Welcome to CSC 276 Data Science

CSC 276: Data Science

Lecture #5

Introduction

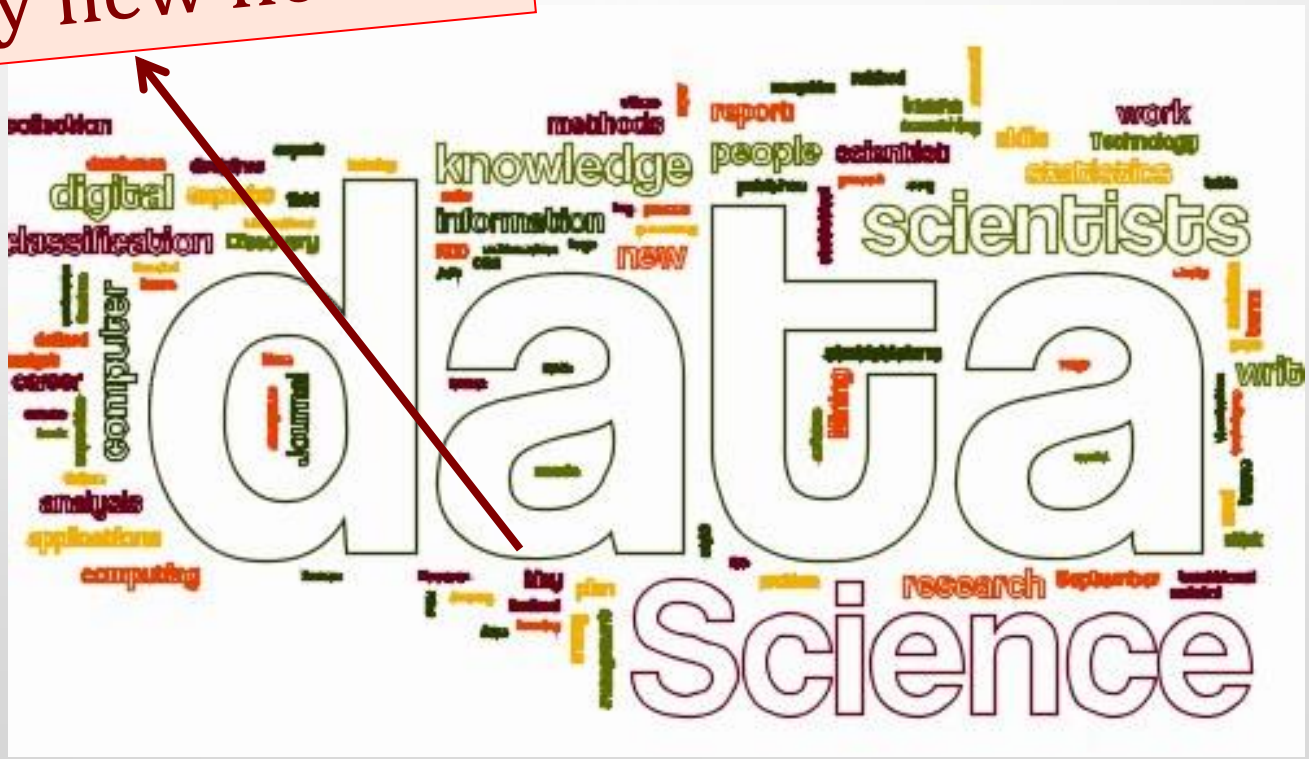
Dr.Fatema Nafa

Fall 2022

Welcome to CSC 276!

This class is truly seminar-style: I'm here, as you are, in order to gain insights into this very new field… .

