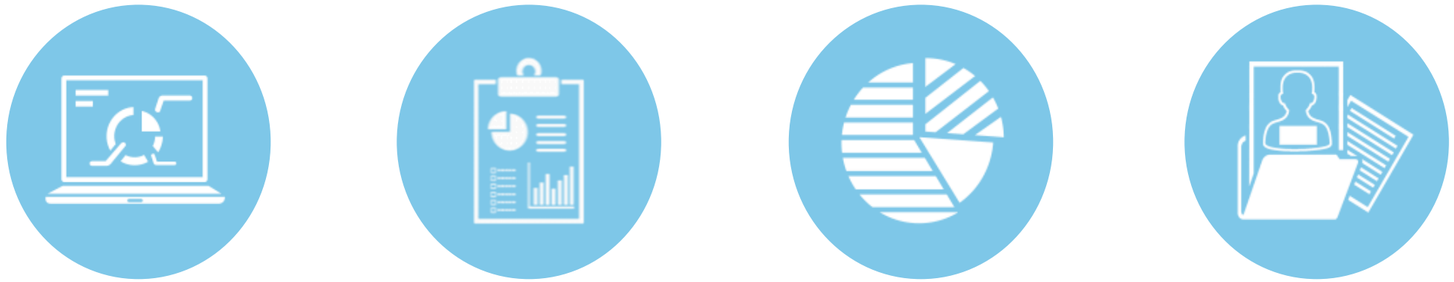
Machine Learning

Lesson 2: Data Wrangling and Manipulation



Concepts Covered

- Data acquisition
- Data exploration techniques
- Data wrangling techniques
- Data manipulation techniques
- Typecasting

Learning Objectives

By the end of this lesson, you will be able to:

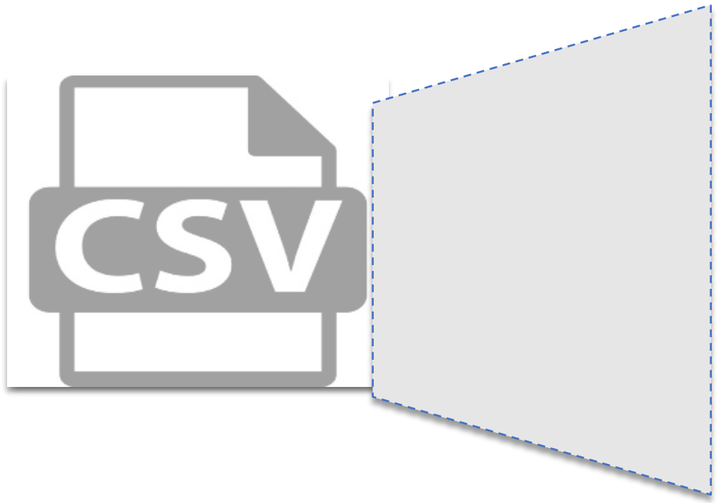
- Demonstrate data import and exploration using Python
- Demonstrate different data wrangling techniques and their significance
- Perform data manipulation in python using coercion, merging, concatenation, and joins

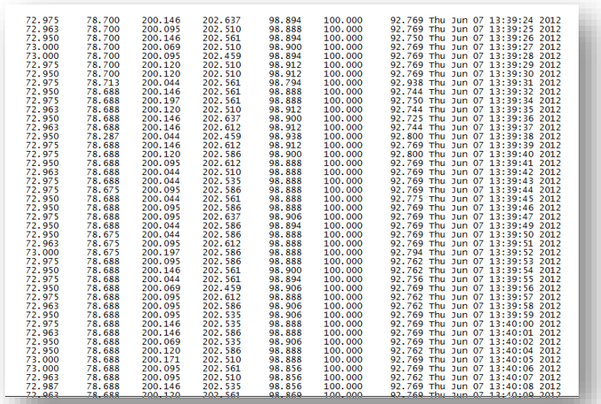
Data Preprocessing

Topic 1: Data Exploration

Loading .csv File in Python

Before starting with a dataset, the first step is to load the dataset. Below is the code for the same:





CSV File

Program Data

Program

Code

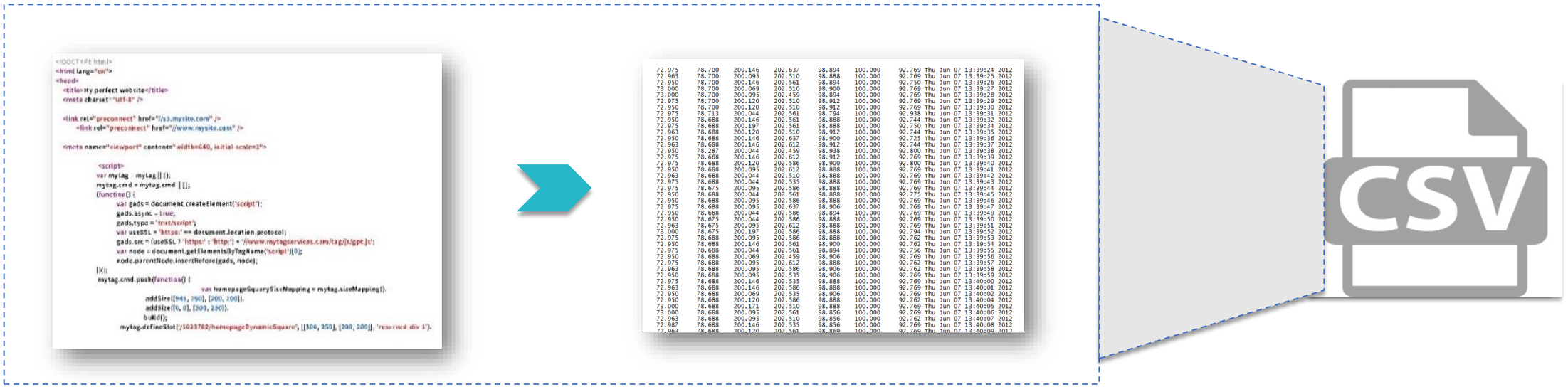
```
df = pandas.read_csv("/home/simpy/Datasets/BostonHousing.csv")
```

Path to file

simplilearn

Loading Data to .csv File

Below is the code for loading the data within an existing csv file:



Program | **Program Data** | **CSV File**

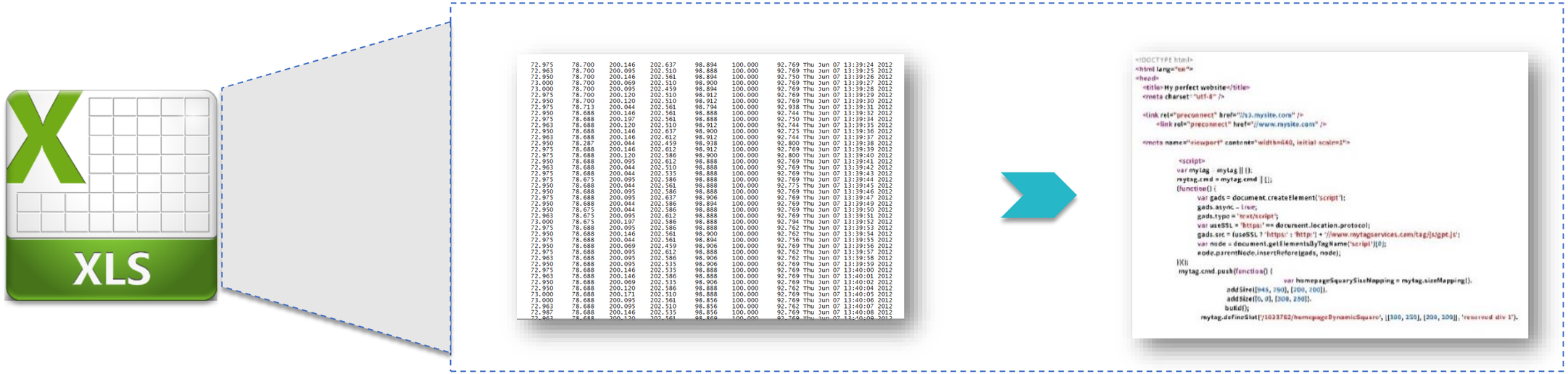
Code

```
df.to_csv("/home/simpy/Datasets/BostonHousing.csv")
```

Path to file

Loading .xlsx File in Python

Below is the code for loading an xlsx file within python:



XLS File

Program Data

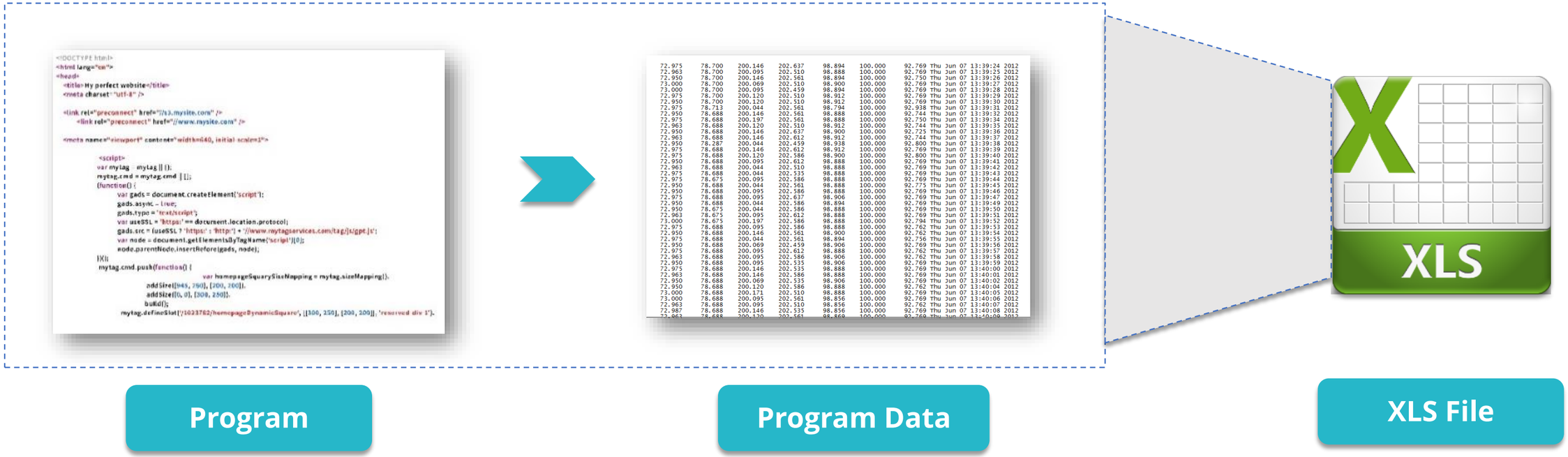
Program

Code

```
df = pandas.read_excel("/home/simpy/Datasets/BostonHousing.xlsx")
```

Loading Data to .xlsx File

Below is the code for loading program data into an existing xlsx file:



Program

Program Data

XLS File

Code

```
df.to_excel("/home/simpy/Datasets/BostonHousing.xlsx")
```

Assisted Practice

Data Exploration

Duration: 5 mins.

