
6	winner_id	61737 non-null	int64
7	winner_seed	26160 non-null	float64
8	winner_entry	7974 non-null	object
9	winner_name	61737 non-null	object
10	winner_hand	61737 non-null	int64
11	winner_ht	60722 non-null	float64
12	winner_ioc	61737 non-null	object
13	winner_age	61734 non-null	float64
14	loser_id	61737 non-null	int64

```
loser_entry
                              12883 non-null
                                              object
      17
          loser_name
                              61737 non-null
                                              object
      18
         loser_hand
                              61733 non-null
                                              float64
      19
          loser ht
                              59582 non-null float64
      20
          loser ioc
                              61737 non-null
                                              object
          loser age
                              61735 non-null float64
      22
          score
                              61737 non-null
                                              object
      23
                              61737 non-null int64
         best of
      24
         round
                              61737 non-null
                                              int64
      25
         minutes
                              55186 non-null float64
      26
                              56827 non-null float64
         w_ace
      27
                              56827 non-null float64
          w_df
      28
         w_svpt
                              56827 non-null float64
      29 w_1stIn
                              56827 non-null
                                              float64
                              56827 non-null float64
         w_1stVon
      31
         w_2ndWon
                              56827 non-null float64
      32 w_SvGms
                              56827 non-null float64
      33
         w_bpSaved
                              56827 non-null float64
      34
         w bpFaced
                              56827 non-null float64
         l ace
      35
                              56827 non-null float64
      36 1 df
                              56827 non-null float64
         l_svpt
                              56827 non-null float64
                              56827 non-null float64
      38 l 1stIn
      39 l_1stWon
                              56827 non-null float64
                              56827 non-null float64
      40 \quad 1_2ndWon
         1_SvGms
                              56827 non-null float64
      41
      42
         l_bpSaved
                              56827 non-null float64
      43
                              56827 non-null float64
          1_bpFaced
         winner_rank
                              61345 non-null float64
          winner_rank_points
                              61345 non-null float64
      46
          loser_rank
                              60659 non-null float64
      47
          loser_rank_points
                              60659 non-null float64
         surface_Carpet
                              61737 non-null bool
      48
          surface Clay
      49
                              61737 non-null bool
      50
          surface Grass
                              61737 non-null bool
      51 surface Hard
                              61737 non-null bool
     dtypes: bool(4), float64(31), int64(8), object(9)
     memory usage: 23.3+ MB
     None
[93]: metrics_list = [
          "surface_Carpet",
          "surface_Clay",
          "surface_Grass",
          "surface_Hard",
          "w_ace",
```

