AMI23B – Business Intelligence Lab1

Using pandas to Explore a Dataset

Tutorial Lab Instructions – follow the tutorial and answer and report all 10questions (in your Jupyter Notebook).

Topics covered in this tutorial lab:

- Calculate metrics about your data.
- Perform basic queries and aggregations.
- Discover and handle incorrect data, inconsistencies, and missing values.
- Visualise your data with plots.

Remember, these only scratch the surface of the pandas Python library!

Task1: Setting up the Environment.

(The references and links in this tutorial are aimed at a Windows environment – if you are not a Windows user, please find suitable alternatives on your own.)

1. You will need a working Python 3 environment. (Important tip! Add Python Path to Environment Variables. Setting up the Python path to system variablesalleviates the need for using full paths.)

2. You will need to install pip if it is not already installed (or an alternative of your preference). pip is a powerful package management system for Python software packages. (Hint: the newest versions of Python come with pip installedas a default, make sure it is installed!)

3. Run in a Jupyter Notebook for sound outputs. (Mandatory)

4. You will need to install pandas Python Library.

5. You will also need to install Matplotlib. Matplotlib is a comprehensive library for creating static, animated, and interactive visualisations in Python. It is a valuable complement to pandas.

Note: If you are using the Anaconda distribution, you are good to go! Anaconda already comes with the pandas Python library installed (same goes for pip). Click here to check out the Anaconda installation guide.

Great! Now that you are all set up let us get started!

Task2: Using the pandas Python Library.

This tutorial will analyse NBA results provided by FiveThirtyEight in a 17MB CSV file. You could download the CSV file using the web browser; however, having a download script has its advantages.

2.1) Create a download script to download the data.

```
1.  import requests
2.
3.  #get the data from a download link
4.  download_url = "https://raw.githubusercontent.com/fivethirtyeight/data/master/nba-
    elo/nbaallelo.csv"
5.  target_csv_path = "nba_all_elo.csv"
6.
7.  response = requests.get(download_url)
8.  response.raise_for_status()     #check that the request was successful
9.
10. #save the file nba_all_elo.csv in your current working directory
11. with open(target_csv_path, "wb") as f:
12.      f.write(response.content)
13. print("download ready")
```

The script will save the file nba_all_elo.csv in your current working directory when you run it.

2.2) Now create a new script in which you will use the pandas Python library to look at your data.

```
1.  #importing pandas in Python with the pd alias
2.  import pandas as pd
3.
4.  #read in the dataset and store it as a DataFrame object in the variable nba
5.  #when you copy the file path, if you get the FileNotFoundError: [Errno 2]
6.  #then flip the backslash to a forward slash, and voila, it works!
7.  nba = pd.read_csv("C:/Users/sbk/PycharmProjects/Lab2/venv/nba_all_elo.csv")
8.
9.  #check nba's type, it should be a DataFrame
10. type(nba)
```

Note: pandas can handle files in formats other than CSV too!

2.3) Let's see how much data is actually in `nba` (report these findings):

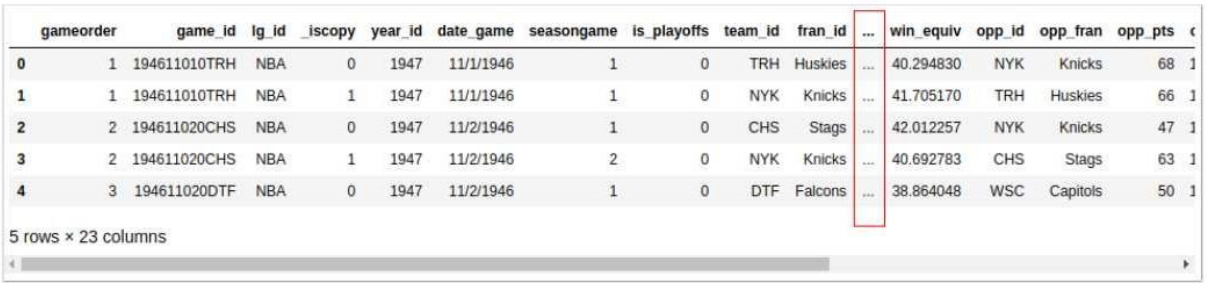
```
1.  #len() determines the number of rows (observations) in a dataset
2.  len(nba)


3.  #.shape determines dimensionality
4.  #the result is a tuple containing number of rows and columns
5.  nba.shape

6.  #take a look at the first five rows to see the actual data
7.  nba.head()
8.  #configure pandas to display all 23 columns
9.  pd.set_option("display.max.columns", None)
10.  #show only two decimal places
11.  pd.set_option("display.precision", 2)
12.  #display last five rows
13.  nba.tail()
```

