

Group A Assignment 1

Title:

Perform the given operations using Python on any open source dataset.

Objective:

1. Import all the required Python Libraries.
2. Locate an open source data from the web (e.g. <https://www.kaggle.com>). Provide a clear description of the data and its source
3. Load the Dataset into pandas data frame.
4. Data Preprocessing: check for missing values in the data using pandas `isnull()`, `describe()` function to get some initial statistics. Provide variable descriptions. Types of variables etc. Check the dimensions of the data frame.

Theory:

What is Data Wrangling?

Data wrangling can be defined as the process of cleaning, organizing, and transforming raw data into the desired format for analysts to use for prompt decision-making.

Importance of Data Wrangling:

Data professionals spend almost 80% of their time wrangling the data, leaving a mere 20% for exploration and modelling?

- Making raw data usable. Accurately wrangled data guarantees that quality data is entered into the downstream analysis.
- Getting all data from various sources into a centralized location so it can be used.
- Piecing together raw data according to the required format and understanding the business context of data
- Automated data integration tools are used as data wrangling techniques that clean and convert source data into a standard format that can be used repeatedly according to end requirements.
- Cleansing the data from the noise or flawed, missing elements
- Data wrangling acts as a preparation stage for the data mining process, which involves gathering data and making sense of it.
- Helping business users make concrete, timely decisions

Data wrangling software performs six iterative steps

1. Discovering
2. Structuring
3. Cleaning
4. Enriching
5. Validating

6. Publishing data

Some examples of basic data wrangling tools are:

- **Spreadsheets / Excel Power Query** - It is the most basic manual data wrangling tool
- **OpenRefine** - An automated data cleaning tool that requires programming skills
- **Tabula** – It is a tool suited for all data types
- **Google DataPrep** – It is a data service that explores, cleans, and prepares data
- **Data wrangler** – It is a data cleaning and transforming tool

1 Import module in Python :

- Import in python is similar to #include header_file in C/C++. Python modules can get access to code from another module by importing the file/function using import.

```
import math  
print(math.pi)
```

syntax :
import module_name.member_name

- **syntax :**

from module_name import *

```
from math import pi
```

In the above code module, math is not imported, rather just pi has been imported as a variable.

All the functions and constants can be imported using *.

- **2 Open source data from web**
- Open Data means the kind of data which is open for anyone and everyone for access, modification, reuse, and sharing.
-
- Following is the link to get open source dataset
- <https://rockcontent.com/blog/data-sources/>

you can download data and upload on colab for execution demonstration.

What is a CSV?

“Comma Separated Values.” It is the simplest form of storing data in tabular form as plain text.

Structure of CSV:

| YearsExperience,Salary |
|------------------------|
| 1.1,39343.00 |
| 1.3,46205.00 |
| 1.5,37731.00 |
| 2.0,43525.00 |
| 2.2,39891.00 |
| 2.9,56642.00 |
| 3.0,60150.00 |
| 3.2,54445.00 |
| 3.2,54445.00 |
| 3.2,64445.00 |

Reading a CSV

copy and paste this code in google colab and show output.

```
import csv
file = open("Salary_Data.csv")
csvreader = csv.reader(file)
header = next(csvreader)
print(header)
rows = []
for row in csvreader:
    rows.append(row)
print(rows)
file.close()
```

2 Implementing the above code using with() statement:

```
import csv
rows = []
with open("Salary_Data.csv", 'r') as file:
    csvreader = csv.reader(file)
    header = next(csvreader)
    for row in csvreader:
        rows.append(row)
print(header)
print(rows)
```

Using pandas:

1. Import pandas library

2. Load CSV files to pandas using read_csv()

Basic Syntax: pandas.read_csv(filename, delimiter=',')

3. Extract the field names

.columns is used to obtain the header/field names.

4. Extract the rows

All the data of a data frame can be accessed using the field names.

```
import pandas as pd
```

```
data= pd.read_csv("Salary_Data.csv")
```

```
data
```

```
data.columns
```

```
data.Salary
```

3. Writing to a CSV file

1 Using csv.writer

Let's assume we are recording 3 Students data(Name, M1 Score, M2 Score)

```
header = ['Name', 'M1 Score', 'M2 Score']
```

```
data = [['Alex', 62, 80], ['Brad', 45, 56], ['Joey', 85, 98]]
```

```
filename = 'Students_Data.csv'
```

```
with open(filename, 'w', newline='') as file:
```

```
    csvwriter = csv.writer(file) # 2. create a csvwriter object
```

```
    csvwriter.writerow(header) # 4. write the header
```

```
    csvwriter.writerows(data) # 5. write the rest of the data
```

