




Pandas and Data

Coding Bootcamp

Learning Objectives

A thick yellow horizontal bar spans the width of the slide, with a vertical yellow bar extending downwards from its right end.

- Familiar understanding of data manipulation
- Basic Pandas commands
- Deeper understanding of data exploration
- Basic ML model prediction

Learning Requirements

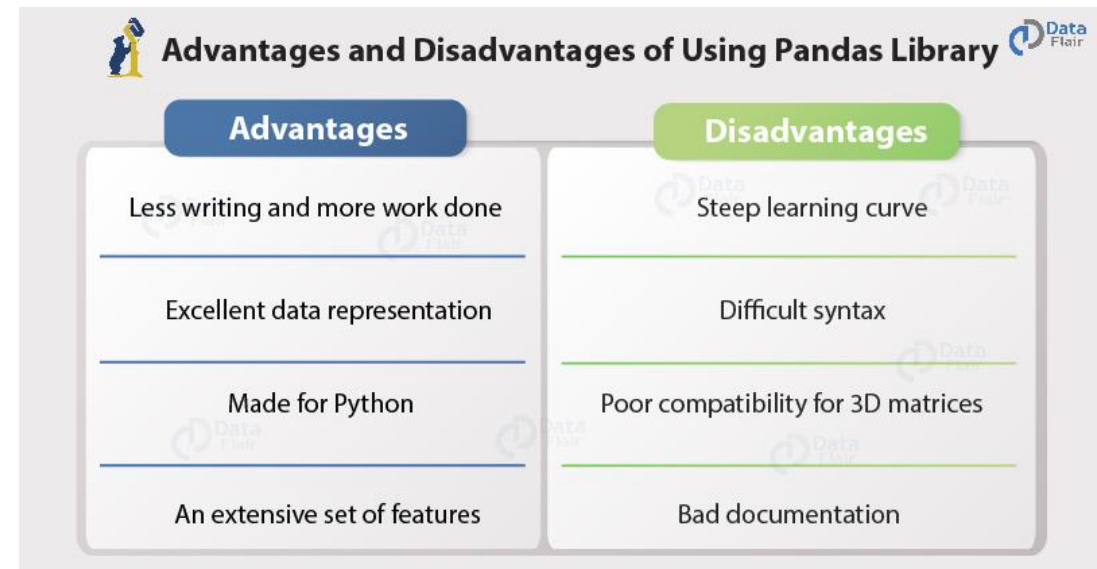
- We will build upon our knowledge of data exploration
- Python syntax.
- Libraries and third party python modules.
- Basic mathematical understanding.

Today's Task!

- Today we will be looking at using the third part library Pandas.
- Pandas is a core library used by Data Scientists around the world, it is completely open-source and pandas Data-Frames are used in Machine Learning very often.
- *“pandas aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language.”*

Why use pandas?

- This handy pros and cons list from Data-Flair helps us visualize the benefits and drawbacks of using pandas.
- The main draw for Data Scientists to pandas are the basic functions they provide to make the manipulation of data very easy. Pandas has some in-built methods that can do some very heavy lifting.



The DataFrame

- One of the best and most commonly used features is the pandas DataFrame.
- The DataFrame turns your input data into an object that is similar to an SQL table, or an excel sheet – and is easy to navigate and manipulate.

The diagram illustrates a DataFrame as a table with 7 rows and 4 columns. The columns are labeled NAME, AGE, GRADE, and MARKS. The rows are numbered 1 through 7. A green bracket on the left side of the table is labeled 'ROWS'. A green bracket at the top of the table is labeled 'COLUMNS'. A green arrow points from the word 'DATA' at the bottom to the table, indicating the data content.

	NAME	AGE	GRADE	MARKS
1	ALEX	16	A	88
2	STEVE	16	C	34
3	JHON	17	B	66
4	WILEY	16	B	75
5	SMITH	18	A	82
6	DAVE	16	A	90
7	KYLE	17	C	44



“IT IS A CAPITAL MISTAKE TO THEORIZE BEFORE ONE HAS DATA. INSENSIBLY ONE BEGINS TO TWIST FACTS TO SUIT THEORIES, INSTEAD OF THEORIES TO SUIT FACTS.”



— SIR ARTHUR CONAN DOYLE, SHERLOCK HOLMES

Download the data either from Kaggle.com, from SeriousWill's github, or manually import it.