Problem Statement: Extract data from the given SalaryGender CSV file and store the data from each column in a separate NumPy array.

Objective: Import the dataset (csv) in/from your Python notebook to local system.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

Data Exploration Techniques

- Dimensionality Check
- Type of Dataset
- Slicing and Indexing
- Identifying Unique Elements
- Value Extraction
- Feature Mean
- Feature Median
- Feature Mode

The shape attribute returns a two-item tuple (number of rows and the number of columns) for the data frame. For a Series, it returns a one-item tuple.

Code

```
df.shape
```

```
Out[12]: (506, 14)
```

Data Exploration Techniques (Contd.)

- Dimensionality Check
- Type of Dataset
- Slicing and Indexing
- Identifying Unique Elements
- Value Extraction
- Feature Mean
- Feature Median
- Feature Mode

You can use the type ( ) in python to return the type of object.

Checking the type of data frame:

Code

```
type(df)
```

```
Out[13]: pandas.core.frame.DataFrame
```

Checking the type of a column (chas) within a data frame:

Code

```
df['chas'].dtype
```

```
Out[21]: dtype('int64')
```

Data Exploration Techniques (Contd.)

- Dimensionality Check
- Type of Dataset
- Slicing and Indexing
- Identifying Unique Elements
- Value Extraction
- Feature Mean
- Feature Median
- Feature Mode

You can use the : operator with the start index on left and end index on right of it to output the corresponding slice.

Slicing a list: `list = [1,2,3,4,5]`

Code

```
list[1:3]
```

```
Out[4]: [2, 3]
```

Slicing a Data frame (df) using iloc indexer:

Code

```
df.iloc[:,1:3]
```

	zn	indus
0	18.0	2.31
1	0.0	7.07
2	0.0	7.07
3	0.0	2.18
4	0.0	2.18
5	0.0	2.18

Data Exploration Techniques (Contd.)

- Dimensionality Check
- Type of Dataset
- Slicing and Indexing
- Identifying Unique Elements
- Value Extraction
- Feature Mean
- Feature Median
- Feature Mode

Using unique ( ) on the column of interest will return a numpy array with unique values of the column.

Extracting all unique values out of "crim" column:

Code

```
df['crim'].unique()
```

```
Out[23]: array([6.32000e-03, 2.73100e-02, 2.72900e-02, 3.23700e-02, 6.90500e-02,
                2.98500e-02, 8.82900e-02, 1.44550e-01, 2.11240e-01, 1.70040e-01,
                2.24890e-01, 1.17470e-01, 9.37800e-02, 6.29760e-01, 6.37960e-01,
                6.27390e-01, 1.05393e+00, 7.84200e-01, 8.02710e-01, 7.25800e-01,
                1.25179e+00, 8.52040e-01, 1.23247e+00, 9.88430e-01, 7.50260e-01,
                8.40540e-01, 6.71910e-01, 9.55770e-01, 7.72990e-01, 1.00245e+00,
                1.13081e+00, 1.35472e+00, 1.38799e+00, 1.15172e+00, 1.61282e+00,
```

Data Exploration Techniques (Contd.)

- Dimensionality Check
- Type of Dataset
- Slicing and Indexing
- Identifying Unique Elements
- Value Extraction
- Feature Mean
- Feature Median
- Feature Mode

Using value ( ) on the column of interest will return a numpy array with all the values of the column.

Extracting values out of "crim" column:

Code

```
df['crim'].values()
```

```
Out[34]: array([6.32000e-03, 2.73100e-02, 2.72900e-02, 3.23700e-02, 6.90500e-02,
                2.98500e-02, 8.82900e-02, 1.44550e-01, 2.11240e-01, 1.70040e-01,
                2.24890e-01, 1.17470e-01, 9.37800e-02, 6.29760e-01, 6.37960e-01,
                6.27390e-01, 1.05393e+00, 7.84200e-01, 8.02710e-01, 7.25800e-01,
                1.25179e+00, 8.52040e-01, 1.23247e+00, 9.88430e-01, 7.50260e-01,
                8.40540e-01, 6.71910e-01, 9.55770e-01, 7.72990e-01, 1.00245e+00,
                1.13081e+00, 1.35472e+00, 1.38799e+00, 1.15172e+00, 1.61282e+00,
                6.41700e-02, 9.74400e-02, 8.01400e-02, 1.75050e-01, 2.76300e-02,
```

Data Exploration Techniques (Contd.)

- Dimensionality Check
- Type of Dataset
- Slicing and Indexing
- Identifying Unique Elements
- Value Extraction
- Feature Mean
- Feature Median
- Feature Mode

Using mean( ) on the data frame will return mean of the data frame across all the columns.

Code

```
df.mean()
```

```
Out[35]: crim        3.613524
         zn         11.363636
         indus      11.136779
         chas        0.069170
         nox         0.554695
         rm          6.284634
         age        68.574901
         dis         3.795043
         rad         9.549407
         tax       408.237154
         ptratio    18.455534
         b         356.674032
         lstat      12.653063
         medv       22.532806
         dtype: float64
```

Data Exploration Techniques (Contd.)

- Dimensionality Check
- Type of Dataset
- Slicing and Indexing
- Identifying Unique Elements
- Value Extraction
- Feature Mean
- Feature Median
- Feature Mode

Using median( ) on the data frame will return median values of the data frame across all the columns.

Code

```
df.median()
```

```
Out[36]: crim          0.25651
         zn            0.00000
         indus         9.69000
         chas          0.00000
         nox           0.53800
         rm            6.20850
         age          77.50000
         dis           3.20745
         rad           5.00000
         tax         330.00000
         ptratio      19.05000
         b           391.44000
         lstat        11.36000
         medv         21.20000
         dtype: float64
```

Data Exploration Techniques (Contd.)

- Dimensionality Check
- Type of Dataset
- Slicing and Indexing
- Identifying Unique Elements
- Value Extraction
- Feature Mean
- Feature Median
- Feature Mode

Using mode( ) on the data frame will return mode values of the data frame across all the columns, rows with axis=0 and axis = 1, respectively.

Code

```
df.mode(axis=0)
```

Out[40]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.01501	0.0	18.1	0.0	0.538	5.713	100.0	3.4952	24.0	666.0	20.2	396.9	6.36	50.0
1	14.33370	NaN	NaN	NaN	NaN	6.127	NaN	NaN	NaN	NaN	NaN	NaN	7.79	NaN
2	NaN	NaN	NaN	NaN	NaN	6.167	NaN	NaN	NaN	NaN	NaN	NaN	8.05	NaN
3	NaN	NaN	NaN	NaN	NaN	6.229	NaN	NaN	NaN	NaN	NaN	NaN	14.10	NaN
4	NaN	NaN	NaN	NaN	NaN	6.405	NaN	NaN	NaN	NaN	NaN	NaN	18.13	NaN
5	NaN	NaN	NaN	NaN	NaN	6.417	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Let’s now consider multiple features and understand the effect of one over other with respect to correlation (using seaborn)

Plotting a Heatmap with Seaborn

Below is the code for plotting a heatmap within Python:

Code

```
import matplotlib.pyplot as plt
import seaborn as sns
correlations = df.corr()
sns.heatmap(data = correlations,square = True, cmap = "bwr")


plt.yticks(rotation=0)
plt.xticks(rotation=90)
```

Rectangular dataset (2D dataset that can be coerced into an ndarray)