2 Using .writelines()

```
header = ['Name', 'M1 Score', 'M2 Score']
```

```
data = [['Alex', 62, 80], ['Brad', 45, 56], ['Joey', 85, 98]]
```

```
filename = 'Student_scores.csv'
```

```
with open(filename, 'w') as file:
```

```
    for header in header:
```

```
        file.write(str(header)+' ', )
```

```
    file.write('\n')
```

```
    for row in data:
```

```

for x in row:
    file.write(str(x)+' ')
file.write('n')

```

3. Using pandas

```

header = ['Name', 'M1 Score', 'M2 Score']
data = [['Alex', 62, 80], ['Brad', 45, 56], ['Joey', 85, 98]]
data = pd.DataFrame(data, columns=header)
data.to_csv('Stu_data.csv', index=False)

```

Data cleaning means fixing bad data in your data set.

Bad data could be:

- Empty cells
- Data in wrong format
- Wrong data
- Duplicates

Sources of Missing Values

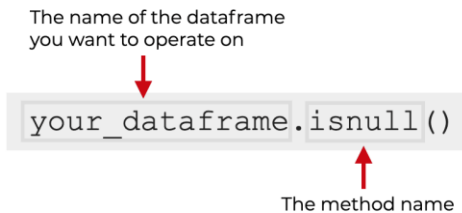
- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

| ST_NUM | ST_NAME | NUM_BEDROOMS | OWN_OCCUPIED |
|--------|------------|--------------|--------------|
| 104 | PUTNAM | 3 | Y |
| 197 | LEXINGTON | 3 | N |
| | LEXINGTON | n/a | N |
| 201 | BERKELEY | 1 | 12 |
| 203 | BERKELEY | 3 | Y |
| 207 | BERKELEY | NA | Y |
| NA | WASHINGTON | 2 | |
| 213 | TREMONT | -- | Y |
| 215 | TREMONT | na | Y |

isnull() on:

- dataframes
- dataframe columns

The name of the dataframe
you want to operate on



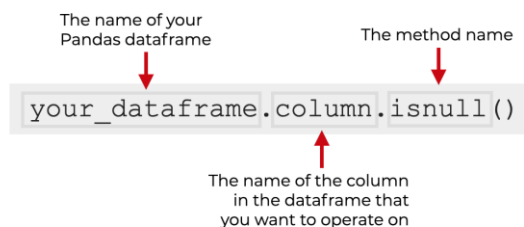
```
your_dataframe.isnull()
```

The method name

The output will be an object of the same size as your dataframe that contains boolean True/False values. These boolean values indicate which dataframe values were missing.

COLUMN SYNTAX

The name of your
Pandas dataframe



```
your_dataframe.column.isnull()
```

The method name

The name of the column
in the dataframe that
you want to operate on

COUNT THE MISSING VALUES IN EVERY COLUMN OF A DATAFRAME

```
(sales_data  
.isnull()  
.sum()  
)
```

OUT:

```
name      0  
region    1  
sales     2  
expenses  2  
dtype: int64
```

Empty cells can potentially give you a wrong result when you analyze data.

1.Remove Rows

```
import pandas as pd
```

```
df = pd.read_csv('data.csv')
```

```
new_df = df.dropna()
```

```
print(new_df.to_string())
```

2. Replace Empty Values

Replace NULL values with the number 130:

```
import pandas as pd
```

```
df = pd.read_csv('data.csv')
```

```
df.fillna(130, inplace = True)
```

3. Replace Only For Specified Columns

```
import pandas as pd
```

```
df = pd.read_csv('data.csv')
```

```
df["Calories"].fillna(130, inplace = True)
```

4. Replace Using Mean, Median, or Mode

```
import pandas as pd
```

```
df = pd.read_csv('data.csv')
```

```
x = df["Calories"].mean()
```

```
df["Calories"].fillna(x, inplace = True)
```

Pandas DataFrame describe() Method

The describe() method returns description of the data in the DataFrame.

If the DataFrame contains numerical data, the description contains these information for each column:

count - The number of not-empty values.

mean - The average (mean) value.

std - The standard deviation.

min - the minimum value.

25% - The 25% percentile*.

50% - The 50% percentile*.

75% - The 75% percentile*.

max - the maximum value.

Conclusion: by importing the required libraries we load the dataset into panda's library, and perform the data pre-processing techniques on it.