Download the data

```
import requests
# 2. Get the link or url
url =
'https://raw.githubusercontent.com/seriouswill/footy_data/main/data/Premier%20League%20All%20Time%20Top%2050%20Goal%20Scorers%20UTF8.csv'
r = requests.get(url, allow_redirects=True)
# 3. Save the content with name.
open('footy_data.csv', 'wb').write(r.content)
```


Virtual Env & Pip

- Create a new jupyter notebook, call it something like main.ipynb
- As always, we'll set up a virtual environment and pip install our relevant packages we'll be working with.

```
PS C:\pandasExample> py -m venv footy
PS C:\pandasExample> footy\Scripts\activate
PS C:\pandasExample> pip install jupyter
(footy) PS C:\pandasexample> pip install pandas
```

Read csv file

- Next we can use the `read_csv()` and the `describe()` methods to see the shape of the data.
- Then run the `.columns` attribute to see the headers of the data. Think of these as column in the table.

```
footy_path_file = "footy_data.csv"
footy_data = pd.read_csv(footy_path_file,
encoding="utf-8")
footy_data.describe() # run this cell first

####

footy_data.columns # then run this cell
```

Read csv file

- Now we can use the pandas in built function `pd.DataFrame()` to turn our csv data into a pandas Data-Frame.
- Then call `.head()` method on it to see the top 5 results.
- Then in a new cell call the `.tail()` method to see the last 5.

```
df = pd.DataFrame(footy_data)

df.head()

### Next Cell

df.tail()
```

Shape of the data

- We can look at the shape of our data with the `df.shape` attribute.
- This will check how large the DataFrame is:



```
df.shape
```

Accessing column data

- We can access specific columns in our data by using square brackets and the specific string or just the index value.
- Give it a go.

```
df["Minutes Per Goal"].head()
```

```
df["Penalty Goals"].tail()
```

```
df[1].head()
```

```
df[4].tail()
```

Finding range / min, max, mean

- Access a short range with start and stop values in square brackets.
- Find the min value, the max value, or the average value quickly with no custom functions with `.min()`, `.max()` and `.mean()`

```
df["Players"][3:7]  
  
df["Penalty Goals"].min()  
df["Penalty Goals"].max()  
  
df["Penalty Goals"].mean()
```

Accessing rows

- To access the row in our data we simply use the `.loc[x]` attribute where `x` is the index variable.
- We can use either dot notation, or square brackets to access columns

```
# access Players with square brackets  
df["Players"].loc[0:7]  
  
# access Players with dot notation  
df.Players.loc[0:7]
```

Indexing in pandas

- We can also access the index using `iloc`, which check the index location of the selection. We can use lists, minus values and other normal python syntax here:

```
df.iloc[[0,2,4], [0,4,5,6]]
```

```
df.iloc[[-1, -2, -3],[0, 5, 6, 10, 11]]
```



```
df.set_index( "Players" )
```

Manipulating the index

We can change which column we can use as our index via the `set_index()` method.

We can group series of data inside the DataFrame together and plot some cool things with it.

Manipulating the data

```
group = df.groupby('Players')['Goals'].sum()  
print(group)  
  
### Run this cell, then the next to see the difference  
  
group = group.sort_values()  
print(group)
```

Displaying data

- Import matplotlib.pyplot as plt and we can use our 'group' variable we made in the previous slide to view our data!
- We can add multiple arguments to customize the graph!

```
import matplotlib.pyplot as plt

group.plot(kind='barh', figsize=(20, 15))
plt.title('Players Goals', fontdict={'fontsize':30,
'weight':'bold'}, pad=30)
plt.show()
```

Displaying data

- We can have a glance at some data, and get a basic idea as to whether or not there is any correlation.
- Import matplotlib.pyplot as plt and use two features for our y axis – that way we can compare them! Set xticks rotation so we can see them more easily.

```
import matplotlib.pyplot as plt

x = df.Players
y1 = df["Penalty Goals"]
y2 = df.Goals

plt.plot(x, y1)
plt.plot(x, y2)
plt.xticks(rotation=90)
plt.show()
```

Pandas helpful functions

- Pandas is chock full of helpful functions to interpret our data.
- Let's say we want to check a player's overall Goals against the average of the five players either side of them, we use the `.mean()` method!
- In this example – write a line, then run the cell before writing the next to see the output change.

```
df.iloc[6]           # Run this line in cell first

df.iloc[6].Goals     # Then this

y = []              # Then this... and so on

y.append(df.iloc[6].Goals)

df.iloc[0:12].Goals

df.iloc[0:12].Goals.mean( )

y.append(df.iloc[0:12].Goals.mean( ))

x = ["Harry K", "Top 12 Mean"]

plt.barh(x, y)
```

