From Exploratory Data Analysis to Machine Learning: Continuing My Python Journey

Leon Rod Davoody

PharmaSUG 2026 – Paper DV-314

ABSTRACT

This paper builds on my previous work using the Summer Olympics 2016 dataset and shows how Python can be used not just for exploring data, but also for machine learning and making predictions. After learning basic data analysis in my earlier paper, I now use more advanced tools like seaborn and matplotlib for visualization, and machine learning models like Random Forest and Decision Trees.

The goal was to predict an athlete's sport based on physical features like age, height, and weight. The results showed that height is the most important factor, with weight also playing a big role. I also looked at patterns in medal performance across different countries and sports like basketball, football, and water polo.

Overall, this project shows how Python can be used to move from simple data analysis to more advanced predictions, while still being understandable for beginners. All code is included so others can follow along.

INTRODUCTION

When I wrote my first paper in 6th grade, I was mainly learning how to read datasets, calculate simple statistics, and make basic graphs. At that time, Python felt like a tool for answering questions I already had.

Now, I've learned that Python can also help find new questions by discovering patterns in data and making predictions. This paper shows my progress from basic data analysis to machine learning.

Using the same Summer Olympics 2016 dataset, I explored three main questions:

1. Can we predict an athlete's sport using their physical features?
2. Are there patterns between physical traits and success in sports?
3. Which countries perform best in certain sports, and does home advantage matter?

Machine learning is a type of artificial intelligence where computers learn patterns from data instead of being told exactly what to do. Instead of writing rules, we give the model examples and let it figure things out.

In this project, I used two models:

● Random Forest (more powerful, less easy to understand)
● Decision Tree (simpler and easier to explain)

What Is Machine Learning?

Machine learning is a type of artificial intelligence where computers learn patterns from data instead of being directly told what to do. Instead of writing step-by-step rules, we give the model examples and let it figure out how to make predictions on its own.

In this project, I used supervised learning, which means the model is trained using labeled data. For example, the dataset already tells us which sport each athlete plays, so the model learns how physical features relate to that sport.

I used two main models:

● Random Forest, which combines many decision trees to improve accuracy ● Decision Tree, which creates simple rules that are easy to understand

Both models take inputs like age, height, and weight, and try to predict the athlete's sport.

DATASET AND ENVIRONMENT

The Summer Olympics 2016 Dataset

The dataset comes from the 2016 Summer Olympics and includes over 11,000 athlete records. It has information like age, height, weight, sport, country, and medal results.

I used the same dataset as my previous paper so I could compare what I can do now with machine learning versus what I could do before with just basic analysis.

For coding, I used Python with tools like:

● scikit-learn (machine learning)
● seaborn (visualization)
● matplotlib (graphs)
● Jupyter Notebook (for easier testing and visualization)

Development Environment

I continued using Anaconda with Spyder for running scripts, but I also started using Jupyter Notebooks for exploratory work. Jupyter Notebooks made it easier to work with the data because I could see the graphs and results right away, which helped me test ideas faster.

In addition to the tools I used in my previous paper, I also used the following Python libraries:

- **scikit-learn** for machine learning models and evaluation
- **seaborn** for better data visualization built on top of matplotlib
- **plotly** for creating interactive charts during exploration

EXAMPLE 1 — CORRELATION HEATMAP

Motivation

Before using machine learning, it's important to understand how the data is related. A correlation heatmap shows how strongly variables are connected to each other.

I focused on age, height, and weight to see if any of them were redundant or especially important.

