Project 3: Data Manipulation with Pandas

John Wesley Mathis

Dr. Anthony Choi

June 08, 2024

ECE/SSE 591, Summer 2024

Table of Contents:

[Deliverable Table 4](#__RefHeading___Toc2569_331549091)

[1. Introduction 5](#__RefHeading___Toc2573_331549091)

[2. Top Movie Data Analysis 6](#__RefHeading___Toc2577_331549091)

[3. Conclusion 7](#__RefHeading___Toc2609_331549091)

[References 8](#__RefHeading___Toc3692_331549091)

Table of Figures:

# Deliverable Table

The purpose of this table is to provide a complete view of the concepts covered in chapter 3 of *"Python Data Science Handbook"* (VanderPlas, 2016) and provide a general page location for where the topic was demonstrated.

|  |  |
| --- | --- |
| Deliverables | Location |
| Introducing Pandas Objects |  |
| Data Indexing and Selection |  |
| Operating on Data in Pandas |  |
| Handling Missing Data |  |
| Hierarchical Indexing |  |
| Combining Datasets: Concat and Append |  |
| Combining Datasets: Merge and Join |  |
| Aggregation and Grouping |  |
| Pivot Tables |  |
| Vectorized String Operations |  |
| Working with Time Series |  |
| High-Performance Pandas: eval() and query() |  |

# 1. Introduction

Python has a rich repository of libraries that aid scientists and researchers in data analysis and manipulation. One of the most common libraries in use is Pandas, which is built on top of NumPy and provides a higher-level, and more flexible interface for data handling. While NumPy excels at efficient numerical computations with arrays, Pandas introduces data structures like Series and DataFrame that offer a more intuitive means to work with structured data.

Because of Pandas’ Series and DataFrame objects, data scientists have an indispensable tool to handle, clean and manipulate data in tabular form. These objects support a wide range of operations, from simple data aggregation and filtering to complex time-series analysis. The library’s ability to handle missing data, merge datasets, and perform group-by operations adds significant value to Python’s data manipulation kit.

This report aims to demonstrate my proficiency in Python data manipulation techniques as covered in Chapter 3 of the “Python Data Science Handbook” by Jake VanderPlas (2016). This report attempts to illustrate the core concepts and functionalities of the Pandas library. The code presented in this report was developed using Visual Studio Code with Jupyter Notebook extensions. I will provide detailed explanations, highlighting key features and operations that make Pandas an essential tool for data analysis.

# 2. Top Movie Data Analysis

I chose the IMDB Top 1000 movies dataset to analyze. This dataset contains information about the top 1000 movies(year). After importing the dataset, I viewed the first few rows by outputting it to the screen using*`head()`.* Figure 1 below shows the code and output. The dataset includes a poser link, series title, released year, certificate, runtime, genre, rating, and much more.

Fig

Using *`describe()`* I obtained summary statistics of the numerical columns. There are only three numerical columns, IMDB rating, meta score, and number of votes. For this particular data set this doesn’t tell me to much. However, we find out that the average IMDB rating is a 7.9. Howeeverr, because we don’t know to much about the data we don’t understand how skewed this information may be. Figure ## shows the output.

Fig.

Using `*info()`* I obtained general information about the DataFrame. From this view, I learned the column names, the data type for each column, and if each column contained all the information. The certificate, Meta Score, and Gross was missing some data. Figure ## shows the code and output.

Fig

Because the data seems to be un-ranked, just simply the top 1000 movies in an arbitraray order, I change the index from the basic 0-999 using the *`set\_index()`* method. I changed the index to be the series title. Figure ### shows this output.

Fig

After viewing the first few rows of data and the last few rows using the *`head()`* and *`tail()`* methods respectively, it seemed that the data was un-ranked. The dataset is simply a list of the top 1000 movies. After reviewing the data info from before (Fig#) I decided to add a rank column(Fig #).

Fig

There are a lot of columns in this dataset, and I wanted to be able to quickly and easily view the ranks. So I retrieved the columns and converted the to a list and stored the information in a variable for access. I removed the string *`Rank`* from the list and inserted it again into the list at position 0. Then I re-ordered the DataFrame using the variable that stored the columns. Figure # shows the output.

Fig

After re-ordering the DataFrame, I used *`sort\_values()`*  using the IMDb rating to sort the movies in ascending order. Then I set the index to be the *Rank* column. Figure # shows the code and output

Fig

From figure ## I already saw that some of the data was missing. So I wanted to clean it up some. So I first checked what columns contain null values using *`isna()`* and *`sum()`* to get number of missing data in each column. Figure ## shows the code and results.

Fig

As I concluded earlier, this further validates that the data missing from the columns is information fromm the Certificate, Meta score, and Gross column. I fill in the missing data using *`fillna()`* and I fill it using 0. Figure # shows the results.

Fig #

I wanted to also check if there are any duplicates in the dataset. Figure # shows the code and output. I used *`duplicated()`* to determine this. The result return 0, meaning there are no duplicated rows.

Fig

There is a release year column so I wanted to turn this into a datetime format. Using *`pd.to\_datetime()`* I attempted to do this. Unfortunately, this only works if it is a full date. This column of data only contains the year. So to demonstrate this functionality, I decided to add the full date to the column. Filling it with the correct year and a placeholder date. Figure ## shows this being done.

Fig

Now I drop any unused coluns using the *`drop()`*  method. Since the poser link is not needed, I drop it. I print the shape of the dataframe to verify that its dropped. Figure ## shows the code and results.

Fig

Next I begin to analyze the dataset. First I determine which year had the most top movies released. I accomplish this by using the *`groupby()`* method and *`count()`* method. I store the results in a variable named *`best\_year`*. From this information I determined that 2014 had the most released movies that were considered top movies. Figure shows this code and output.

Fig

For this project I attempt to begin the programming of a recommendation system for movies. Due to the scope ocf this class and time limitations, only the beginning movie data analysis will be completed. In subsequent projects, there may be an attempt to implement a complete recommendation guide.

Pandas and NumPy were imported into the file and the dataset for movies’ credits and movies’ metadata were read into the python script as csv files, creating two DataFrames.

# 3. Conclusion

This report documents my journey in mastering NumPy which involved grasping fundamental concepts such as arrays, computations using universal functions and broadcasting, incorporating comparisons and boolean logic into arrays, indexing and sorting arrays, and learning how to create structured arrays.

Through the process of coding, many errors were encountered. Due to the simplicity of the projects and exercises, all of the errors were syntax errors. My experience is mainly in C and C++ so I am used to the nuance of adding semicolons to the end of each line. Many times in Python, I found myself accidentally adding semicolons to the ends of blocks and statements, confused why my code block continually showed errors. Other examples of syntax errors I encountered include forget a quotation, forgetting a colon when defining a function or loop, or simply misspelling a variable that I named.

Engaging in mini-projects and working through the exercises has proved to be a valuable source in aiding me to take the theory of Python fundamentals and the NumPy package and put the concepts into practice and think about practical data science applications. This has allowed me to grasp the concepts that make the NumPy package a valuable tool for efficiently managing large sets of data.

# References

1. VanderPlas, J. (*2016*).  *Python Data Science Handbook*. O’Reilly Media. Retrieved from https://jakevdp.github.io/PythonDataScienceHandbook/index.html