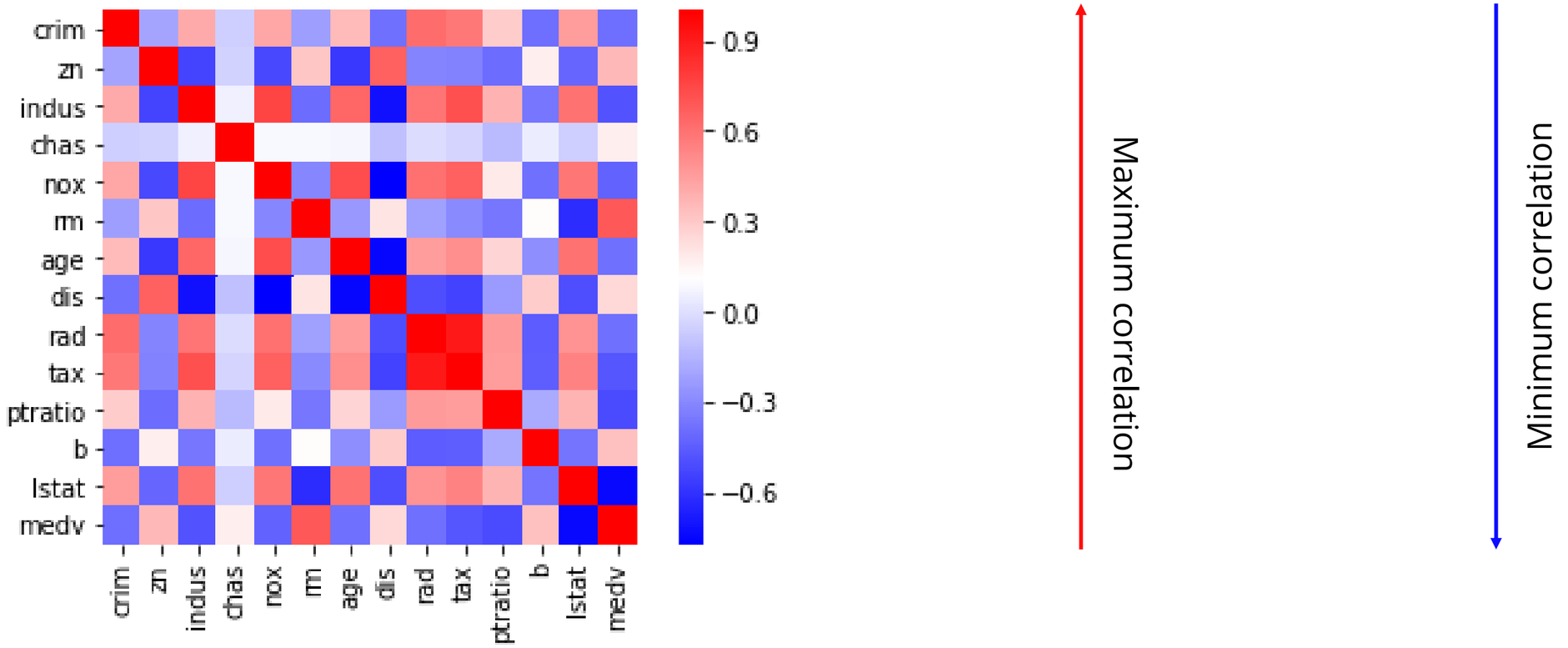
If True, set the Axes aspect to “equal” so each cell will be square-shaped

Matplotlib colormap name or object, or list of colors

Plotting a Heatmap with Seaborn (Contd.)

Below is the heatmap obtained, where, approaching red colour means maximum correlation and approaching blue means minimal correlation.

```
Out[33]: (array([ 0.5,  1.5,  2.5,  3.5,  4.5,  5.5,  6.5,  7.5,  8.5,  9.5, 10.5,
                 11.5, 12.5, 13.5]), <a list of 14 Text xticklabel objects>)
```



Assisted Practice

Data Exploration

Duration: 15 mins.

Problem Statement: Suppose you are a public school administrator. Some schools in your state of Tennessee are performing below average academically. Your superintendent under pressure from frustrated parents and voters approached you with the task of understanding why these schools are under-performing. To improve school performance, you need to learn more about these schools and their students, just as a business needs to understand its own strengths and weaknesses and its customers. The data includes various demographic, school faculty, and income variables.

Objective: Perform exploratory data analysis which includes: determining the type of the data, correlation analysis over the same. You need to convert the data into useful information:

- Read the data in pandas data frame
- Describe the data to find more details
- Find the correlation between 'reduced_lunch' and 'school_rating'

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

Unassisted Practice

Data Exploration

Duration: 15 mins.

Problem Statement: Mtcars, an automobile company in Chambersburg, United States has recorded the production of its cars within a dataset. With respect to some of the feedback given by their customers they are coming up with a new model. As a result of it they have to explore the current dataset to derive further insights out if it.

Objective: Import the dataset, explore for dimensionality, type and average value of the horsepower across all the cars. Also, identify few of mostly correlated features which would help in modification.

Note: This practice is not graded. It is only intended for you to apply the knowledge you have gained to solve real-world problems.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login

Data Import

The first step is to import the data as a part of exploration.

Code

```
df1 = pandas.read_csv("mtcars.csv")
```

Out[35]:

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
5	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
6	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
7	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
8	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
9	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4

Data Exploration

The shape property is usually used to get the current shape of an array/df.

- Dimensionality Check
- Type of Dataset
- Identifying mean value

Code

```
df1.shape
```

```
Out[36]: (32, 12)
```

Data Exploration

type(), returns type of the given object.

- Dimensionality Check
- Type of Dataset
- Identifying mean value

Code

```
type(df1)
```

```
Out[37]: pandas.core.frame.DataFrame
```

Data Exploration

mean( ) function can be used to calculate mean/average of a given list of numbers.

- Dimensionality Check
- Type of Dataset
- Identifying mean value

Code

```
df1['hp'].mean()
```

```
Out[44]: 146.6875
```

Identifying Correlation Using a Heatmap

Heatmap function in seaborn is used to plot the correlation matrix.

Code

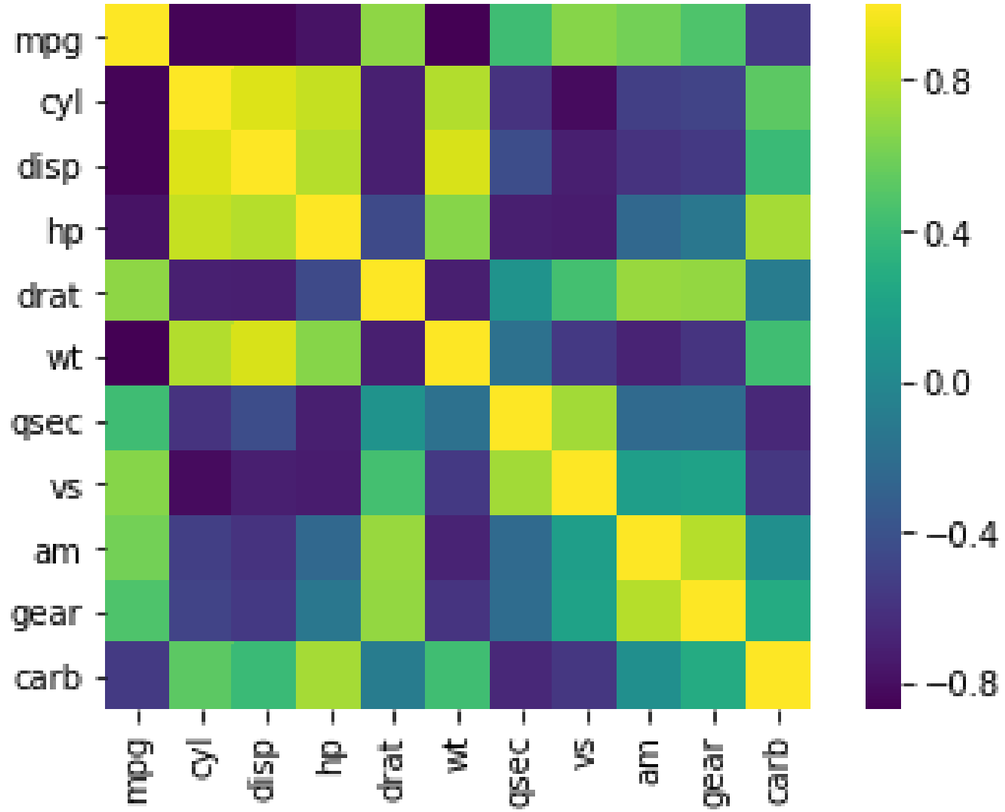
```
import matplotlib.pyplot as plt
import seaborn as sns
correlations = df1.corr()
sns.heatmap(data = correlations,square = True, cmap = "viridis")


plt.yticks(rotation=0)
plt.xticks(rotation=90)
```

Identifying Correlation Using a Heatmap

Graphical representation of data where the individual values contained in a matrix are represented in colors.

```
Out[45]: (array([ 0.5,  1.5,  2.5,  3.5,  4.5,  5.5,  6.5,  7.5,  8.5,  9.5, 10.5]),
          <a list of 11 Text xticklabel objects>)
```



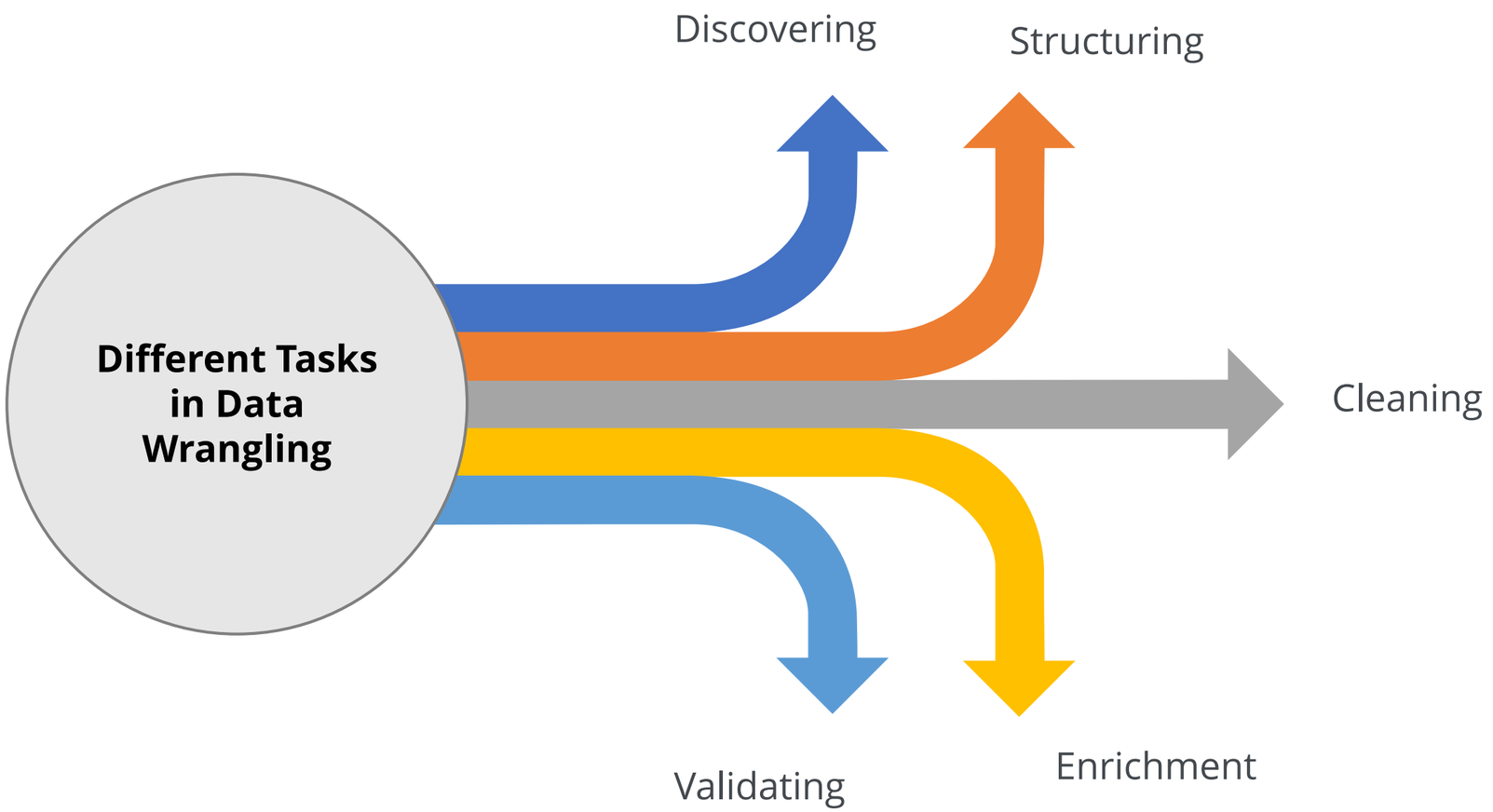
From the adjacent map, you can clearly see that cylinder (cyl) and displacement (disp) are the most correlated features.