Pandas reviewing metrics

- Hang on though, some players had been playing for far longer than others to get all the goals in!
- Lets look at another metric! Does it tell a different story?

```
group_goals_per_game = df.groupby('Players')['Goals Per Game/Ratio'].sum()  
  
group_goals_per_game = group_goals_per_game.sort_values()  
print(group_goals_per_game)  
  
group_goals_per_game.plot(kind='barh', figsize=(20, 15))  
plt.title('Players Goals Per Game Ratio', fontdict={'fontsize':30})  
plt.show()
```

Feature selection

- Let's take some of our columns to use as features for a prospective machine learning model.
- We can do this by using a normal list of strings or by adding them dynamically.

```
df.columns # to remind us

footy_features = ["Players", "Minutes Per Goal", "Goals Per Game/Ratio",
                  "Goals"]

# Could do it like this by selecting with strings.
# Or we could do it dynamically if there was an issue with some characters.

df.columns[0]

features = []

features.append(df.columns[0])
features.append(df.columns[5])
features.extend(df.columns[10:12])
print(features)

x = footy_data[features]

x.describe()
```

Prediction target selection

- Ok, so what is it we want to predict?
- How about how many Appearances it will take for a player to top the chart, based on our feature data?

```
# First lets tackle that typo!  
df.rename(columns={df.columns[4]: "Appearances"}, inplace=True)  
df.head() # to check the name change  
y = df.Appearances # set our Y value
```


THE DATA IS READY!



Choosing a ML model

- The data is all prepared and ready, so let's make some predictions. We will use Sci-Kit Learn, a Machine Learning library to pick one pre-built.

```
from sklearn.tree import DecisionTreeRegressor

# Define model. Specify a number for random_state to ensure same results each
run
footy_model = DecisionTreeRegressor(random_state=1)

# Fit model
footy_model.fit(x, y)
```