14467 non-null float64

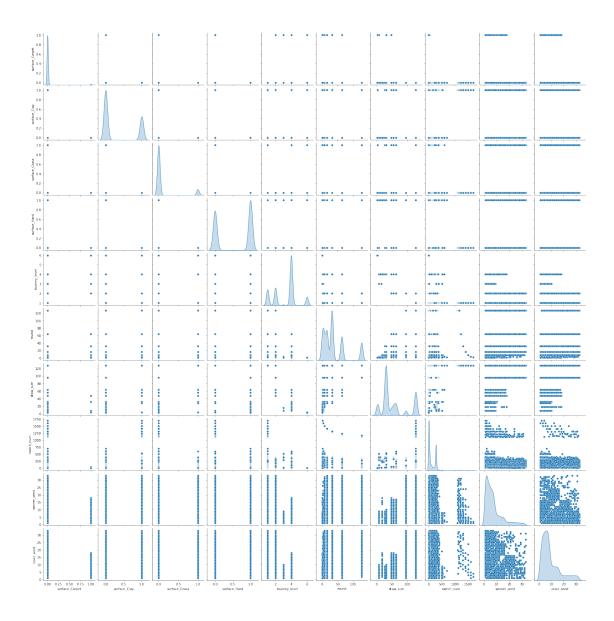
15 loser_seed

```
"w_df",
          "w_svpt",
          "w_1stWon",
          "w_1stIn",
          "w_2ndWon",
          "w_SvGms",
          "w_bpSaved",
          "w_bpFaced",
          "l_ace",
          "l_df",
          "l_svpt",
          "l_1stWon",
          "l_1stIn",
          "1_2ndWon",
          "l_SvGms",
          "1_bpSaved",
          "l_bpFaced",
      ]
      general_list = [
          "surface_Carpet",
          "surface_Clay",
          "surface_Grass",
          "surface_Hard",
          "tourney_level",
          "round",
          "draw_size",
          "match_num",
          "winner_seed",
          "loser_seed",
      ]
      metrics_df = matches_df[metrics_list].copy()
      general_df = matches_df[general_list].copy()
[94]: # Choose randomly 1000 rows from the dataframe
      sample_metrics_df = metrics_df.sample(n=1000, random_state=1)
      # Pairplot of the metrics data
      sns.pairplot(sample_metrics_df, diag_kind="kde")
      plt.savefig("pairplot_metrics.png")
      plt.show()
```



```
[95]: # Choose randomly 1000 rows from the dataframe
# general_df = general_df.sample(n=10000, random_state=1)

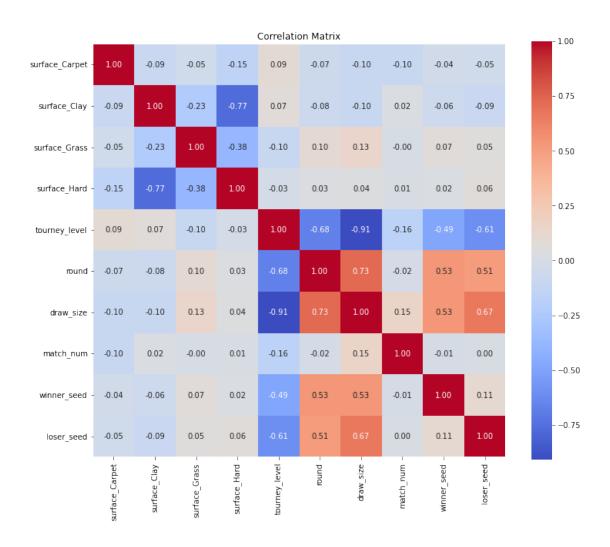
# Pairplot of the metrics data
sns.pairplot(general_df, diag_kind="kde")
plt.savefig("pairplot_metrics.png")
plt.show()
```



```
[96]: # Correlation matrix
    corr = metrics_df.corr()
    # Plot the correlation matrix
    plt.figure(figsize=(12, 10))
    sns.heatmap(corr, annot=True, fmt=".2f", cmap="coolwarm", square=True)
    plt.title("Correlation Matrix")
    plt.savefig("correlation_matrix.png")
    plt.show()
```

```
Correlation Matrix
                                                                                                                                   1.0
surface Clay -0.09100-0.230.77-0.280.08-0.04-0.09-0.02-0.07-0.070.06 0.09-0.240.09-0.05-0.09-0.03-0.08-0.070.05 0.10
surface Grass -0.05-0.23100 -0.38 0.19 0.06 0.11 0.17 0.13 0.08 0.15-0.00-0.20.15 0.07 0.13 0.17 0.14 0.09 0.15 0.02 -0.01
                                                                                                                                  - 0.8
 surface Hard -0.15-0.77-0.38 1.00 0.14 0.04 -0.03-0.02-0.06 0.02-0.05-0.06 0.12 0.04 -0.03-0.02-0.06 0.02-0.05-0.06
        w ace -0.02 -0.28 0.19 0.14 1.00 0.24 0.37 0.49 0.35 0.31 0.43 0.06 0.03 0.31 0.12 0.38 0.45 0.38 0.37 0.42 0.06 -0.03
                                                                                                                                   - 0.6
          w df -0.02-0.080.06 0.04 0.24 1.00 0.49 0.37 0.37 0.41 0.44 0.34 0.40 0.19 0.24 0.42 0.37 0.37 0.38 0.43 0.18 0.21
       w_svpt -0.03-0.040.11-0.03 0.37 0.49 1.00 0.91 0.95 0.78 0.94 0.61 0.69 0.48 0.33 0.90 0.86 0.85 0.76 0.94 0.34 0.35
    w 1stWon -0.01-0.09 0.17-0.02 0.49 0.37 0.91 1.00 0.96 0.56 0.91 0.46 0.50 0.53 0.28 0.85 0.87 0.83 0.73 0.90 0.29 0.25
