Unit III

Working with Data, Simple Linear Regression, R programming basics

Reading Data

- File format:
  - Standard way in which data is collected and stored
  - Most commonly used format for storing data is the spreadsheet format where data is stored in rows and columns
    - Each row is called a record
    - Each column in a spreadsheet holds data belonging to same data type
  - Commonly used spreadsheet formats are
    - Comma separated values and excel sheets
  - Other formats include
    - plain text, json, html, mp3,mp4 etc.

Spreadsheet Format

- Format .csv’
  - Each record is separated by a comma
  - Files where records are separated using a tab are called tab separated values
  - .csv files can be opened with notepad or Microsoft excel

Importing data

- Importing necessary libraries
  - import pandas as pd
- Read csv file
  - data= pd.read_csv(“Iris_data_sample.csv”)
- Blank cells read as ‘nan’
- Removing the extra id column by passing
  - data= pd.read_csv(“Iris_data_sample.csv”, index_col=0)
- Junk values can be converted to missing values by passing them as a list to the parameter ‘na_values ’
  - data= pd.read_csv(“Iris_data_sample.csv”, index_col=0,na_values=[“??”,”###”])

Index	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-set…
2	4.9	nan	1.4	0.2	nan
3	4.7	3.2	1.3	0.2	Iris-set…
4	??	3.1	1.5	0.2	Iris-set…
5	5	3.6	###	0.2	Iris-set…

- For Excel spread sheets use
  - data= pd.read_excel("Iris_data_sample.xls", index_col=0,na_values=["??","###"])
- For text format with default tab separator use
  - data= pd.read_table("Iris_data_sample.txt", index_col=0,na_values=["??","###"])
- For other delimiter
  - data= pd.read_excel("Iris_data_sample.txt", delimiter=" ")

Why Pandas?

- Provides high-performance, easy-to-use data structures and analysis tools for the Python programming language
- Open-source Python library providing high- performance data manipulation and analysis tool using its powerful data structures
- Name pandas is derived from the word Panel Data – an econometrics term for multidimensional data
- Pandas deals with dataframes: it's two dimentional mutable data structure having labeled axes
- We will use pandas and numpy for data manipulation and numeric operations

Creating Copy of data

- In Python, there are two ways to create copies
- Shallow copy
- Deep copy
- Cars_data=pd.read_csv("Toyota.csv")

	Shallow copy	**Deep Copy**
Function	data = cars_data.copy(deep=false)	data = cars_data.copy(deep=false)
Description	• It only creates a new variable that shares the reference of the original object • Any changes made to a copy of object will be reflected in the original object as well	• In case of deep copy, a copy of object is copied in other object with no reference to the original • Any changes made to a copy of object will not be reflected in the original object

Attributes of data

To get the index (row labels) of the dataframe	• DataFrame.index • cars_data.index
To get the column labels of the dataframe	• DataFrame.columns • cars_data.columns
To get the total number of elements from the dataframe	• DataFrame.size • cars_data.size
To get the dimensionality of the dataframe	• DataFrame.shape • cars_data.shape
The memory usage of each column in bytes	• DataFrame.memory_usage([index, deep]) • cars_data.memory_usage()
The number of axes / array dimensions	• DataFrame.ndim • cars_data.ndim

- The memory usage of each column in bytes
  - DataFrame.memory_usage([index, deep])
  - cars_data.memory_usage()
- The number of axes / array dimensions
  - DataFrame.ndim
  - cars_data.ndim

Indexing and selecting data

- Python slicing operator '[ ]' and attribute/dot operator '.' are used for indexing
- Provides quick and easy access to pandas data structures
- **head():** The function head returns the first n rows from the dataframe (By default it returns first 5 rows)
  - DataFrame.head([n])
  - cars_data.head(10)
- **tail():** The function tail returns the last n rows for the object based on position(By default it returns last 5 rows)
  - DataFrame.tail([n])
  - cars_data.tail(10)
  - It is useful for quickly verifying data
  - Ex: after sorting or appending rows.