Code

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

# Load and clean dataset
path = "SummerSports2016.xlsx"
data = pd.read_excel(path)
data_clean = data.dropna(subset=['Age', 'Height', 'Weight', 'Sport'])

# Select numeric columns
numeric_data = data_clean[['Age', 'Height', 'Weight']]

# Create correlation heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(numeric_data.corr(), annot=True, cmap='coolwarm', center=0)
plt.title("Correlation Between Physical Attributes")
plt.tight_layout()
plt.show()
```

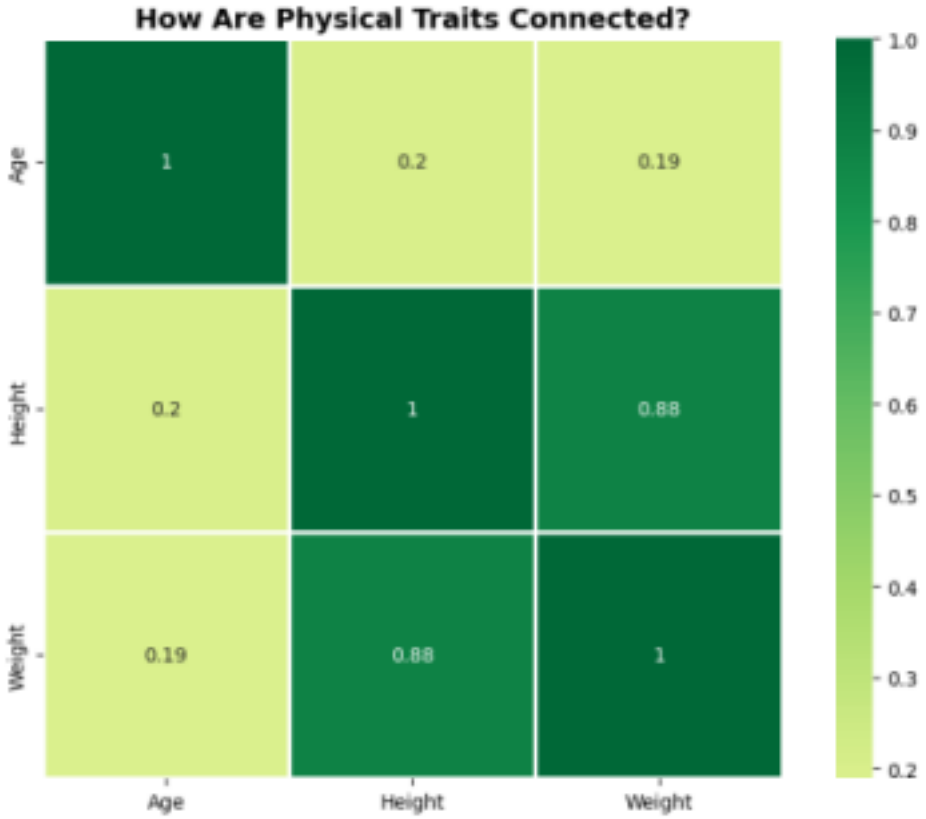

Figure 1. Correlation heatmap of Age, Height, and Weight for Olympic athletes (2016).

Results and Interpretation

The heatmap showed that height and weight are strongly related, which makes sense because taller people usually weigh more.

Age, however, had very little relationship with either height or weight. This means athletes of many different ages can have similar body types.

This result is important because it shows that height and weight both provide useful information, and neither one should be removed.

EXAMPLE 2 — HEIGHT DISTRIBUTION BY SPORT

Motivation

After looking at general relationships, I wanted to see how physical traits differ between sports. I used a box plot to compare athlete heights across basketball, football, and water polo.

Code

```
# Filter to three sports of interest
sports = ['Basketball', 'Football', 'Water Polo']
sport_data = data_clean[data_clean['Sport'].isin(sports)]

# Create box plot
plt.figure(figsize=(10, 6))
sns.boxplot(x='Sport', y='Height', data=sport_data, palette='Set2')
plt.title('Height Distribution by Sport')
plt.ylabel('Height (cm)')
plt.tight_layout()
plt.show()