When using `.head()`, you `display` the first 5 rows (the default is 5). Unless your screen is quite

large, your output probably will not display all 23 columns, and you will see a result like this:



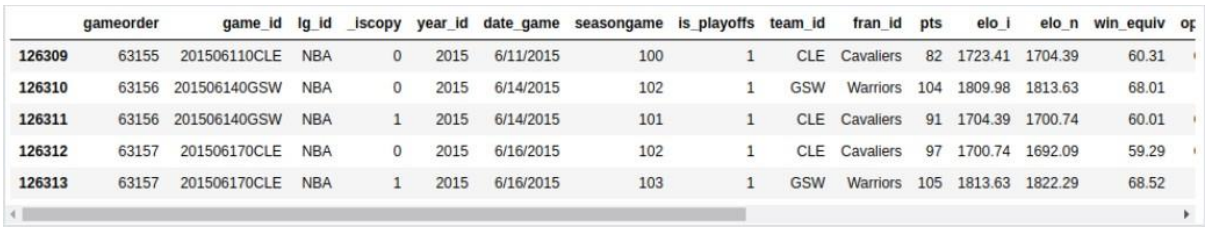
	gameorder	game_id	lg_id	_iscopy	year_id	date_game	seasongame	is_playoffs	team_id	fran_id	...	win_equiv	opp_id	opp_fran	opp_pts	c
0	1	194611010TRH	NBA	0	1947	11/1/1946	1	0	TRH	Huskies	...	40.294830	NYK	Knicks	68	1
1	1	194611010TRH	NBA	1	1947	11/1/1946	1	0	NYK	Knicks	...	41.705170	TRH	Huskies	66	1
2	2	194611020CHS	NBA	0	1947	11/2/1946	1	0	CHS	Stags	...	42.012257	NYK	Knicks	47	1
3	2	194611020CHS	NBA	1	1947	11/2/1946	2	0	NYK	Knicks	...	40.692783	CHS	Stags	63	1
4	3	194611020DTF	NBA	0	1947	11/2/1946	1	0	DTF	Falcons	...	38.864048	WSC	Capitols	50	1

5 rows × 23 columns

You can configure pandas to display all 23 columns using

```
pd.set_option("display.max.columns", None)
```

To verify the changes you made, execute `nba.tail()` to view the last 5 rows. Youcan see that the output of `nba.head()` is now easier to read:



	gameorder	game_id	lg_id	_iscopy	year_id	date_game	seasongame	is_playoffs	team_id	fran_id	pts	elo_i	elo_n	win_equiv	op
126309	63155	201506110CLE	NBA	0	2015	6/11/2015	100	1	CLE	Cavaliers	82	1723.41	1704.39	60.31	
126310	63156	201506140GSW	NBA	0	2015	6/14/2015	102	1	GSW	Warriors	104	1809.98	1813.63	68.01	
126311	63156	201506140GSW	NBA	1	2015	6/14/2015	101	1	CLE	Cavaliers	91	1704.39	1700.74	60.01	
126312	63157	201506170CLE	NBA	0	2015	6/16/2015	102	1	CLE	Cavaliers	97	1700.74	1692.09	59.29	
126313	63157	201506170CLE	NBA	1	2015	6/16/2015	103	1	GSW	Warriors	105	1813.63	1822.29	68.52	

Question.1 (report your answer): Display the first 3 rows of your dataset.Remember that the default `nba.head()` shows the first 5 rows.

Hint: always refer to the documentation; the chances are that you will find a solution by tweaking some optional parameters.

Task3: Get to Know Your Data.

In Task2, you imported a CSV file and had a quick peek at the dataset's contents. So far, you have only seen the size of your dataset and its first and last few rows. Next, you will learn how to examine your data more systematically.

3.1) Discover the different data types your dataset contains.

```
#you can put anything into a list, but the columns of a DataFrame contain values of
 a specific data type.
#by comparing pandas and Python data structures, you will see that this behaviour makes pandas
much faster!
#display all columns and their data types with .info()
nba.info()
```

From the output, you will see that this dataset contains `int64`, `float64` and `object`.pandas use the NumPy library to work with these types.

The object type is noteworthy; according to the pandas' Cookbook, the object data type is "a catch-all for columns that pandas does not recognise as any other specific type." In practice, it often means that all of the values in the column are strings. Although you can store arbitrary Python objects in the object data type, strange values in an object column can harm pandas' performance and interoperability with other libraries. You will see how to deal with this later in the tutorial.

3.2) Show basic statistics.