Data Preprocessing

Topic 2: Data Wrangling

Data Wrangling

The process of manually converting or mapping data from one raw format into another format is called data wrangling. This includes munging and data visualization.



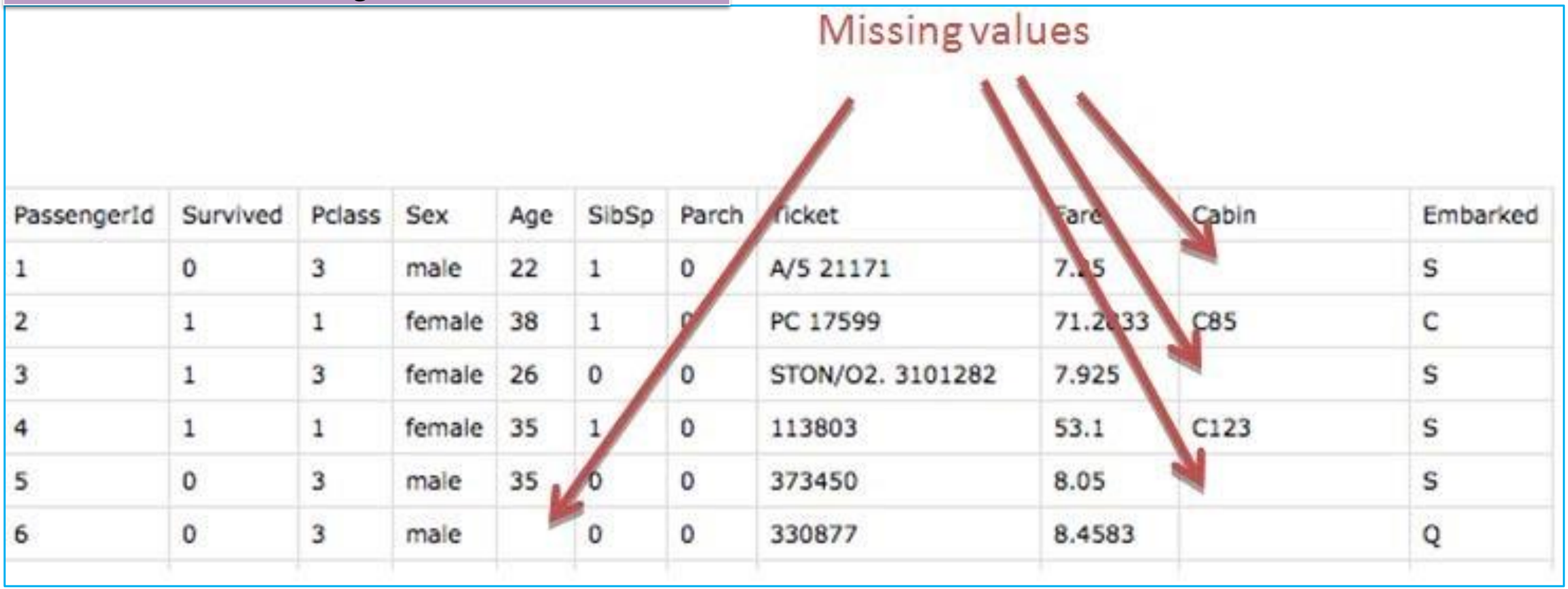
Need of Data Wrangling

Following are the problems that can be avoided with wrangled data:

- Missing data, a very common problem
- Presence of noisy data (erroneous data and outliers)
- Inconsistent data
- Develop a more accurate model
- Prevent data leakage

Missing Values in a Dataset

Consider a random dataset given below, illustrating missing values.



PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	male	22	1	0	A/5 21171	7.25		S
2	1	1	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	female	35	1	0	113803	53.1	C123	S
5	0	3	male	35	0	0	373450	8.05		S
6	0	3	male		0	0	330877	8.4583		Q

Missing Value Detection

Consider a dataset below, imported as df1 within Python, having some missing values.

	Prefix	Assignment	Tutorial	Midterm	TakeHome	Final
0	5	57.14	34.09	64.38	51.48	52.50
1	8	95.05	105.49	67.50	99.07	68.33
2	8	83.70	83.17	30.00	63.15	48.89
3	7	81.22	96.06	49.38	105.93	80.56
4	8	91.32	93.64	95.00	107.41	73.89
5	7	95.00	92.58	93.12	97.78	68.06
6	8	95.05	102.99	56.25	99.07	50.00
7	7	72.85	86.85	60.00	NaN	56.11
8	8	84.26	93.10	47.50	18.52	50.83

Detecting missing values

Code

```
df1.isna().any()
```

```
Out[16]: Prefix        False
         Assignment    False
         Tutorial      False
         Midterm       False
         TakeHome       True
         Final         False
         dtype: bool
```

Missing Value Treatment

Mean Imputation: Replace the missing value with variable's mean

Code

```
from sklearn.preprocessing import Imputer
mean_imputer =
Imputer(missing_values=np.nan,strategy='mean',axis=1)
mean_imputer = mean_imputer.fit(df1)
imputed_df = mean_imputer.transform(df1.values)
df1 = pd.DataFrame(data=imputed_df,columns=cols)
df1
```

Out[75]:

	Prefix	Assignment	Tutorial	Midterm	TakeHome	Final
0	5.0	57.14	34.09	64.38	51.480	52.50
1	8.0	95.05	105.49	67.50	99.070	68.33

Missing Value Treatment (Contd.)

Mean Imputation: Replace the missing value with variable's mean

Median Imputation: Replace the missing value with variable's median

Code

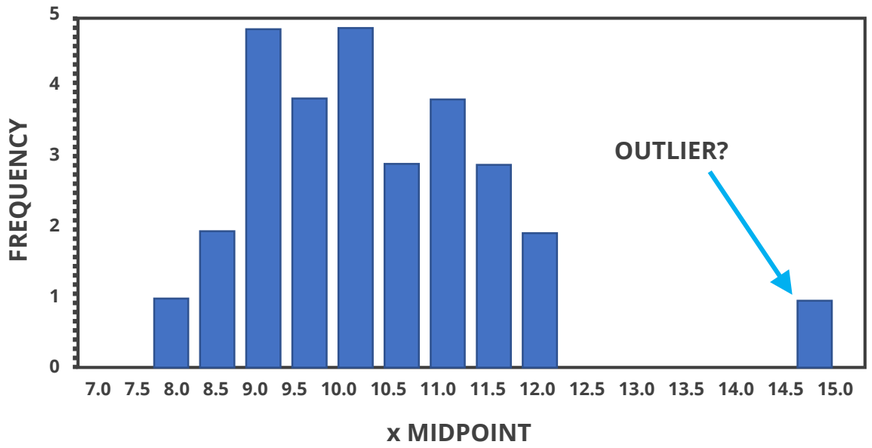
```
from sklearn.preprocessing import Imputer
median_imputer=Imputer(missing_values=np.nan,strategy
='median',axis=1)
median_imputer = median_imputer.fit(df1)
imputed_df = median_imputer.transform(df1.values)
df1 = pd.DataFrame(data=imputed_df,columns=cols)
df1
```

Out[84]:

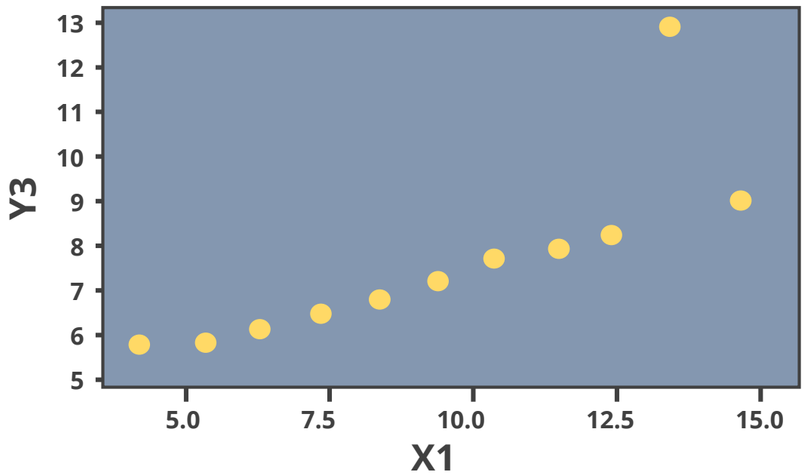
	Prefix	Assignment	Tutorial	Midterm	TakeHome	Final
0	5.0	57.14	34.09	64.38	51.480	52.50
1	8.0	95.05	105.49	67.50	99.070	68.33

Note: *Mean imputation/Median imputation is again model dependent and is valid only on numerical data.*

Outlier Values in a Dataset



An outlier is a value that lies outside the usual observation of values.