- **at() & iat():** To access a scalar value, the fastest way is to use the at and iat methods
  - at provides label-based scalar lookups
    - cars_data.at[4,'FuelType']
  - iat provides integer-based lookups
    - cars_data.iat[5,6]
- loc(): To access a group of rows and columns by label(s) .loc[ ] can be used
  - cars_data.loc[:, 'FuelType']

Data types

- The way information gets stored in a dataframe or a python object affects the analysis and outputs of calculations
- There are two main types of data
  - numeric and character types
- Numeric data types includes integers and floats
  - For example: integer – 10, float – 10.53
- Strings are known as objects in pandas which can store values that contain numbers and / or characters
  - For example:‘category1’

- **Numeric types in Pandas:**
  - int64: Numeric characters
  - float64: Numeric characters with decimals
  - ‘64’ simply refers to the memory allocated to store data in each cell which effectively relates to how many digits it can store in each “cell”
  - 64 bits is equivalent to 8 bytes
  - Allocating space ahead of time allows computers to optimize storage and processing efficiency

- **Character types:** Category and object

category	object
A string variable consisting of only a few different values. Converting such a string variable to a categorical variable will save some memory	The column will be assigned as object data type when it has mixed types (numbers and strings). If a column contains ‘nan’(blank cells), pandas will default to object datatype.
A categorical variable takes on a limited, fixed number of possible values	For strings, the length is not fixed

- **Checking data types of each column:**
  - dtypes returns a series with the data type of each column
  - Syntax: DataFrame.dtypes
  - cars_data.dtypes
- **Count of unique data types**
  - get_dtype_counts()returns counts of unique data types in the dataframe
  - DataFrame.get_dtype_counts()
  - cars_data.get_dtype_counts()

- **Selecting data based on data types:**
  - pandas.DataFrame.select_dtypes() returns a subset of the columns from dataframe based on the column dtypes
  - Syntax: DataFrame.select_dtypes(include=None, exclude=None)
  - cars_data.select_dtypes(exclude=[object])

Concise summary of dataframe

- info() returns a concise summary of a dataframe
  - data type of index
  - data type of columns
  - count of non-null values
  - memory usage
  - Syntax: DataFrame.info()
  - cars_data.info()

- **Unique elements of columns:**
  - unique() is used to find the unique elements of a column
  - Syntax: numpy.unique(array)
  - np.unique(cars_data[‘KM’])
- Using unique we can find out various unique calues in column, and from this we can find out why data type mismatches are there.
- For example: ‘KM’ has been read as object instead of integer because ‘KM’ has special character to it –’??’

Importing Data Considering Missing values

- Importing data: We need to know how missing values are represented in the dataset in order to make reasonable decisions
- The missing values exist in the form of ‘nan’
- Python, by default replace blank values with ‘nan’
- Now, importing the data considering other forms of missing values in a dataframe
- cars_data=pd.read_csv(“Toyota.csv”, index_col=0, na_values=[“??”,”????”])
- Check summary with info() to see the change in Data type due to missing values

Converting variable's data types

- astype() method is used to explicitly convert data types from one to another
  - Syntax: DataFrame.astype(dtype)
  - cars_data['MetColor']=cars_data['MetColor'].astype('object')
- category vs object data type
  - nbytes() is used to get the total bytes consumed by the elements of the columns
  - Syntax: ndarray.nbytes
  - If 'FuelType' is of object data type, cars_data['FuelType'].nbytes = 11488
  - If 'FuelType' is of category data type, cars_data['FuelType'].nbytes = 1460

Cleaning column

- Checking unique values of variable ‘Doors’ :
  - print(np.unique(cars_data[‘Doors’])
  - replace() is used to replace a value with the desired value
  - Syntax: DataFrame.replace([to_replace, value, …])
  - cars_data[‘Doors’].replace(‘three’),3, inplace=True)
  - cars_data[‘Doors’].replace(‘four’),4, inplace=True)
  - cars_data[‘Doors’].replace(‘five’),5, inplace=True)
- Converting ‘Doors’ to int64:
  - cars_data[‘Doors’]=cars_data[‘Doors’].astype(‘int64’)

To detect missing values

- To check the count of missing values present in each column Dataframe.isnull().sum() is used. This will give you count of missing values in each columns.