Get an idea of the values each column contains.

```
#get an idea of the values each column contains
nba.describe()
```

The output shows basic descriptive statistics for all numeric columns.

`.describe()` only analyses numeric columns by default, but you can provide otherdata types if you use the `include` parameter

```
#you can provide other data types using the include parameter
import numpy as np
nba.describe(include=np.object)
```

Here `.describe()` will not calculate a mean or a standard deviation for object columns since they mainly include text strings. It just displays descriptive statistics.

Look at the output you got and answer the following question:

	game_id	lg_id	date_game	team_id	fran_id	opp_id	opp_fran	game_location	game_result	notes
count	126314	126314	126314	126314	126314	126314	126314	126314	126314	5424
unique	63157	2	12426	104	53	104	53	3	2	231
top	198801240POR	NBA	4/16/2014	BOS	Lakers	BOS	Lakers	H	W	at New York NY
freq	2	118016	30	5997	6024	5997	6024	63138	63157	440

Question.2 (report your answer): Take a look at the `team_id` and `fran_id` (franchise) columns; what observations can you make at this point (i.e. do you see anything strange here)? Write your initial observation, then carry on with section 3.3 to be able to answer it by exploring your dataset.

3.3) Exploring the dataset

Exploratory data analysis helps you answer questions about your dataset, for example, exploring how often specific values occur in a column. Let us take a look at the two columns `team_id` and `fran_id`:

```
1. nba["team_id"].value_counts()
2. nba["fran_id"].value_counts()
```

A team named "Lakers" played 6024 games, but only 5078 of those were played by the Los Angeles Lakers. To find out who the other "Lakers" team is, execute the following line of code:

```
nba.loc[nba["fran_id"] == "Lakers", "team_id"].value_counts()
```

`pandas.DataFrame.loc` is used to access a group of rows and columns by label(s) or aboolean array.

The output shows that the Minneapolis Lakers ("MNL") played the other 946games.

Let us find out when they played those games:

```
1. #find out when they played those games
2. nba.loc[nba["team_id"] == "MNL", "date_game"].min()
3. nba.loc[nba["team_id"] == "MNL", "date_game"].max()
4. nba.loc[nba["team_id"] == "MNL", "date_game"].agg(("min", "max"))  #aggregate the
   two functions
```

Question.3 (report your answer): Find out how many wins and losses the Minneapolis Lakers had and how many points they scored during the matches in the dataset.

Question.4 (report your answer): Now you understand why the Boston Celtics team "BOS" played the most games in the dataset, find out how many points theBoston Celtics have scored during all matches contained in this dataset.

Question.5 (report your answer): After exploring your dataset, explain your observations from Question.2 in a structured way.

Task4: Data Access methods (loc and iloc).

Check pandas' official docs for these two functions.

With data access methods like `.loc` and `.iloc`, you can select just the suitable subsetof your DataFrame to help you answer questions about your dataset.

`.loc` uses the label and `.iloc` the positional index

```
#Examples:

#accessing data using a label
city_data.loc["Amsterdam"]

#accessing data between two labels
city_data.loc["Tokyo": "Toronto"]

#accessing data using the positional index
city_data.iloc[1]
```

Question.6 (report your answer):

6.1) Use a data access method to display the 4th row from the bottom of the `nba` dataset.

6.2) Use a data access method to display the 2nd row from the top of the `nba` dataset.

6.3) Access all games between the labels 5555 and 5559; you only want to see the names of teams and the scores.

Task5: Querying the Dataset.

You have seen how to access subsets of a huge dataset based on its indices, and now, you will select rows based on the values in your dataset's columns to query your data.

```
#create a new DataFrame that contains only games played after 2010
current_decade = nba[nba["year_id"] > 2010]
current_decade.shape
```

All the columns are still there, but `current_decade` only consists of rows where the value in the `year_id` column is greater than 2010.

Question.7 (report your answer): Create a new DataFrame that consists of thegames played between 2000 and 2009.

Selecting rows where a specific field is not null

```
#selecting rows where a specific field is not null .notnull() or .notna()
games_with_notes = nba[nba["notes"].notnull()]
games_with_notes.shape
```

The DataFrame nba has 126314 rows, and when you select only the rows where `notes` are not null, you end up with a DataFrame of 5424 rows.

You can access the object data type values as str and perform string methodson them.

```
#filter your dataset and find all games where the home team's name ends with "ers".
ers = nba[nba["fran_id"].str.endswith("ers")]
ers.shape



#search for Baltimore games where both teams scored over 100 points.
#In order to see each game only once, you'll need to exclude duplicates
nba[(nba["_iscopy"] == 0) & (nba["pts"] > 100) & (nba["opp_pts"] > 100) & (nba["team

_id"] == "BLB")]
```

You use `df["_iscopy"] == 0` to include only the entries that aren't copies.