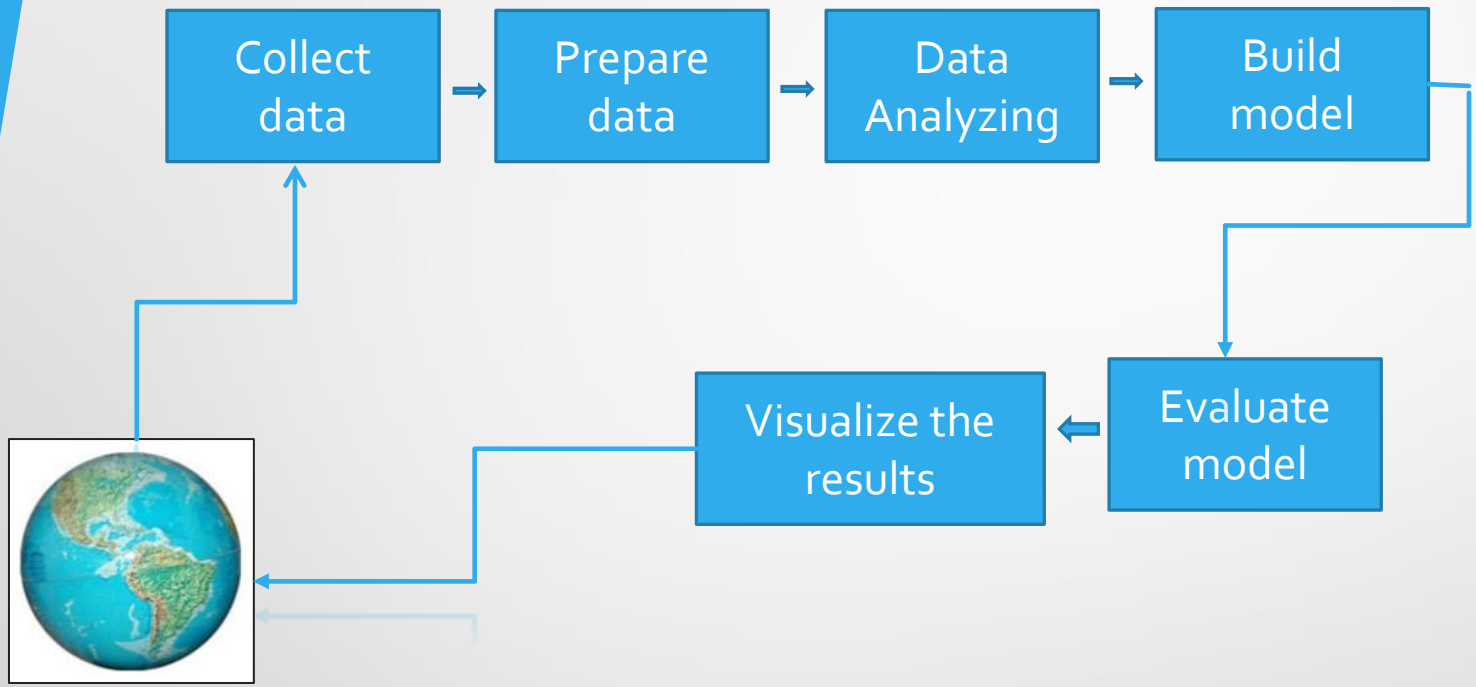
Data Science

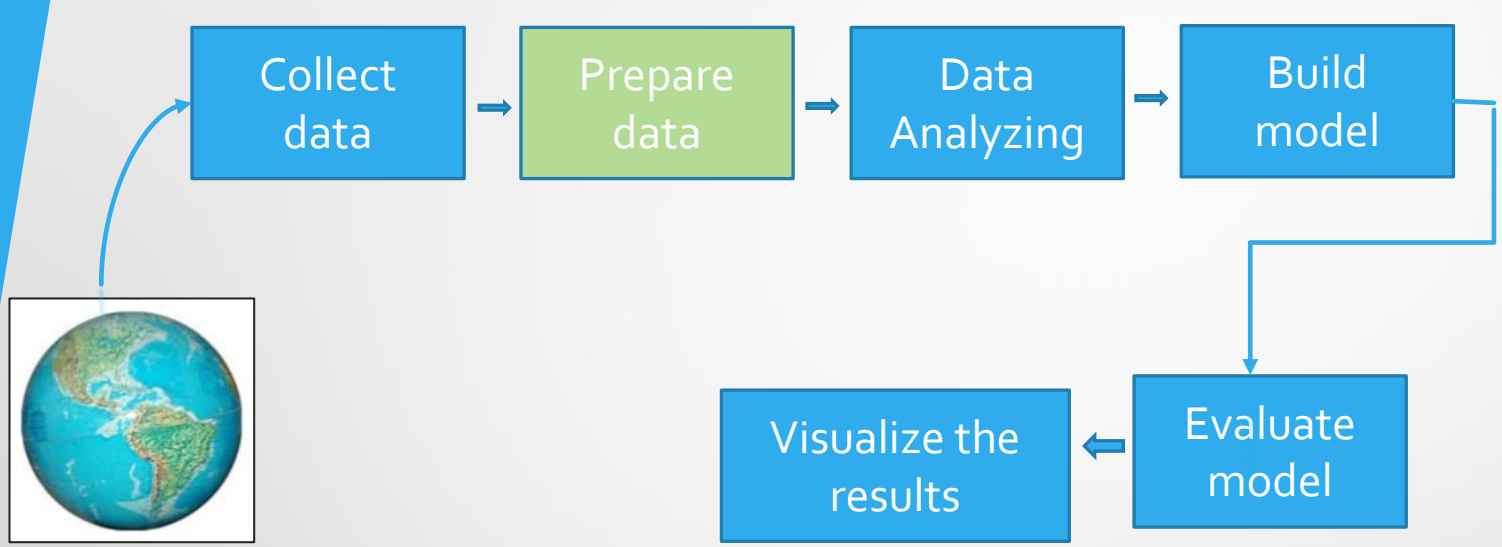


Lecture Outline

- **The Art of Data Science**
- Volume, Velocity, Variety
- The Logic of Data Science
- How to Be Agile
- Treating Data as Evidence
- **Python**
  - Fundamentals of Data Manipulation
  - Basic Data Processing with Pandas
  - Answering Questions with Messy Data


Any
Questions


Collect data
Prepare data
Data Analyzing
Build model
Evaluate model
Visualize the results


Collect data
Prepare data
Data Analyzing
Build model
Evaluate model
Visualize the results


Any
Questions

Loading Python Libraries

```
#Import Python Libraries
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib as mpl