# Print descriptive statistics
stats = sport_data.groupby("Sport")["Height"].agg(
 Mean="mean", Min="min", Max="max", Median="median"
).round(1)
print(stats)
```

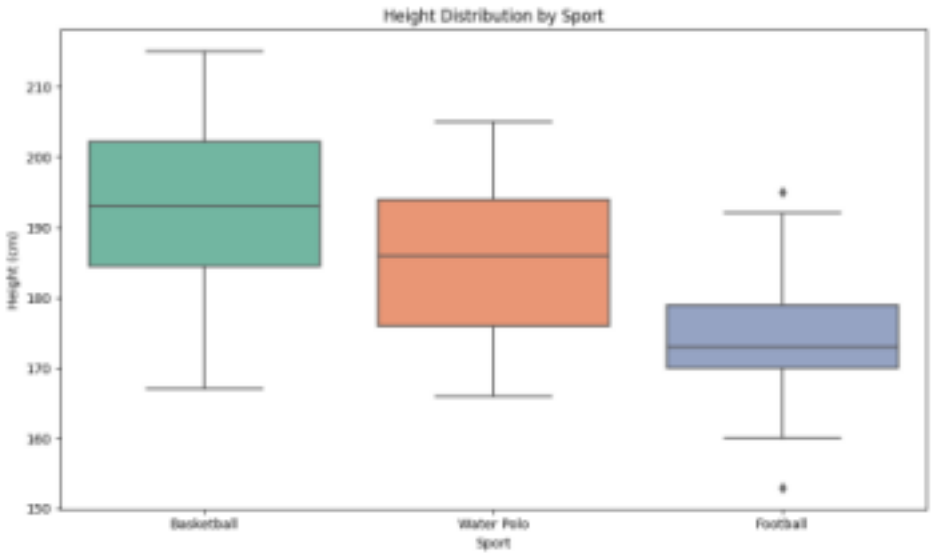

Figure 2. Height distributions for basketball, football, and water polo athletes.

Basketball	195.2	175	215	196.0
Football	174.8	153	193	175.0
Water Polo	184.5	166	203	184.0

Table 1. Descriptive height statistics by sport (2016 Summer Olympics).

Results and Interpretation

The results clearly showed differences between sports:

- Basketball players are the tallest
- Water polo players are in the middle
- Football players are the shortest

This matches what we would expect based on how each sport is played.

This also helped confirm that height would be a strong feature for predicting sport in the machine learning models.

EXAMPLE 3 — FEATURE IMPORTANCE WITH RANDOM FOREST

Motivation

The graphs suggested that height was important, but I wanted a more exact way to measure how much each feature mattered. Random Forest provides a feature importance score for each variable.

Code

```
from sklearn.ensemble import RandomForestClassifier

# Filter and prepare data
sports = ['Basketball', 'Football', 'Water Polo']
ml_data = data_clean[data_clean['Sport'].isin(sports)]

X = ml_data[['Age', 'Height', 'Weight']]
y = ml_data['Sport']

# Train Random Forest classifier
model = RandomForestClassifier(random_state=42)
model.fit(X, y)

# Extract and plot feature importances
importances = model.feature_importances_
features = ['Age', 'Height', 'Weight']

plt.figure(figsize=(10, 6))
bars = plt.bar(features, importances)
plt.title("Feature Importance for Sport Prediction")
plt.ylabel("Importance Score")
for i, v in enumerate(importances):
 plt.text(i, v + 0.01, f"{v:.3f}", ha="center")
plt.tight_layout()
```