Group A Assignment 2

Title: Create an “Academic performance” dataset of students and perform the given operations using Python.

Objective:

1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them.
2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.
3. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution.

Theory:

#Reading the dataset in a dataframe using Pandas

```
df = pd.read_csv("C:/Users/admin/Documents/diabetes.csv")
```

#describe the given data

```
print(df.describe())
```

#Display first 10 rows of data

```
print(df.head(10))
```

#Missing values In Pandas missing data is represented by two values:

None: None is a Python singleton object that is often used for missing data in Python code.

NaN: NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems

- ☐ isnull()
- ☐ notnull()
- ☐ dropna()
- ☐ fillna()
- ☐ replace()
- ☐ interpolate()

identify missing items

```
print(df.isnull())
```

#outlier data items :-

An outlier is an observation of a data point that lies an abnormal distance from other values in a given population. (odd man out)

- Like in the following data point (Age)
 - 18,22,45,67,89,**125**,30

It is an abnormal observation during the Data Analysis stage, that data point lies far away from other values.

- List of Animals
 - cat, fox, rabbit, **fish**

Methods

1. Using Box Plot

It captures the summary of the data effectively and efficiently with only a simple box and whiskers.

```
# Box Plot
import seaborn as sns
sns.boxplot(df_boston['DIS'])
```

2. Using ScatterPlot

It is used when you have paired numerical data, or when your dependent variable has multiple values for each reading independent variable, or when trying to determine the relationship between the two variables.

```
# Scatter plot
fig, ax = plt.subplots(figsize = (18,10))
ax.scatter(df_boston['INDUS'], df_boston['TAX'])

# x-axis label
ax.set_xlabel('(Proportion non-retail business acres)/(town)')

# y-axis label
ax.set_ylabel('(Full-value property-tax rate)/( $10,000)')
plt.show()
```

3. Z-score

It is also called a standard score. This value/score helps to understand that how far is the data point from the mean. And after setting up a threshold value one can utilize z score values of data points to define the outliers.

Zscore = (data_point - mean) / std. deviation

```
# Z score
from scipy import stats
import numpy as np

z = np.abs(stats.zscore(df_boston['DIS']))
print(z)
```

Now to define an outlier threshold value is chosen which is generally 3.0. As 99.7% of the data points lie between +/- 3 standard deviation (using Gaussian Distribution approach).

```
threshold = 3
# Position of the outlier
print(np.where(z > 3))
```

4. IQR (Inter Quartile Range)

Inter Quartile Range approach to finding the outliers is the most commonly used and most trusted approach used in the research field.

$IQR = Quartile3 - Quartile1$

```
# IQR
Q1 = np.percentile(df_boston['DIS'], 25,
                    interpolation = 'midpoint')
```

```
Q3 = np.percentile(df_boston['DIS'], 75,
                    interpolation = 'midpoint')
```

```
IQR = Q3 - Q1
```

To define the outlier base value is defined above and below datasets normal range namely Upper and Lower bounds, define the upper and the lower bound (1.5*IQR value is considered) :

$upper = Q3 + 1.5 * IQR$

$lower = Q1 - 1.5 * IQR$

```
# Above Upper bound
upper = df_boston['DIS'] >= (Q3+1.5*IQR)
```

```
print("Upper bound:",upper)
print(np.where(upper))
```

```
# Below Lower bound
lower = df_boston['DIS'] <= (Q1-1.5*IQR)
print("Lower bound:", lower)
print(np.where(lower))
```

Removing the outliers

For removing the outlier, one must follow the same process of removing an entry from the dataset using its exact position in the dataset because in all the above methods of detecting the outliers end result is the list of all those data items that satisfy the outlier definition according to the method used.

`dataframe.drop(row_index, inplace = True`

The above code can be used to drop a row from the dataset given the row_indexes to be dropped. Inplace = True is used to tell python to make the required change in the original dataset. row_index can be only one value or list of values or NumPy array but it must be one dimensional.