import seaborn as sns
```

Press `Shift+Enter` to execute the *jupyter* cell

Reading data using pandas

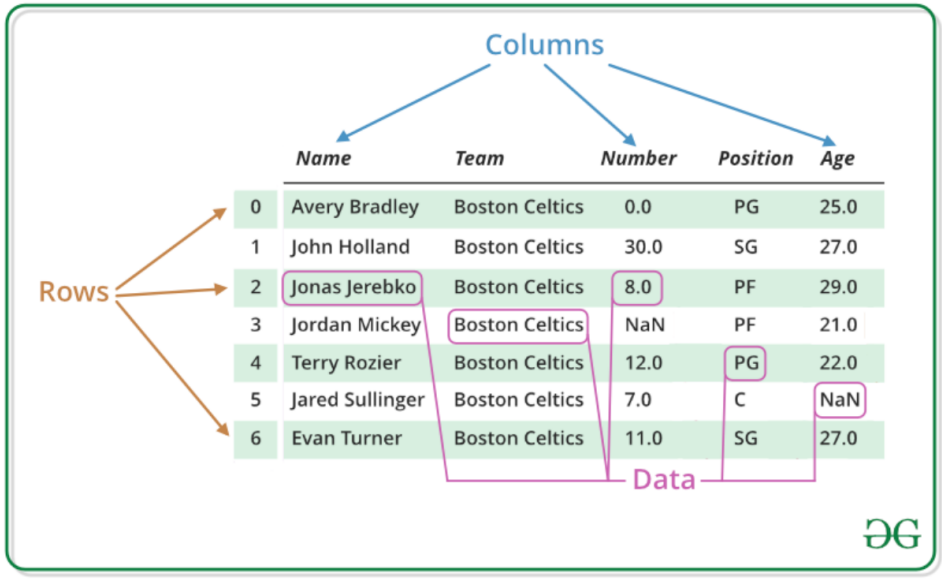
```
#Read csv file
Import pandas as pd
df = pd.read_csv("events.csv")
```

Note: *The above command has many optional arguments to fine-tune the data import process.*

There is a number of pandas commands to read other data formats:

```
pd.read_excel('myfile.xlsx',sheet_name='Sheet1', index_col=None,
na_values=['NA'])
pd.read_stata('myfile.dta')
pd.read_sas('myfile.sas7bdat')
pd.read_hdf('myfile.h5','df')
```

Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the **data**, **rows**, and **columns**.