Note: *Outliers skew the data when you are trying to do any type of average.*

Dealing with an Outlier

Outlier Detection

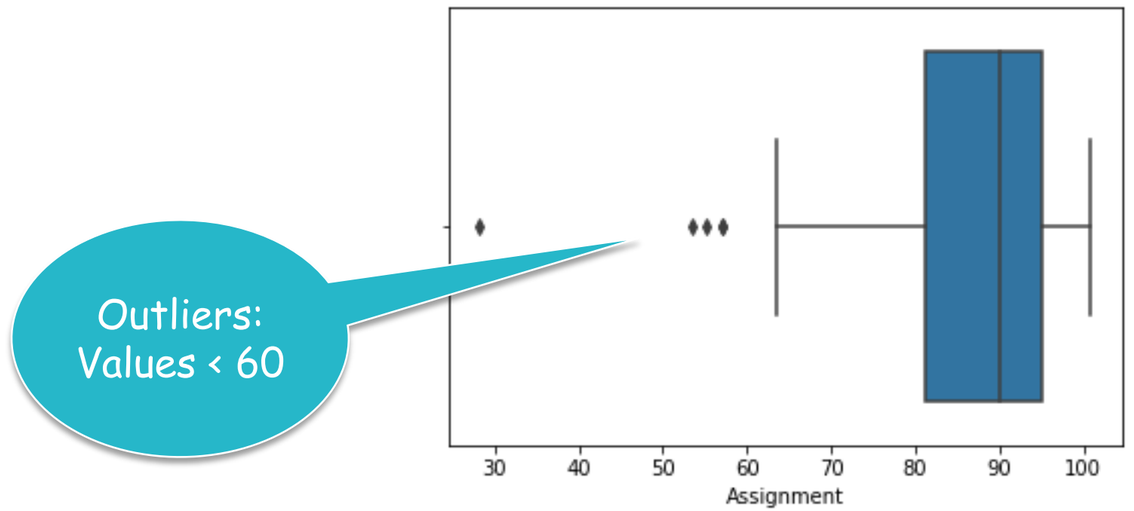
Outlier Treatment

Detect any outlier in the first column of df1

Code

```
import seaborn as sns
sns.boxplot(x=df1['Assignment'])
```

```
Out[88]: <matplotlib.axes._subplots.AxesSubplot at 0x232d5273da0>
```



Dealing with an Outlier

Outlier Detection

Outlier Treatment

Create a filter based on the boxplot obtained and apply the filter to the data frame

Code

```
filter=df1['Assignment'].values>60
df1_outlier_rem=df1[filter]
df1_outlier_rem
```

Out[106]:

	Prefix	Assignment	Tutorial	Midterm	TakeHome	Final
1	8.0	95.05	105.49	67.50	99.070	68.33
2	8.0	83.70	83.17	30.00	63.150	48.89
3	7.0	81.22	96.06	49.38	105.930	80.56
4	8.0	91.32	93.64	95.00	107.410	73.89
5	7.0	95.00	92.58	93.12	97.780	68.06
6	8.0	95.05	102.99	56.25	99.070	50.00
7	7.0	72.85	86.85	60.00	56.562	56.11

Assisted Practice

Data Wrangling

Duration: 15 mins.

Problem Statement: Load the load_diabetes datasets internally from sklearn and check for any missing value or outlier data in the ‘data’ column. If any irregularities found treat them accordingly.

Objective: Perform missing value and outlier data treatment.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

Unassisted Practice

Data Wrangling

Duration: 5 mins.

Problem Statement: Mtcars, the automobile company in the United States have planned to rework on optimizing the horsepower of their cars, as most of the customers feedbacks were centred around horsepower. However, while developing a ML model with respect to horsepower, the efficiency of the model was compromised. Irregularity might be one of the causes.

Objective: Check for missing values and outliers within the horsepower column and remove them.

Note: This practice is not graded. It is only intended for you to apply the knowledge you have gained to solve real-world problems.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

Check for Irregularities

Check for missing values

Code

```
df1['hp'].isna().any()
```

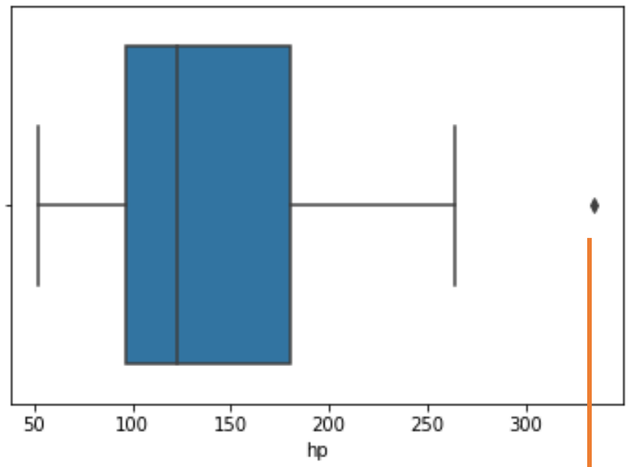
```
Out[114]: False
```

Check for Outliers

Code

```
sns.boxplot(x=df1['hp'])
```

```
Out[113]: <matplotlib.axes._subplots.AxesSubplot at 0x232d55e6f98>
```



Outlier

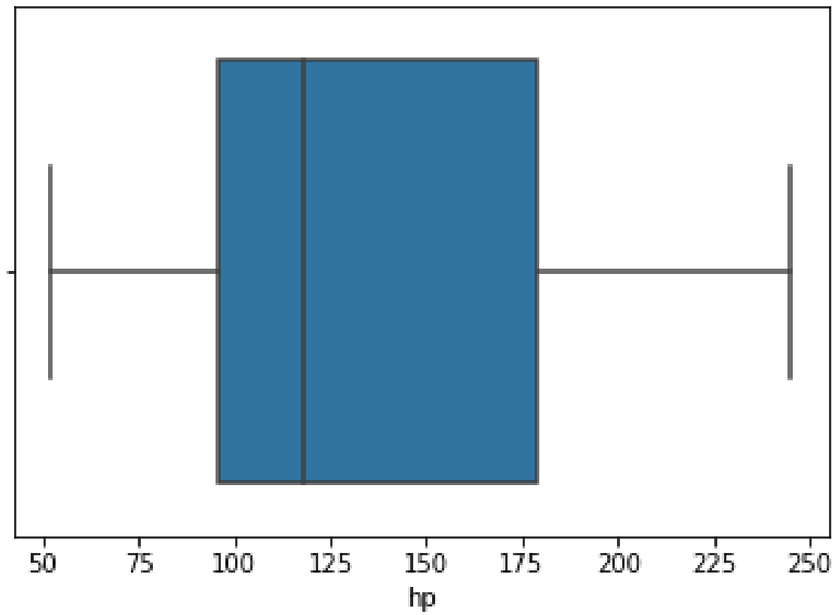
Outlier Treatment

Data with hp>250 is the outlier data. Therefore, you can filter it accordingly.

Code

```
filter = df1['hp']<250
df1_out_rem = df1[filter]
sns.boxplot(x=df2_out_rem['hp'])
```

```
Out[120]: <matplotlib.axes._subplots.AxesSubplot at 0x232d52d6470>
```



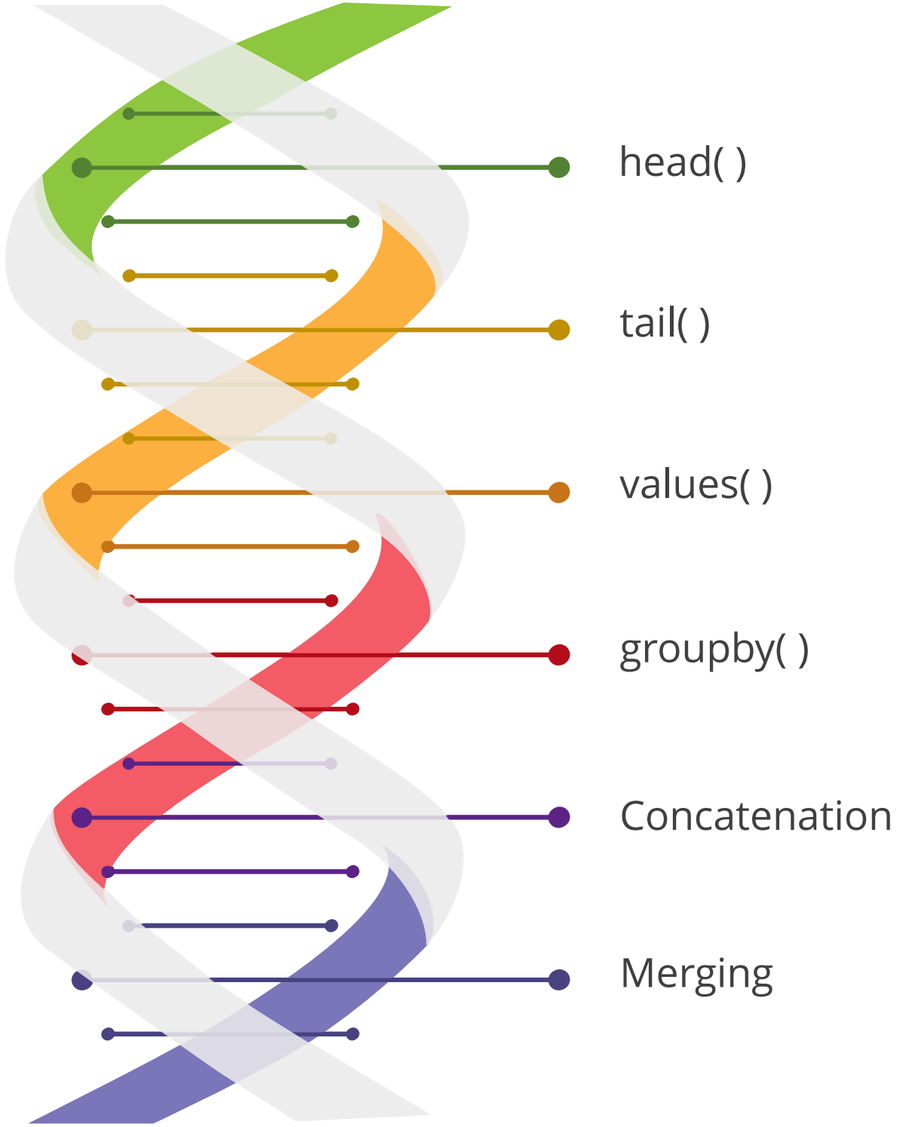
Outlier filtered data

Data Preprocessing

Topic 3: Data Manipulation

Functionalities of Data Object in Python

A data object is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.