To change the scale for better understanding of the variable

Consider,

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

df = pd.DataFrame({
    'Income': [15000, 1800, 120000, 10000],
    'Age': [25, 18, 42, 51],
    'Department': ['HR','Legal','Marketing','Management']
})
```

Before directly applying any feature transformation or scaling technique, we need to remember the categorical column: Department and first deal with it. This is because we cannot scale non-numeric values.

For that, we 1st create a copy of our dataframe and store the numerical feature names in a list, and their values as well:

```
df_scaled = df.copy()
col_names = ['Income', 'Age']
features = df_scaled[col_names]
```

MinMax Scaler

It just scales all the data between 0 and 1. The formula for calculating the scaled value is-

$$x_scaled = (x - x_min)/(x_max - x_min)$$

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df_scaled[col_names] = scaler.fit_transform(features.values)
```

You can see how the values were scaled. The minimum value among the columns became 0, and the maximum value was changed to 1, with other values in between. However, suppose we don't want the income or age to have values like 0. Let us take the range to be (5, 10)

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(5, 10))
```

```
df_scaled[col_names] = scaler.fit_transform(features.values)
df_scaled
```

Standard Scaler

For each feature, the Standard Scaler scales the values such that the mean is 0 and the standard deviation is 1(or the variance).

$$x_scaled = x - \text{mean}/\text{std_dev}$$

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```
df_scaled[col_names] = scaler.fit_transform(features.values)
df_scaled
```

MaxAbsScaler

In simplest terms, the MaxAbs scaler takes the absolute maximum value of each column and divides each value in the column by the maximum value.

Thus, it first takes the absolute value of each value in the column and then takes the maximum value out of those. This operation scales the data between the range [-1, 1].

```
df["Balance"] = [100.0, -263.0, 2000.0, -5.0]
```

```
from sklearn.preprocessing import MaxAbsScaler
scaler = MaxAbsScaler()
```

```
df_scaled[col_names] = scaler.fit_transform(features.values)
df_scaled
```

Robust Scaler

The Robust Scaler, as the name suggests is not sensitive to outliers. This scaler-

1. removes the median from the data
2. scales the data by the InterQuartile Range(IQR)

$$x_scaled = (x - Q1)/(Q3 - Q1)$$

```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
```

```
df_scaled[col_names] = scaler.fit_transform(features.values)
df_scaled
```

To decrease the skewness and convert the distribution into a normal distribution

Log Transform

It is primarily used to convert a [skewed distribution](#) to a normal distribution/less-skewed distribution. In this transform, we take the log of the values in a column and use these values as the column instead.

```
df['log_income'] = np.log(df['Income'])
# We created a new column to store the log values
```

This is how the dataframe looks like:

| | Income | Age | Department | log_income |
|---|--------|-----|------------|------------|
| 0 | 15000 | 25 | HR | 9.615805 |
| 1 | 1800 | 18 | Legal | 7.495542 |
| 2 | 120000 | 42 | Marketing | 11.695247 |
| 3 | 10000 | 51 | Management | 9.210340 |

```
df['log_income'].plot.hist(bins = 5)
```

Conclusion: Thus we scan all variables for missing values & inconsistencies, outliers and applied suitable technique to deal with them. We also applied data transformation on variables.

Group A Assignment 3

Title:

Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) variable.

Objective:

1. If your categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups.
2. Create a list that contains a numeric value for each response to the categorical variable.
3. Display some basic statistical details like percentile, mean, standard deviation etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris-versicolor' of iris.csv dataset.

Theory:

What is Statistics?

- ✓ Statistics is a branch of mathematics that deals with collecting, analyzing, interpreting, and visualizing empirical data.
- ✓ Descriptive statistics and inferential statistics are the two major areas of statistics.
- ✓ Descriptive statistics are for describing the properties of sample and population data (what has happened).
- ✓ Inferential statistics use those properties to test hypotheses, reach conclusions, and make predictions (what can you expect).

Use of Statistics in Data Science

- ✓ Asking questions about the data
- ✓ Cleaning and preprocessing the data
- ✓ Selecting the right features
- ✓ Model evaluation
- ✓ Model prediction



Mean

- ✓ The arithmetic mean of a given data is the sum of all observations divided by the number of observations.

- ✓ For example, a cricketer's scores in five ODI matches are as follows: 12, 34, 45, 50, 24. To find his average score in a match, we calculate the arithmetic mean of data using the mean formula:

$$\text{Mean} = \frac{\text{Sum of terms}}{\text{Number of terms}}$$

- ✓ Mean = Sum of all observations/Number of observations

$$\text{Mean} = (12 + 34 + 45 + 50 + 24)/5$$

$$\text{Mean} = 165/5 = 33$$

Mean is denoted by \bar{x} (pronounced as x bar).