                                                                                                                                   0.4
       w 1stin -0.02-0.020.13-0.060.35 0.37 0.95 0.96 1.00 0.58 0.90 0.56 0.63 0.47 0.29 0.85 0.84 0.82 0.71 0.89 0.32 0.32
    w 2ndWon -0.02-0.07 0.08 0.02 0.31 0.41 0.78 0.56 0.58 1.00 0.76 0.42 0.46 0.43 0.30 0.72 0.70 0.67 0.65 0.76 0.26 0.25
                                                                                                                                  - 0.2
     w SvGms -0.02-0.070.15-0.020.43 0.44 0.94 0.91 0.90 0.76 1.00 0.43 0.54 0.52 0.34 0.93 0.90 0.88 0.79 0.99 0.36 0.37
   w bpSaved -0.030.06-0.00-0.050.06 0.34 0.61 0.46 0.56 0.42 0.43 1.00 0.94 0.13 0.17 0.42 0.36 0.38 0.34 0.43 0.20 0.25
   w bpFaced ~0.030.09-0.02-0.060.03 0.40 0.69 0.50 0.63 0.46 0.54 0.94 1.00 0.13 0.23 0.53 0.43 0.49 0.41 0.55 0.28 0.38
         Lace -0.02 -0.24 0.15 0.12 0.31 0.19 0.48 0.53 0.47 0.43 0.52 0.13 0.13 1.00 0.25 0.48 0.58 0.45 0.43 0.52 0.11 -0.00
          l df -0.02-0.09 0.07 0.04 0.12 0.24 0.33 0.28 0.29 0.30 0.34 0.17 0.23 0.25 1.00 0.39 0.29 0.26 0.35 0.35 0.25 0.29
                                                                                                                                  - -0.2
        l svpt -0.03-0.05 0.13-0.03 0.38 0.42 0.90 0.85 0.85 0.72 0.93 0.42 0.53 0.48 0.39 1.00 0.92 0.94 0.81 0.94 0.55 0.53
     I 1stWon -0.01-0.090.17-0.020.45 0.37 0.86 0.87 0.84 0.70 0.90 0.36 0.43 0.58 0.29 0.92 1.00 0.95 0.65 0.91 0.41 0.32
       I 1stin -0.03-0.03 0.14 -0.06 0.38 0.37 0.85 0.83 0.82 0.67 0.88 0.38 0.49 0.45 0.26 0.94 0.95 1.00 0.63 0.90 0.50 0.48
                                                                                                                                   -0.4
    I_2ndWon -0.01-0.080.09 0.02 0.37 0.38 0.76 0.73 0.71 0.65 0.79 0.34 0.41 0.43 0.35 0.81 0.65 0.63 1.00 0.79 0.38 0.31
     I SvGms -0.02-0.070.15-0.02 0.42 0.43 0.94 0.90 0.89 0.76 0.99 0.43 0.55 0.52 0.35 0.94 0.91 0.90 0.79 1.00 0.37 0.39
                                                                                                                                   -0.6
    l bpSaved -0.030.05 0.02 -0.05 0.06 0.18 0.34 0.29 0.32 0.26 0.36 0.20 0.28 0.11 0.25 0.55 0.41 0.50 0.38 0.37 1.00 0.92
    bpFaced -0.040.10-0.01-0.08-0.030.21 0.35 0.25 0.32 0.25 0.37 0.25 0.38 0.000.29 0.53 0.32 0.48 0.31 0.39 0.92 1.00
                         urface_Grass
```

```
[97]: # Correlation matrix
    corr = general_df.corr()
    # Plot the correlation matrix
    plt.figure(figsize=(12, 10))
    sns.heatmap(corr, annot=True, fmt=".2f", cmap="coolwarm", square=True)
    plt.title("Correlation Matrix")
    plt.savefig("correlation_matrix.png")
    plt.show()
```



```
[98]: # Output all unique values for surface
# print(matches_df['surface'].unique())

# Separate the data by surface
hard_df = matches_df["surface_Hard"]
clay_df = matches_df["surface_Clay"]
grass_df = matches_df["surface_Grass"]
carpet_df = matches_df["surface_Carpet"]

print(hard_df.info())
```

dtypes: bool(1)

memory usage: 542.6 KB

None