Functionalities of Data Object in Python (Contd.)

- head( )
- tail( )
- values( )
- groupby( )
- Concatenation
- Merging

Head( ) returns the first n rows of the data structure

Code

```
import pandas as pd
import numpy as np
df=pd.Series(np.arange(1,51))
print(df.head(6))
```

```
0    1
1    2
2    3
3    4
4    5
5    6
dtype: int64
```

Functionalities of Data Object in Python (Contd.)

- head( )
- tail( )
- values( )
- groupby( )
- Concatenation
- Merging

Tail( ) returns the last n rows of the data structure

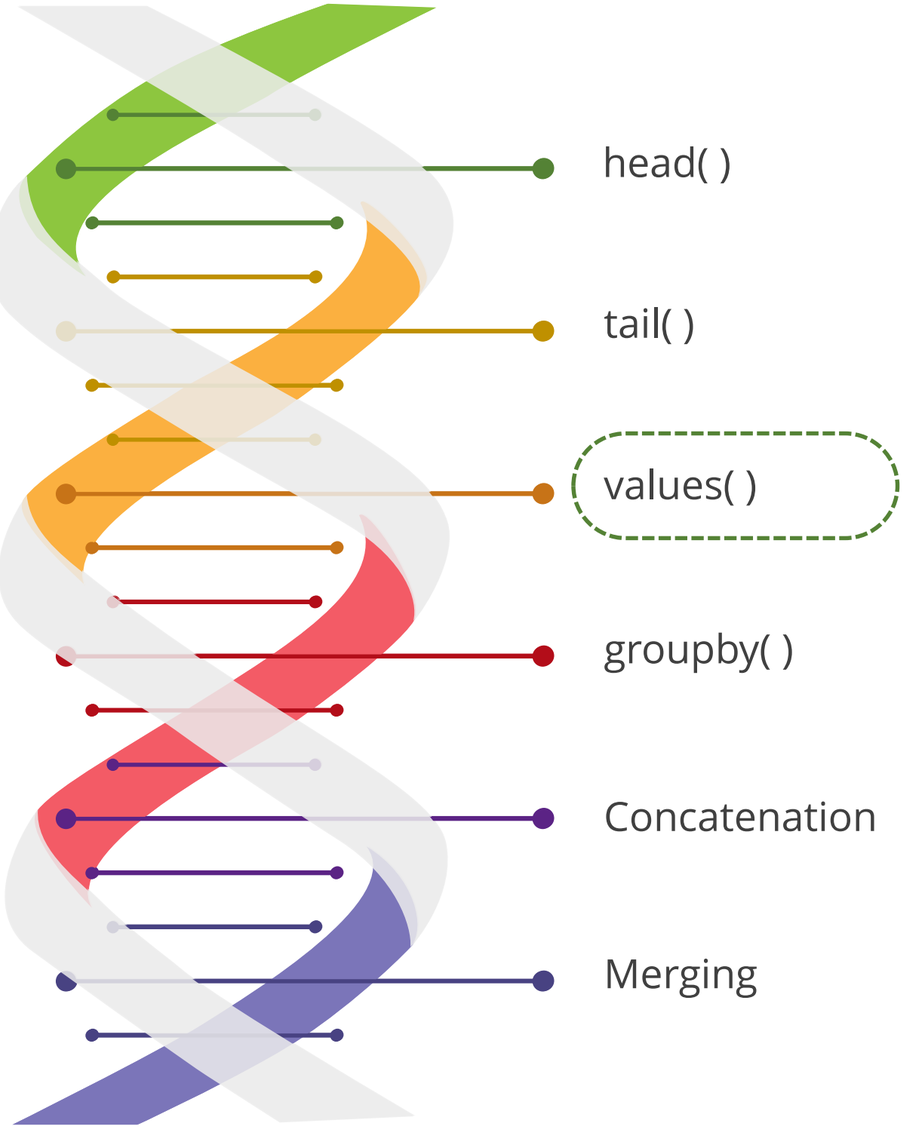
Code

```
import pandas as pd
import numpy as np

df=pd.Series(np.arange(1,51))
print(df.tail(6))
```

```
44    45
45    46
46    47
47    48
48    49
49    50
dtype: int64
```

Functionalities of Data Object in Python (Contd.)



- head( )
- tail( )
- values( )
- groupby( )
- Concatenation
- Merging

values( ) returns the actual data in the series of the array

Code

```
import pandas as pd
import numpy as np

df=pd.Series(np.arange(1,51))
print(df.values)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50]
```

Functionalities of Data Object using Python (Contd.)

- head( )
- tail( )
- values( )
- groupby( )
- Concatenation
- Merging

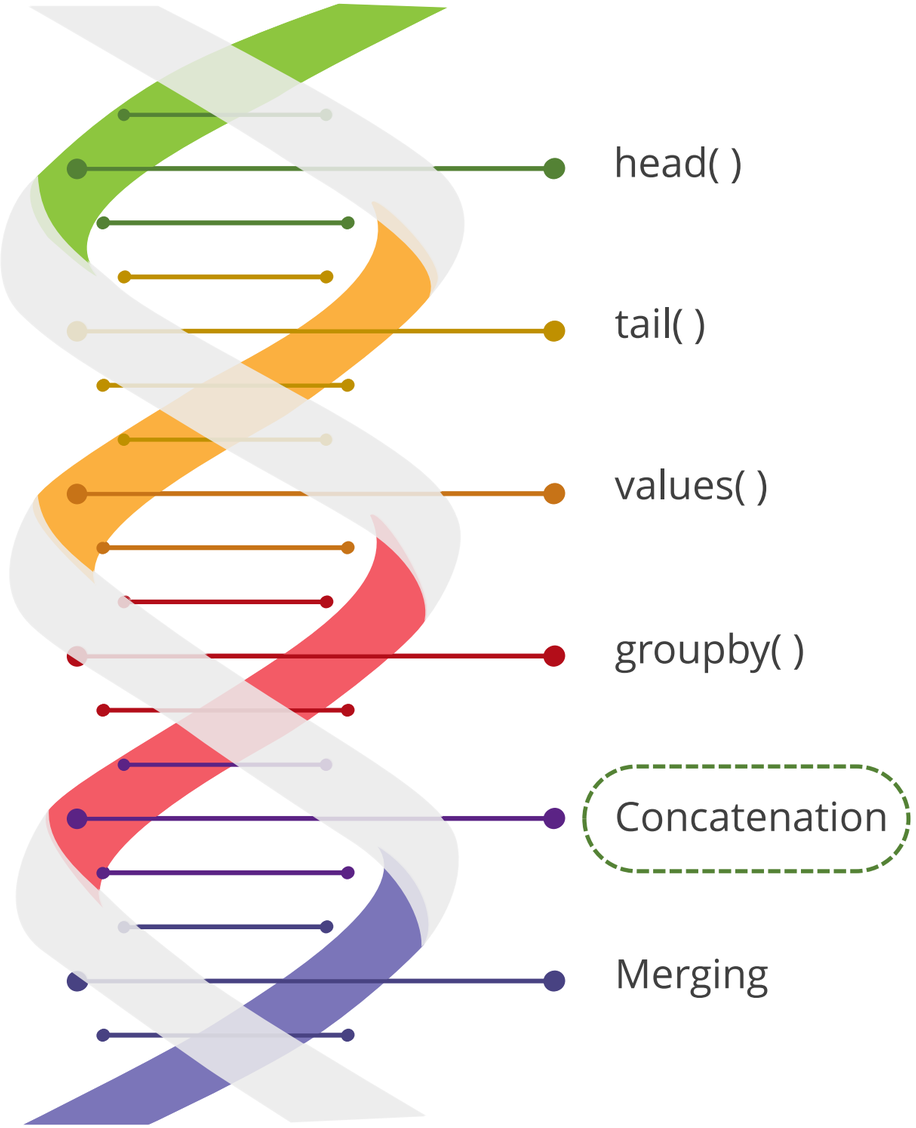
The Data Frame is grouped according to the 'Team' and 'ICC_Rank' columns

Code

```
import pandas as pd
world_cup={'Team':['West Indies','West
indies','India','Australia','Pakistan','Sri
   Lanka','Australia','Australia','Australia','
Insia','Australia'],
           'Rank':[7,7,2,1,6,4,1,1,1,2,1],
'Year':[1975,1979,1983,1987,1992,1996,1999,2003
,2007,2011,2015]}
df=pd.DataFrame(world_cup)
print(df.groupby(['Team','Rank']).groups)
```

```
{('Australia', 1): Int64Index([3, 6, 7, 8, 10], dtype='int64'), ('India', 2): Int64Index([2], dtype='int64'), ('Insia', 2): Int
64Index([9], dtype='int64'), ('Pakistan', 6): Int64Index([4], dtype='int64'), ('Sri Lanka', 4): Int64Index([5], dtype='int64'),
('West Indies', 7): Int64Index([0], dtype='int64'), ('West indies', 7): Int64Index([1], dtype='int64')}
```

Functionalities of Data Object in Python (Contd.)