Question.8 (report your answer): Filter your dataset and find all the playoffs games where the number of points scored by both home and away was more than 100 in 2011 and make sure you do not include duplicates (do notforget the parentheses).

Task6: Grouping and Aggregating your Data.

You may also want to learn other features of your dataset, like the sum, mean,or average value of a group of elements. Luckily, the pandas Python library offers grouping and aggregation functions to help you accomplish this task.

```
#grouping - group all games for fran_id and sum their points and override the default of sorting
nba.groupby("fran_id", sort=False)["pts"].sum()

#group by multiple columns
nba[(nba["fran_id"] == "Spurs") & (nba["year_id"] > 2010)].groupby(["year_id", "game_result"])["game_id"].count()
```

Question.9 (report your answer): Take a look at the New York Knicks 2011-12season (year_id: 2012). How many wins and losses did they score during the regular season and the playoffs?

Task7: Manipulating Columns.

You can add and drop columns as part of the initial data cleaning phase or laterbased on the insights of your analysis.

```
#create a copy of your original DataFrame to work with
df = nba.copy()
df.shape

#define new columns based on the existing ones
df["difference"] = df.pts - df.opp_pts
df.shape

#use an aggregation function .max() to find the largest value of your new column
df["difference"].max()

#rename the columns of your dataset
renamed_df = df.rename(columns={"game_result": "result", "game_location": "location"
})
renamed_df.info()
```

Note that there is a new object, `renamed_df`. Like several other data manipulationmethods, `.rename()` returns a new DataFrame by default.

```
#delete unwanted columns – you will not be analysing Elo ratings here, so go ahead and delete them
df.shape
elo_columns = ["elo_i", "elo_n", "opp_elo_i", "opp_elo_n"]
df.drop(elo_columns, inplace=True, axis=1)
df.shape
```

Understanding the `df.drop` function:

```
DataFrame.drop(self, labels=None, axis=0, index=None, columns=None, level=None,
inplace=False, errors='raise')
```

It is translated into:

Index or column labels to drop, Whether to drop labels from the index (0 or 'index') or columns (1 or 'columns'), 'inplace=True' to make permanent changesto the data frame

Task8: Specifying Data Types.

When you create a new DataFrame, pandas assigns a data type to each columnbased on its values. It is sometimes not too accurate. Choose the correct data type for your columns upfront to improve performance.

Take another look at the columns of the `nba` dataset:

```
#take a look at the DataFrame
df.info()
```

Ten of your columns have the data type `object`, and some of these are goodcandidates for data type conversion.

```
# use .to_datetime() to specify all game dates as datetime objects.
df["date_game"] = pd.to_datetime(df["date_game"])
```

Similarly, `game_location` can have only three different values. In a relational database, you would use the type `enum` for this column. pandas provides the`categorical` data type for that exact purpose.

```
#game_location column can have only three different values.
#you can see this by executing this code
df["game_location"].nunique()
df["game_location"].value_counts()

#change the data type to categorical and check it
df["game_location"] = pd.Categorical(df["game_location"])
df["game_location"].dtype
```

After changing to categorical data, execute `df.info`. You will notice a drop inmemory usage, hence improving performance.

Question.10 (report your answer): Find another column in the `nba` dataset with a generic data type and convert it to a more specific one.

Task9: Cleaning the Data

9.1) Missing Values

`.info()` shows how many non-null values a column contains. That is critical information for you to have about your data. Null values oftenindicate a problem in the data-gathering process.

When you inspect the dataset with `nba.info()`, you will see that it is pretty neat except for the `notes` column, which contains null values for most of its rows. This output shows that the notes column has only 5424 non-null values.

That means that over 120,000 rows of your dataset have null values in thiscolumn.

Here are a few ways to deal with null values:

```
#1st way- usually best approach is to ignore them, remove all rows with missing values
rows_without_missing_data = nba.dropna()
rows_without_missing_data.shape

#2nd way - drop columns if they are not relevant to your analysis
data_without_missing_columns = nba.dropna(axis=1)
data_without_missing_columns.shape

#3rd way - replace the missing values with a meaningful default value for your use case

data_with_default_notes = nba.copy()
data_with_default_notes["notes"].fillna(value="no notes at all", inplace=True)
data_with_default_notes["notes"].describe()
```

Regarding the first way, that kind of data clean-up does not make sense for your `nba` dataset because it is not a problem for a game to lack notes. However, if your dataset contains a million good records and a hundred where relevant data is missing, then dropping the incomplete records can be a reasonable solution.