```
plt.show()
```

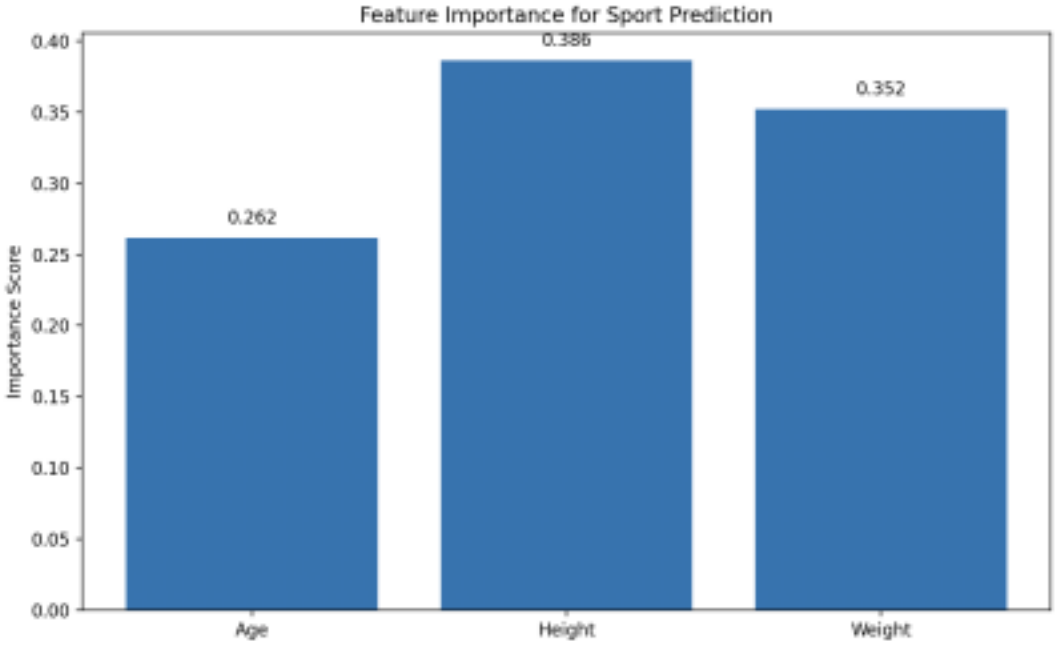

Figure 3. Feature importance scores from the Random Forest classifier (Height = 0.386, Weight = 0.352, Age = 0.262).

Results and Interpretation

The model showed:

- Height = most important
- Weight = second
- Age = third

Even though height was the top factor, all three features contributed to the model. This means the best predictions come from using all of them together.

This also matches the earlier visual results, which makes the findings more reliable.

EXAMPLE 4 — DECISION TREE CLASSIFICATION

Motivation

Random Forest gives strong results but is harder to understand. Decision Trees are simpler and show clear rules, which makes them easier to explain.

Code

```
from sklearn.tree import DecisionTreeClassifier, export_text

# Train decision tree with max depth of 3 for readability
tree = DecisionTreeClassifier(max_depth=3, random_state=42)
tree.fit(X, y)

# Print the learned classification rules
rules = export_text(tree, feature_names=['Age', 'Height', 'Weight'])
print(rules)
```

```
|--- Height <= 189.0
|   |--- Weight <= 75.0
|   |   |--- class: Football
|   |--- Weight > 75.0
|   |   |--- class: Water Polo
|--- Height > 189.0
|   |--- class: Basketball
```

Figure 4. Decision tree output showing learned classification rules (max depth = 3).

Learned Rules and Interpretation

The Decision Tree created simple rules like:

- If height is greater than about 189 cm → likely basketball
- If height is lower but weight is higher → likely water polo
- If both height and weight are lower → likely football

These rules are easy to understand and match real-world expectations. Even though Decision Trees are simpler, they are useful for explaining how predictions are made.

EXAMPLE 5 — MEDAL WINNERS BY COUNTRY AND SPORT

Motivation

I also wanted to see which countries perform best in different sports and whether hosting the Olympics gives an advantage.

Code

```
# Filter to medal winners only
medal_data = data_clean.dropna(subset=['Medal'])
medal_data = medal_data[medal_data['Sport'].isin(sports)]

# Count medal winners by sport and team
medal_count = (
 medal_data.groupby(['Sport', 'Team'])
 .size()
 .reset_index(name='Count')
)

top_sports = medal_count.sort_values('Count', ascending=False).head(15)