In [29]:
```
import pandas as pd
Mylist = ['Java', 'R', 'Python', 'C++']
df = pd.DataFrame(Mylist)
print(df)
```

In [30]:
```
import pandas as pd

MyDic = {'language':['Java', 'R', 'Python', 'C++'],
         'Level':[3, 2, 1, 0]}
df = pd.DataFrame(MyDic)
df
```

In [38]:
```
MySet = {"Java","Python","R"}
df = pd.DataFrame(MySet)
df
```

List

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

Lists are created using square brackets:

```
In [34]: MyList = ["Java", "Python", "R", "C++"]
         print(len(MyList))

         4
```

```
In [36]: list1 = ["Python", 34, True, 2021, "DataScience"]
         list1

Out[36]: ['Python', 34, True, 2021, 'DataScience']
```

Creating a Dictionary

In Python, a Dictionary can be created by placing a sequence of elements within curly **{}** braces, separated by 'comma'. Dictionary holds a pair of values, one being the Key and the other corresponding pair element being its **Key:value**. Values in a dictionary can be of any data type and can be duplicated, whereas keys can't be repeated and must be *immutable*.

Note – Dictionary keys are case sensitive, the same name but different cases of Key will be treated distinctly.

```
In [32]:
DicExa = {1: 'Java', 2: 'C++', 3: 'Python'}
DicExa

Out[32]: {1: 'Java', 2: 'C++', 3: 'Python'}
```

Set

Sets are used to store multiple items in a single variable.

Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Tuple, and Dictionary, all with different qualities and usage.

A set is a collection which is both *unordered* and *unindexed*.

Sets are written with curly brackets.

```
In [37]: myset = {"Java", "Python", "R"}
         myset

Out[37]: {'Java', 'Python', 'R'}
```

Exploring data frames

In [40]:
```
import pandas as pd
df = pd.read_csv("events.csv")
df.head()
```

Hands-on exercises

- ✓ Try to read the first 10, 20, 50 records;
- ✓ Can you guess how to view the last few records;

Data Frame data types

Pandas Type	Native Python Type	Description
object	string	The most general dtype. Will be assigned to your column if column has mixed types (numbers and strings).
int64	int	Numeric characters. 64 refers to the memory allocated to hold this character.
float64	float	Numeric characters with decimals. If a column contains numbers and NaNs(see below), pandas will default to float64, in case your missing value has a decimal.
datetime64, timedelta[ns]	N/A (but see the datetime module in Python's standard library)	Values meant to hold time data. Look into these for time series experiments.


Any
Questions

Data Frame data types

```
#Check a particular column type
df['Colum_Name'].dtype
```

```
#Check types for all the columns
df.dtypes
```

Hands-on exercises

Statistical Information about my data	
File Type	**What does my file look like?**
Description	What type of data and why?
#of records	**How many records**
#of columns	**How many columns**
Size	number of dimensions
Colum's Names	**How many columns and records**
shape	return a tuple representing the dimensionality
values	numpy representation of the data
descriptive statistics	describe()

Hands-on exercises

Statistical Information about my data	
File Type	My file format is CSV File
Description	What type of data and why?
#of records	**How many records**
#of columns	**How many columns**
Size	number of dimensions
Colum's Names	**How many columns and records**
shape	return a tuple representing the dimensionality
values	numpy representation of the data
descriptive statistics	describe()

Hands-on exercises

The dataset provides a granular view of 9,074 games, totaling 941,009 events from the biggest 5 European football (soccer) leagues: England, Spain, Germany, Italy, France from 2011/2012 season to 2016/2017 season as of **25.01.2017.**
There are games that have been played during these seasons for which I could not collect detailed data. Overall, **over 90% of the played games during these seasons have event data**.

Hands-on exercises

Statistical Information about my data	
File Type	My file format is CSV File
Description	What type of data and why?
#of records	**How many records**
#of columns	**How many columns**
Size	number of dimensions
Colum's Names	**How many columns and records**
shape	return a tuple representing the dimensionality
values	numpy representation of the data
descriptive statistics	describe()

- df.info()
- Get the number of rows: len(df)
- Get the number of columns: len(df.columns)
- Get the number of rows and columns: df.shape
- row, col = df.shape print(row)

Statistical Information about my data		
File Type	**What does my file look like?**	
Description	What type of data and why?	
#of records	**How many records**	
#of columns	**How many columns**	
Size	number of dimensions	Print(df.size)
Colum's Names	**Print columns names**	
shape	return a tuple representing the dimensionality	
descriptive statistics	describe()	