9.2) Invalid Values

Use `.describe` to understand more about your dataset. It can help you identifyinvalid values that may throw off your analysis.

	gameorder	_iscopy	year_id	seasongame	is_playoffs	pts	elo_i	elo_n	win_equiv	opp_pts	opp_elo_i	opp_elo_n	forecast
count	126314.00	126314.0	126314.00	126314.00	126314.00	126314.00	126314.00	126314.00	126314.00	126314.00	126314.00	126314.00	126314.00
mean	31579.00	0.5	1988.20	43.53	0.06	102.73	1495.24	1495.24	41.71	102.73	1495.24	1495.24	0.50
std	18231.93	0.5	17.58	25.38	0.24	14.81	112.14	112.46	10.63	14.81	112.14	112.46	0.22
min	1.00	0.0	1947.00	1.00	0.00	0.00	1091.64	1085.77	10.15	0.00	1091.64	1085.77	0.02
25%	15790.00	0.0	1975.00	22.00	0.00	93.00	1417.24	1416.99	34.10	93.00	1417.24	1416.99	0.33
50%	31579.00	0.5	1990.00	43.00	0.00	103.00	1500.95	1500.95	42.11	103.00	1500.95	1500.95	0.50
75%	47368.00	1.0	2003.00	65.00	0.00	112.00	1576.06	1576.29	49.64	112.00	1576.06	1576.29	0.67
max	63157.00	1.0	2015.00	108.00	1.00	186.00	1853.10	1853.10	71.11	186.00	1853.10	1853.10	0.98

While looking at the output, you will see that the `year_id` varies between 1947 and 2015. That sounds plausible. However, how can the minimum points of a game be 0? Take a look at those games to find out if it makes sense.

```
#selecting the games where pts are 0
nba[nba["pts"] == 0]
```

It seems the game was forfeited. Depending on your analysis, you may want toremove it from the dataset.

9.3) Inconsistent Values

Always check for inconsistent values. The values of the fields `pts`, `opp_pts` and `game_result` should be consistent with each other.

```
#check using the .empty() attribute
nba[(nba["pts"] > nba["opp_pts"]) & (nba["game_result"] != 'W')].empty
nba[(nba["pts"] < nba["opp_pts"]) & (nba["game_result"] != 'L')].empty
```

Fortunately, both of these queries return an empty DataFrame. Nevertheless, be preparedfor surprises; always check the consistency.

Task10: Data Visualisation

Sometimes, the numbers speak for themselves, but a chart often helps communicate your insights.

Data visualisations make big and small data more uncomplicated for the human brain to understand, and visualisation also makes it easier to detect patterns, trends, and outliers in data groups.

Data visualisation is one of the things that works much better in a Jupyter notebook than in a terminal. If you need help getting started, then check out Jupyter Notebook: An Introduction.

Both Series and DataFrame objects have a `.plot()` method, which is a wrapperaround `matplotlib.pyplot.plot()`.

Visualise how many points the Knicks scored throughout the seasons.

```
#include this line to show plots directly in the notebook
%matplotlib inline

#visualise how many points the Knicks scored throughout the seasons
nba[nba["fran_id"] == "Knicks"].groupby("year_id")["pts"].sum().plot()

#create a bar plot to show the franchises with the most games played
nba["fran_id"].value_counts().head(10).plot(kind="bar")
```

Question.11 (report your answer):

11.1) Explain what the above line plot, showing how many points the Knicks scored throughout the seasons, reveals to you (i.e. describe what you find out).

11.2) Describe what the above bar plot reveals about the franchises with the most games played.

11.3) In 2013, the Miami Heat won the championship. Create a pie plot showing the count of their wins and losses during that season. (First, define a criterion to include only the Heat's games from 2013. Then, create a plot in the same way asyou have seen above).

"Without data, you are just another person with an opinion."

~W. Edwards Deming

References:

pandas Docs: https://pandas.pydata.org/pandas-docs/stable/index.html

Download Python: https://www.python.org/downloads/

How to install pip: https://www.liquidweb.com/kb/install-pip-windows/

Jupyter Notebook: https://realpython.com/jupyter-notebook-introduction/

Install pandas Python: https://pandas.pydata.org/pandas-docs/stable/getting_started/install.html

Install Matplotlib: https://matplotlib.org/users/installing.html#installing-from-source

Visualisation with pandas: https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html

Tutorial inspiration (Real Python): https://realpython.com/pandas-python-explore-dataset/

Data Source: https://fivethirtyeight.com/

The raw data: https://raw.githubusercontent.com/fivethirtyeight/data/master/nba-elo/nbaallelo.csv