# Color by sport
colors_map = {
 'Basketball': '#FF6B6B',
 'Football': '#45B7D1',
 'Water Polo': '#4ECDC4'
}
colors = [colors_map[s] for s in top_sports['Sport']]

# Plot
plt.figure(figsize=(12, 8))
plt.barh(range(len(top_sports)), top_sports['Count'],
 color=colors, edgecolor='black', linewidth=1.5)
plt.yticks(range(len(top_sports)),
 [f"{row['Team']} ({row['Sport']})" for _, row in top_sports.iterrows()])
plt.xlabel('Number of Medal Winners', fontsize=12)
plt.title('Top Medal-Winning Teams by Sport', fontsize=14, fontweight='bold')
plt.grid(axis='x', alpha=0.3)
plt.tight_layout()
plt.gca().invert_yaxis()
plt.show()
```

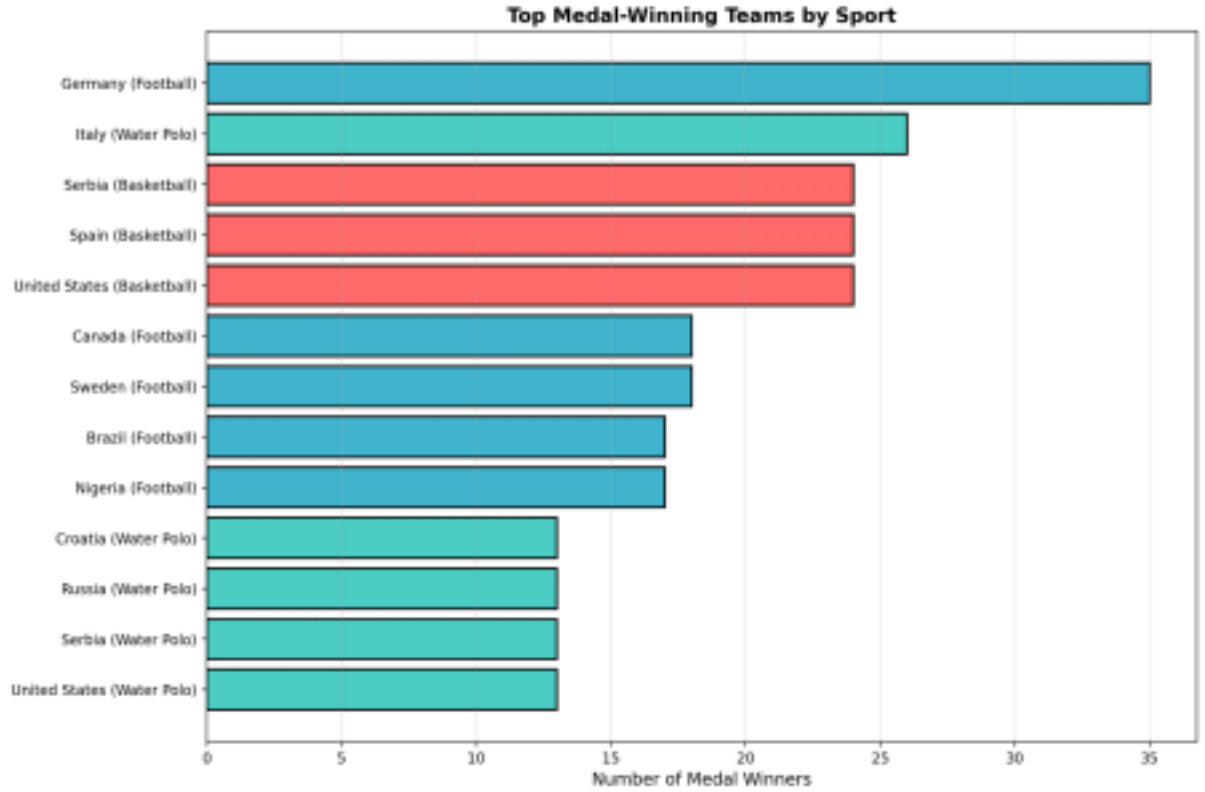

Figure 5. Top 15 medal-winning teams by sport at the 2016 Summer Olympics.

Basketball	Serbia	24
Basketball	Spain	24
Basketball	United States	24
Football	Brazil	17
Football	Germany	35
Football	Canada	18
Water Polo	Croatia	13
Water Polo	Italy	26

Table 2. Top medal-winning teams across basketball, football, and water polo.

Results and Interpretation

The results showed clear patterns:

- The United States, Spain, and Serbia performed strongly in basketball
- Brazil performed very well in football, especially as the host country
- Italy and Croatia stood out in water polo

Brazil's success in football supports the idea of home advantage, where teams perform better when competing in their own country.

Overall, this shows that both resources and location can impact success in sports.

CHALLENGES AND SOLUTIONS

Challenge 1: Missing Data

The dataset had some missing values for height, weight, and age. To handle this, I removed any rows that were missing these values using dropna().

This approach helps keep the data accurate and avoids adding incorrect values, but it also slightly reduces the size of the dataset.

Challenge 2: Multi-class Classification

The Random Forest model in scikit-learn already supports multi-class classification, so I did not need to do anything extra for that.

However, one issue is that some sports have more athletes than others, which can cause the model to favor the larger groups. This problem was not fully addressed in this paper, but it could be improved in the future using methods like stratified sampling or class weighting.

Challenge 3: Interpretability vs. Accuracy

Decision Trees are easy to understand but can become too complex if they grow too much. To prevent this, I limited the tree to a maximum depth of 3, which made the rules easier to read.