Concatenation combines two or more data structures.

Code

```
import pandas
world_champions={'Team':['India','Australia','West
Indies','Pakistan','Sri Lanka’],
'ICC_rank':[2,3,7,8,4],
'World_champions_Year':[2011,2015,1979,1992,1996],
            'Points':[874,787,753,673,855]}
chokers={'Team':['South Africa','New
Zealand','Zimbabwe'],'ICC_rank':[1,5,9],
'Points':[895,764,656]}
df1=pandas.DataFrame(world_champions)
df2=pandas.DataFrame(chokers)
print(pandas.concat([df1,df2],axis=1))
```

Functionalities of Data Object in Python (Contd.)

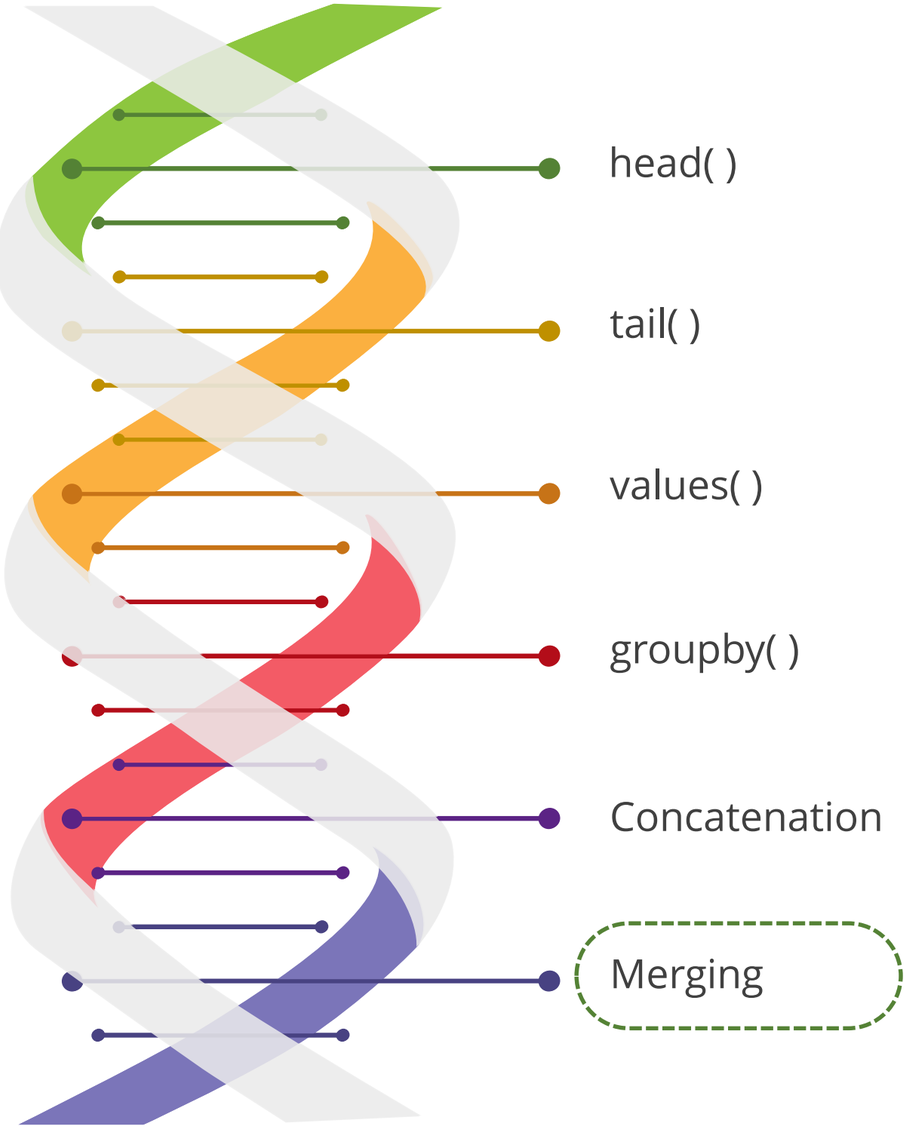
- head( )
- tail( )
- values( )
- groupby( )
- Concatenation
- Merging

The concatenated output:

	ICC_rank	Points	Team	World_champions_Year	ICC_rank	Points	\
0	2	874	India	2011	1.0	895.0	
1	3	787	Australia	2015	5.0	764.0	
2	7	753	West Indies	1979	9.0	656.0	
3	8	673	Pakistan	1992	NaN	NaN	
4	4	855	Sri Lanka	1996	NaN	NaN	

	Team
0	South Africa
1	New Zealand
2	Zimbabwe
3	NaN
4	NaN

Functionalities of Data Object in Python (Contd.)



Merging is the Pandas operation that performs database joins on objects

Code

```
import pandas
champion_stats={'Team':['India','Australia','West
Indies','Pakistan','Sri Lanka'],
            'ICC_rank':[2,3,7,8,4],
'World_champions_Year':[2011,2015,1979,1992,1996],
            'Points':[874,787,753,673,855]}
match_stats={'Team':['India','Australia','West
Indies','Pakistan','Sri Lanka'],
              'World_cup_played':[11,10,11,9,8],
              'ODIs_played':[733,988,712,679,662]}
df1=pandas.DataFrame(champion_stats)
df2=pandas.DataFrame(match_stats)
print(df1)
print(df2)
print(pandas.merge(df1,df2,on='Team'))
```

Functionalities of Data Object in Python (Contd.)

- head( )
- tail( )
- values( )
- groupby( )
- Concatenation
- Merging

	ICC_rank	Points	Team	World_champions_Year	ODIs_played	\
0	2	874	India	2011	733	
1	3	787	Australia	2015	988	
2	7	753	West Indies	1979	712	
3	8	673	Pakistan	1992	679	
4	4	855	Sri Lanka	1996	662	

	World_cup_played
0	11
1	10
2	11
3	9
4	8

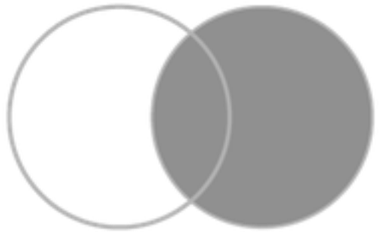
The merged object contains all the columns of the data frames merged

Different Types of Joins

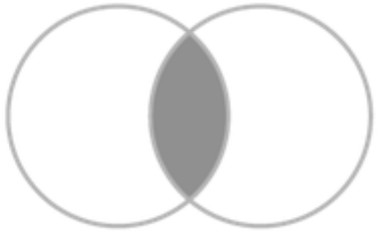
Joins are used to combine records from two or more tables in a database. Below are the four most commonly used joins:

Left Join

Right Join



Inner Join



Full Outer Join

Left Join

Left Join

Returns all rows from the **left** table, even if there are no matches in the right table

Code

```
import pandas
world_champions={'Team':['India','Australia','West
Indies','Pakistan','Sri Lanka'],
            'ICC_rank':[2,3,7,8,4],
'World_champions_Year':[2011,2015,1979,1992,1996],
            'Points':[874,787,753,673,855]}
chokers={'Team':['South Africa','New
Zealand','Zimbabwe'],
'ICC_rank':[1,5,9],'Points':[895,764,656]}
df1=pandas.DataFrame(world_champions)
df2=pandas.DataFrame(chokers)
print(pandas.merge(df1,df2,on='Team',how='left'))
```

```
   ICC_rank_x  Points_x         Team  World_champions_Year  ICC_rank_y  \
0           2       874        India                  2011         NaN
1           3       787    Australia                  2015         NaN
2           7       753  West Indies                  1979         NaN
3           8       673     Pakistan                  1992         NaN
4           4       855    Sri Lanka                  1996         NaN

   Points_y
0       NaN
1       NaN
2       NaN
3       NaN
4       NaN
```

Right Join

Right Join

Preserves the unmatched rows from the second (right) table, joining them with a NULL in the shape of the first (left) table

Code

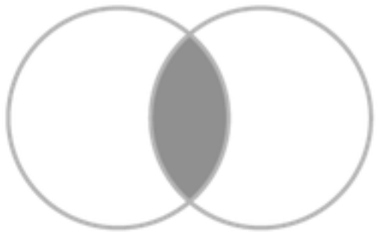
```
import pandas
world_champions={'Team':['India','Australia','West
Indies','Pakistan','Sri Lanka'],
            'ICC_rank':[2,3,7,8,4],
'World_champions_Year':[2011,2015,1979,1992,1996],
            'Points':[874,787,753,673,855]}
chokers={'Team':['South Africa','New
Zealand','Zimbabwe'],'ICC_rank':[1,5,9],'Points':[89
5,764,656]}
df1=pandas.DataFrame(world_champions)
df2=pandas.DataFrame(chokers)
print(pandas.merge(df1,df2,on='Team',how='right'))
```

```
   ICC_rank_x  Points_x          Team  World_champions_Year  ICC_rank_y  \
0         NaN       NaN  South Africa                   NaN           1
1         NaN       NaN   New Zealand                   NaN           5
2         NaN       NaN      Zimbabwe                   NaN           9

   Points_y
0       895
1       764
2       656
```

Inner Join

Inner Join



Selects all rows from both participating tables if there is a match between the columns

Code

```
import pandas
world_champions={'Team':['India','Australia','West
Indies','Pakistan','Sri Lanka'],
           'ICC_rank':[2,3,7,8,4],
'World_champions_Year':[2011,2015,1979,1992,1996],
           'Points':[874,787,753,673,855]}
chokers={'Team':['South Africa','New
Zealand','Zimbabwe'],'ICC_rank':[1,5,9],'Points':[89
5,764,656]}
df1=pandas.DataFrame(world_champions)
df2=pandas.DataFrame(chokers)
print(pandas.merge(df1,df2,on='Team',how='inner'))
```

```
Empty DataFrame
Columns: [ICC_rank_x, Points_x, Team, World_champions_Year, ICC_rank_y, Points_y]
Index: []
```