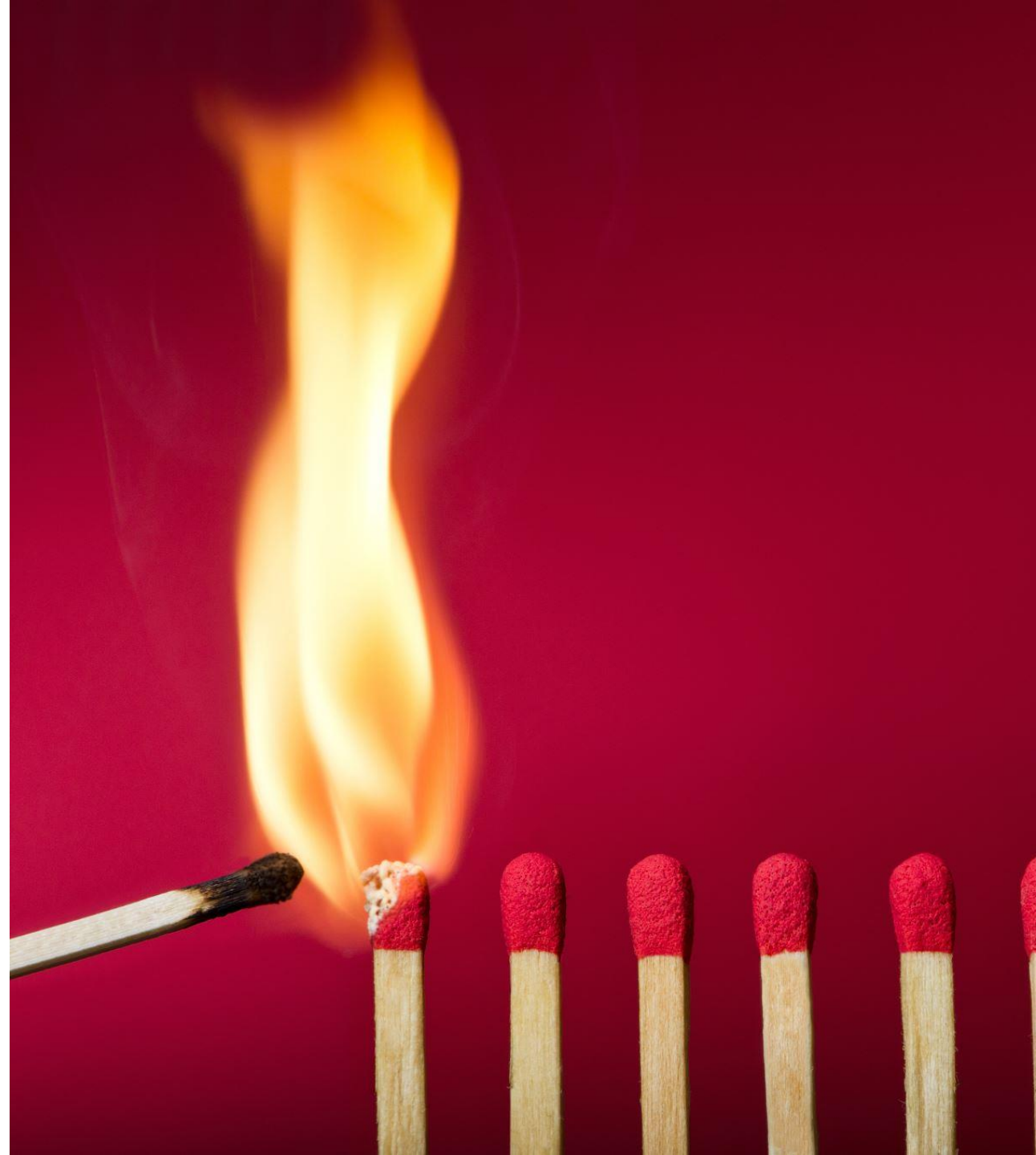
Checking the predictions

- The model predictions should be the same as the Appearances column, let's check to see if the model has been trained correctly.

```
print(df.Appearances.head()) # The pre-existing data  
  
print("Model predictions:")  
print(footy_model.predict(x.head())) # The prediction data
```

Making new predictions

- Great! Let's make a new bit of data to give to our model to predict the amount of matches they'll need to play in order to reach a hundred goals.



Making new predictions

- First off we'll make up some data inside a python dictionary – just for practice! Remember, the data we need is Minutes per Goal, Goal/Game Ratio and total Goals.
- Then we'll take the keys and values from that dictionary.

```
new_player_dict = {"Will `The Hitman` O'Connell": ["180", "0.42", "100"],  
                  "Jordan `The Darling` Darlington": ["102", "0.72", "100"]}  
# Made a new dictionary with some totally accurate data  
  
key_list = list(new_player_dict.keys())    # Gets the keys  
val_list = list(new_player_dict.values())  # Gets the values
```

Making new predictions

- Next, we can use our `.predict()` function to make the predictions based on the new made up data, and couch it inside some f-strings to display the data with a bit more flair.

```
print(f"The matches necessary for {key_list[0]} to score a hundred goals is  
{footy_model.predict(val_list[0])}!")  
  
print(f"Whilst, the matches necessary for {key_list[1]} to score a hundred  
goals is {footy_model.predict(val_list[1])}!")
```

That code together:

Making new predictions

```
new_player_dict = {"Will `The Hitman` O'Connell":[["180", "0.42", "100"]],  
                  "Jordan `The Darling` Darlington":[["102", "0.72", "100"]]}  
# Made a new dictionary with some totally accurate data  
  
key_list = list(new_player_dict.keys())    # Gets the keys  
val_list = list(new_player_dict.values())  # Gets the values  
  
print(f"The matches neccesary for {key_list[0]} to score a hundred goals is  
{footy_model.predict(val_list[0])}!")  
  
print(f"Whilst, the matches neccesary for {key_list[1]} to score a hundred  
goals is {footy_model.predict(val_list[1])}!")
```



Back of the net

Fun with Pandas

Superb work through those steps!

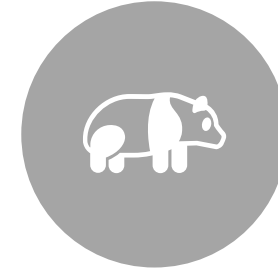
Pandas has absolutely tons to offer as a library for data manipulation, exploration, and so much more.

We will return to pandas later on in the course, but for now, we'll leave things there.

Learning Objectives



FAMILIAR
UNDERSTANDING OF
DATA MANIPULATION



BASIC PANDAS
COMMANDS



DEEPER
UNDERSTANDING OF
DATA EXPLORATION



BASIC ML MODEL
PREDICTION



Activity Goals

- **Activity 1:**
 - **If you're after more data exploration, look no further than [Kaggle.com](https://www.kaggle.com) – you'll be able to find tons of free datasets to load and explore.**
 - **Find a new dataset, load it into a Jupyter notebook, and do some Light Data Exploration, with pandas!**
-