DATA MUNGING

- Data Munging is the process by which the data is identified, extracted, cleaned as well as integrated in order to gain a good dataset which is suitable for both exploration and analysis.
- Data Munging can also be referred to as data wrangling and it includes various aspects such as data quality, merging of different sources, reproducible processes, managing data, etc.
- Whatever you want to call it — data wrangling, data munging, or data transformation, the part of the *Data Science Process* sitting in between data acquisition and exploratory data analysis (EDA) is one of the core skills a data scientist must have.
- It includes a set of tasks you have to perform in order to understand your data and prep it for machine learning.
- Why Is It Important
  - Data munging plays a crucial role in an organisation. The process can be time-consuming but the valuable insights it is producing plays an important role in the organisation. The wrangled data can be organised into a standard repeatable process which can be moved and transformed in a common format and can be reused later for multiple times.

- **The Importance of Good Data Wrangling Skills:** In the real world, you rarely get flawless data, especially when working with constantly evolving technology. This means that you have to know the business context for the data well enough to be able to interpret it, clean it, and transform it into an ingestible form.
- A good data wrangler knows how to integrate information from multiple data sources, solving common transformation problems, and resolve data cleansing and quality issues. A data wrangler also knows their data intimately, and is always looking for ways to enrich the data.
- Data wrangling skills are so integral to the job, many leading tech companies typically ask new data science candidates to perform a series of data transformations, including merging, ordering, aggregation, etc., using data science programming languages R, Python, Julia, or even SQL, along with a specific data set designed to demonstrate their capabilities in this area.
- Without solid data wrangling skills, the rest of the data science process simply can't progress in any meaningful way.

- **How to Approach Data Wrangling:**
  - **It is a six steps process**

1. **Discovering** — includes some of the EDA steps in the data science process, i.e. getting to know your data in terms of patterns and correlations. You'll often work with a domain expert here.
2. **Structuring** — since data comes in all shapes and sizes, you'll need to be able to merge, order, and reshape the data to be suitable for machine learning.
3. **Cleaning** — enterprise data is often dirty and inconsistent. Missing data values will affect the accuracy of your models. Date values can cause particular frustrations due to the many ways of representing dates in a database.
4. **Enriching** — how can you derive data from what you already have? For instance, if you have a business address in your data set, for machine learning and data visualization purposes, it would be helpful to supplement the address with longitude and latitude values.

5. **Validating** — validating the data is really the next step after cleaning, by taking a deeper look at the data values to make sure they make sense statistically and to the correct business context.
6. **Publishing** — after completing data wrangling, you'll need to integrate all the individual steps in a "data pipeline" so when the data set needs to be refreshed, you can simply re-run the pipeline and execute all the data wrangling tasks at once. You should fully document the data wrangling steps so you won't forget the decisions you made along the way.

- In lab sessions we will learn Data Cleaning and Munging using Pandas.

Data Manipulation

- The process of data transformation, formatting & structuring.
- Data manipulation can be interpreted as the process of changing data in order to make it easier to read or be more organized.
- For an instance, the information in the data becomes easier to locate by arranging its log of data in alphabetical order thereby by presenting individual entries for each of it.
- Examples updating, adding/removing, sorting, selection, merging, shifting, aggregation, etc.
- Tip: In Data Science, data come with collection & science starts with manipulation
- **Different Ways To Manipulate Data:**
  - There is no specific way for indicating whether the particular procedure for data manipulation is correct or wrong. As Data manipulation is everything about understanding the data, so whichever procedure you follow to achieve this doesn't matter much.

- Why do we need it?
  - Big Data and Internet of Things result in large amount of unstructured data.
  - New data collection opportunities require advanced data manipulation techniques.
  - Involvement with data manipulation becomes more likely & required nowadays.

- How to manipulate data
  - Most programming languages and several software tools can manipulate data:
    - Java, Python, C, C++, etc.
    - Matlab, R, Excel, etc.
  - Criteria for selection:
    - Ease of use
    - Library support
    - Portability
    - Performance
    - Data format
- We will use Python's pandas library to do this task. Check Lab material for the same.