Random Forest models are usually more accurate, but they are harder to interpret. To help with this, I used feature importance scores to better understand how the model was making predictions.

CONCLUSION

This paper shows how Python can be used to move from simple data analysis to machine learning and prediction.

The results showed that height and weight are strong predictors of an athlete's sport, and machine learning models can use these features to make accurate predictions.

It also showed that countries specialize in certain sports and that home advantage can play a role in performance.

Overall, machine learning helps uncover patterns that are not obvious from basic analysis. This project reflects my growth in Python and shows how powerful these tools can be.

REFERENCES

Davoody, L. R. (2023). From Data Access to Exploratory Data Analysis — My Journey into the World of Python. *Paper 184-2023, Western Users of SAS Software (WUSS) Conference Proceedings.*

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R.* Springer.

Kaggle. (2016). *Rio 2016 Olympic Games Dataset.* Retrieved from https://www.kaggle.com/datasets/rio2016/olympic-games

Matthes, E. (2019). *Python Crash Course: A Hands-On, Project-Based Introduction to Programming* (2nd ed.). No Starch Press.

McKinney, W. (2022). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter* (3rd ed.). O'Reilly Media.

Nevill, A. M., & Holder, R. L. (1999). Home advantage in sport: An overview of studies on the advantage of playing at home. *Sports Medicine, 28*(4), 221–236.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Plotly Technologies Inc. (2015). *Collaborative data science.* Retrieved from https://plot.ly

Python Software Foundation. (2024). *Python Documentation.* Retrieved from https://docs.python.org/3/

Waskom, M. L. (2021). seaborn: Statistical data visualization. *Journal of Open Source Software, 6*(60), 3021. https://doi.org/10.21105/joss.03021

CONTACT INFORMATION

Comments and questions are welcomed. Please contact the author:

Leon Rod Davoody | lrdavoody@gmail.com

AUTHOR BIOGRAPHY

Leon Rod Davoody is currently in 8th grade and has been programming in Python for four years. After publishing his first paper at WUSS 2023, he became more involved in applying his skills to real-world projects and joined his school's robotics team, where he contributed to building and programming autonomous systems.

In addition to his work in robotics, Leon explores data analysis and machine learning through independent projects. He is especially interested in how software and AI can be used to solve practical problems and improve performance in different fields.