Full Outer Join

Full Outer Join

Returns all records when there is a match in either left (table1) or right (table2) table records

Code

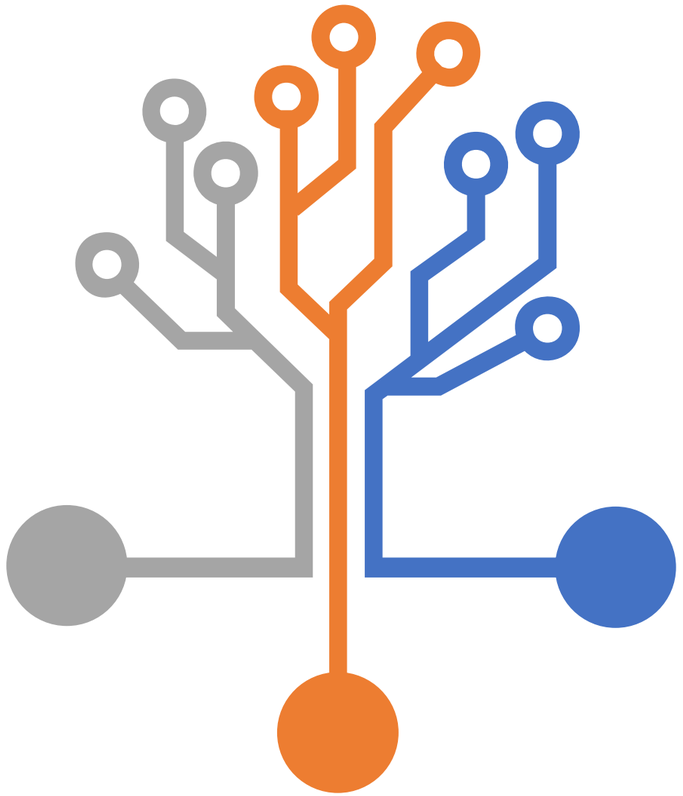
```
import pandas
world_champions={'Team':['India','Australia','West
Indies','Pakistan','Sri Lanka'],
            'ICC_rank':[2,3,7,8,4],
'World_champions_Year':[2011,2015,1979,1992,1996],
            'Points':[874,787,753,673,855]}
chokers={'Team':['South Africa',’New
Zealand','Zimbabwe'],'ICC_rank':[1,5,9],'Points':[89
5,764,656]}
df1=pandas.DataFrame(world_champions)
df2=pandas.DataFrame(chokers)
print(pandas.merge(df1,df2,on='Team',how='outer'))
```

```
   ICC_rank_x  Points_x          Team  World_champions_Year  ICC_rank_y  \
0         2.0     874.0         India                2011.0         NaN
1         3.0     787.0     Australia                2015.0         NaN
2         7.0     753.0   West Indies                1979.0         NaN
3         8.0     673.0      Pakistan                1992.0         NaN
4         4.0     855.0     Sri Lanka                1996.0         NaN
5         NaN       NaN  South Africa                   NaN         1.0
6         NaN       NaN   New Zealand                   NaN         5.0
7         NaN       NaN      Zimbabwe                   NaN         9.0

   Points_y
0       NaN
1       NaN
2       NaN
3       NaN
4       NaN
5     895.0
6     764.0
7     656.0
```

Typecasting

It converts the data type of an object to the required data type.



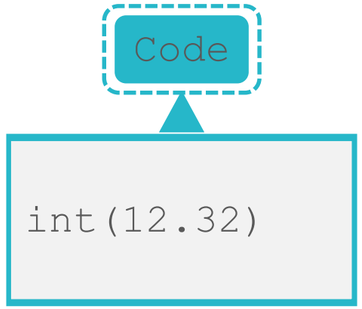
string( )
Returns string from any numeric object or converts any number to string

Int( )
Returns an integer object from any number or string.

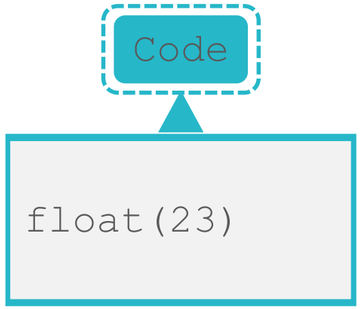
float( )
Returns a floating-point number from a number or a string

Typecasting Using Int, float and string( )

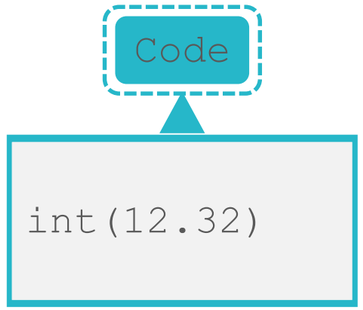
Few typecasted data types



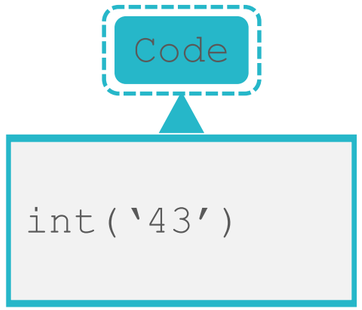

```
Out[121]: 12
```



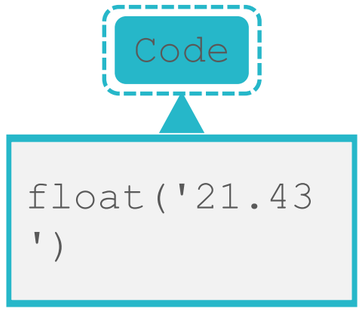

```
Out[123]: 23.0
```




```
Out[125]: '21'
```




```
Out[122]: 43
```




```
Out[124]: 21.43
```

Assisted Practice

Data Manipulation

Duration: 10 mins.

Problem Statement: As a macroeconomic analyst at the Organization for Economic Cooperation and Development (OECD), your job is to collect relevant data for analysis. It looks like you have three countries in the north_america data frame and one country in the south_america data frame. As these are in two separate plots, it's hard to compare the average labor hours between North America and South America. If all the countries were into the same data frame, it would be much easier to do this comparison.

Objective: Demonstrate concatenation.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

Unassisted Practice

Data Manipulation

Duration: 10 mins.

Problem Statement: SFO Public Department - referred to as SFO has captured all the salary data of its employees from year 2011-2014. Now in 2018 the organization is facing some financial crisis. As a first step HR wants to rationalize employee cost to save payroll budget. You have to do data manipulation and answer the below questions:

1. How much total salary cost has increased from year 2011 to 2014?
2. Who was the top earning employee across all the years?

Objective: Perform data manipulation and visualization techniques

Note: This practice is not graded. It is only intended for you to apply the knowledge you have gained to solve real-world problems.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

Answer 1

Check the mean salary cost per year and see how it has increased per year.

Code

```
salary = pd.read_csv('Salaries.csv')
mean_year =
salary.groupby('Year').mean()['TotalPayBenefits']
print ( mean_year)
```

```
Year
2011     71743.819645
2012    100551.886807
2013    101440.519714
2014    100261.438668
Name: TotalPayBenefits, dtype: float64
```

Answer 2

Group the total salary with respect to employee name:

Code

```
top_sal =
salary.groupby('EmployeeName').sum()['TotalPayBenefi
ts']
print((top_sal.sort_values(axis=0)))
```

```
EmployeeName
Joe Lopez                    -618.13
David P Kucia                 -33.89
Mark E Laherty                 -8.20
Timothy E Gibson               -2.73
Mark W Mcclure                  0.00
PAULETTE ADAMS                  0.00
KAUKAB MOHSIN                   0.00
JOSEPHINE MCCREARY              0.00
Charlene D Mccully              0.00
JOE BROWN JR                    0.30
```

Key Takeaways

Now, you are able to:

- Demonstrate data import and exploration using Python
- Demonstrate different data wrangling techniques and their significance
- Perform data manipulation in python using coercion, merging, concatenation, and joins

Knowledge Check

Knowledge Check 1	Which of the following plots can be used to detect an outlier?

a. Boxplot

b. Histogram

c. Scatter plot

d. All of the above

Knowledge Check 1	Which of the following plots can be used to detect an outlier?

a. Boxplot

b. Histogram

c. Scatter plot

d. All of the above

The correct answer is **d . All of the above**

All the above plots can be used to detect an outlier.

Knowledge Check

2

What is the output of the below Python code?

```
import numpy as np percentiles = [98, 76.37, 55.55, 69, 88]
first_subject = np.array(percentiles) print first_subject.dtype
```

a. float32

b. float

c. int32

d. float64

Knowledge Check

2

What is the output of the below Python code?

```
import numpy as np
percentiles = [98, 76.37, 55.55, 69, 88]
first_subject = np.array(percentiles)
print first_subject.dtype
```

a. float32

b. float

c. int32

d. float64

The correct answer is **d. float64**

Float64's can represent numbers much more accurately than other floats and has more storage capacity.

Lesson-End Project

Duration: 20 mins.

Problem Statement: From the raw data below create a data frame:
'first_name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'], 'last_name': ['Miller', 'Jacobson', ".", 'Milner', 'Cooze'],
'age': [42, 52, 36, 24, 73], 'preTestScore': [4, 24, 31, ".", "."],'postTestScore': ["25,000", "94,000", 57, 62, 70]

Objective: Perform data processing on raw data:

- Save the data frame into a csv file as project.csv
- Read the project.csv and print the data frame
- Read the project.csv without column heading
- Read the project.csv and make the index columns as 'First Name' and 'Last Name'
- Print the data frame in a Boolean form as True or False. True for Null/ NaN values and false for non-null values
- Read the data frame by skipping first 3 rows and print the data frame

Access: Click the Labs tab in the left side panel of the LMS. Copy or note the username and password that are generated. Click the Launch Lab button. On the page that appears, enter the username and password in the respective fields and click Login.

Thank You