Statistical Information about my data		
File Type	**What does my file look like?**	
Description	What type of data and why?	
#of records	**How many records**	
#of columns	**How many columns**	
Size	number of dimensions	Print(df.size)
Colum's Names	**Print columns names**	
shape	return a tuple representing the dimensionality	
values	numpy representation of the data	
descriptive statistics	describe()	

```
In [16]: for Colum_name in df.columns:
             print(Colum_name)

id_odsp
id_event
sort_order
time
text
event_type
event_type2
side
event_team
opponent
player
player2
player_in
player_out
shot_place
shot_outcome
is_goal
location
bodypart
assist_method
situation
fast_break
```

```
In [26]: mylist

Out[26]: ['id_odsp',
          'id_event',
          'sort_order',
          'time',
          'text',
          'event_type',
          'event_type2',
          'side',
          'event_team',
          'opponent',
          'player',
          'player2',
          'player_in',
          'player_out',
          'shot_place',
          'shot_outcome',
          'is_goal',
          'location',
          'bodypart',
          'assist_method',
          'situation',
          'fast_break']
```

Statistical Information about my data		
File Type	**What does my file look like?**	
Description	What type of data and why?	
#of records	**How many records**	
#of columns	**How many columns**	
Size	number of dimensions	Print(df.size)
Colum's Names	**Print columns names**	
shape	return a tuple representing the dimensionality	
values	numpy representation of the data	
descriptive statistics	describe()	

```
In [27]: df.describe()
```

Out[27]:

	sort_order	time	event_type	event_type2	side	shot_place	shot_outcome	is_goal	location	boc
count	941009.000000	941009.000000	941009.000000	214293.000000	941009.000000	227459.000000	228498.000000	941009.000000	467067.000000	229185.0
mean	53.858826	49.663663	4.326575	12.233764	1.481170	5.733693	1.926555	0.025978	6.209073	1.6
std	32.014268	26.488977	2.995313	0.468850	0.499646	3.326100	0.797055	0.159071	5.421736	0.7
min	1.000000	0.000000	1.000000	12.000000	1.000000	1.000000	1.000000	0.000000	1.000000	1.0
25%	27.000000	27.000000	2.000000	12.000000	1.000000	2.000000	1.000000	0.000000	2.000000	1.0
50%	53.000000	51.000000	3.000000	12.000000	1.000000	5.000000	2.000000	0.000000	3.000000	1.0
75%	79.000000	73.000000	8.000000	12.000000	2.000000	9.000000	3.000000	0.000000	11.000000	2.0
max	180.000000	100.000000	11.000000	15.000000	2.000000	13.000000	4.000000	1.000000	19.000000	3.0

Data Frames methods

Unlike attributes, python methods have *parenthesis.*
All attributes and methods can be listed with a *dir()* function: `dir(df)`

df.method()	description
head([n]), tail([n])	first/last n rows
max(), min()	return max/min values for all numeric columns
mean(), median()	return mean/median values for all numeric columns
std()	standard deviation
sample([n])	returns a random sample of the data frame
dropna()	drop all the records with missing values

Taking Care of Missing Values

- There is a famous Data Science phrase which you might have heard that is

Garbage in Garbage out

- If your data set is full of **NaNs** and garbage values, then surely your model will perform garbage too. So taking care of such missing values is important. see how we can tackle this problem of taking care of garbage values.
- Let's see the missing values in the data set.

```
In [28]: df.isna().sum()

Out[28]: id_odsp              0
         id_event             0
         sort_order           0
         time                 0
         text                 0
         event_type           0
         event_type2     726716
         side                 0
         event_team           0
         opponent             0
         player           61000
         player2         649699
         player_in       889294
         player_out      889271
         shot_place      713550
         shot_outcome    712511
         is_goal              0
         location        473942
         bodypart        711824
         assist_method        0
         situation       711872
         fast_break           0
         dtype: int64
```

```
In [31]: df = df.dropna()

In [32]: df.isna().sum()

Out[32]: id_odsp          0.0
         id_event         0.0
         sort_order       0.0
         time             0.0
         text             0.0
         event_type       0.0
         event_type2      0.0
         side             0.0
         event_team       0.0
         opponent         0.0
         player           0.0
         player2          0.0
         player_in        0.0
         player_out       0.0
         shot_place       0.0
         shot_outcome     0.0
         is_goal          0.0
         location         0.0
         bodypart         0.0
         assist_method    0.0
         situation        0.0
         fast_break       0.0
         dtype: float64
```


Any
Questions