To find the mean or the average salary of the employees, you can use the mean() functions in Python.

```
print(df['Salary'].mean())
71000.0
```

Mode

- ✓ The Mode refers to the most frequently occurring value in your data.
- ✓ You find the frequency of occurrence of each number and the number with the highest frequency is your mode. If there are no recurring numbers, then there is no mode in the data.
- ✓ Using the mode, you can find the most commonly occurring point in your data. This is helpful when you have to find the central tendency of categorical values, like the flavor of the most popular chip sold by a brand. You cannot find the average based on the orders; instead, you choose the chip flavor with the highest orders.
- ✓ Usually, you can count the most frequently occurring values and get your mean. But this only works when the values are discrete. Now, again take the example of class marks.
- ✓ Example: Take the following marks of students :

Marks = 35, 40, 45, 49, 34, 47, 39, 25, 19, 35, 28, 48

Over here, the value 35 occurs the most frequently and hence is the mode.

- ✓ But what if the values are categorical? In that case, you must use the formula below:

$$\text{Mode} = l + \left(\frac{f_1 - f_0}{2f_1 - f_0 - f_2} \right) \times h$$

Where,

l = lower limit of modal class

h = lower limit of preceding modal class

f_1 = frequency of modal class

f_0 = frequency of class preceding modal class

f_2 = frequency of class succeeding modal class

The modal class is simply the class with the highest frequency. Consider the range of frequencies given for the marks obtained by students in a class:

| Marks | 10-20 | 20-30 | 30-40 | 40-50 |
|--------------------|-------|-------|-------|-------|
| Number of Students | 1 | 3 | 5 | 4 |

In this case, you can see that class 30-40 has the highest frequency, hence it is the modal class. The remaining values are as follows: $l = 30$, $h = 20$, $f_1 = 5$, $f_0 = 3$, $f_2 = 4$

In that case, the mode becomes :

$$\begin{aligned}\text{Mode} &= 30 + \left(\frac{5-3}{2*5-3-4} \right) \times 20 \\ &= 43.33\end{aligned}$$

Hence, the mark which occurs most frequently is 43.33. The mode of salary from the salary data frame can be calculated as:

```
print(df['Salary'].mode())
```

```
0    50000  
dtype: int64
```

Median

- ✓ Median refers to the middle value of a data. To find the median, you first sort the data in either ascending or descending order or then find the numerical value present in the middle of your data.
- ✓ It can be used to figure out the point around which the data is centered. It divides the data into two halves and has the same number of data points above and below.
- ✓ The median is especially useful when the data is skewed data. That is, it has high data distribution towards one side. In this case, the average wouldn't give you a fair mid-value but would lean more towards the higher values. In this case, you can use the middle data point as the central point instead.

- ✓ Consider n terms $X_1, X_2, X_3, \dots, X_n$. The basic formula for the median is by dividing the total number of observations by 2. This works fine when you have an odd number of terms because you will have one middle term and the same number of terms above and below. For an even number of terms, consider the two middle terms and find their average.

$$\text{Median} = \frac{n+1}{2} \text{ th term , } n = \text{odd}$$
$$\left\{ \frac{n}{2} \text{ th term} + \frac{n}{2} + 1 \text{ th term} \right\} / 2 , n = \text{even}$$

Example: Consider following are students marks

Marks = 35, 40, 45, 49, 34, 47, 39, 25, 19, 35, 28, 48

To find the middle term, you first have to sort the data or arrange the data in ascending or descending order. This ensures that consecutive terms are next to each other.

Sorted Marks = 19, 25, 28, 30, 34, 35, 39, 40, 45, 47, 48, 49

You can see that we have 12 data points, so use the median formula for even numbers.

$$\begin{aligned} \text{Median} &= \left\{ \left(\frac{12}{2} \text{ th term} \right) + \left(\frac{12}{2} + 1 \text{ th term} \right) \right\} / 2 \\ &= \{ 6^{\text{th}} + 7^{\text{th}} \} / 2 = (35 + 39) / 2 \\ &= 37 \end{aligned}$$

So, the middle term in the range of marks is 37. This means that the other marks lie in a frequency range of around 37.

The `median()` function in Python can help you find the median value of a column. From the salary data frame, you can find the median salary as:

```
print(df['Salary'].median())
```

```
54000.0
```

Standard Deviation and Variance

- ✓ Deviation just means how far from the normal
- ✓ The Standard Deviation is a measure of how spread out numbers are.
- ✓ Its symbol is σ (the greek letter sigma)

- ✓ The formula is easy: it is the square root of the Variance.

$$\sigma^2 = \frac{\sum_{i=1}^n (x_i - \mu)^2}{n}$$

- ✓ The Variance is defined as: The average of the squared differences from the Mean.
- ✓ To calculate the variance follow these steps:
1. Work out the Mean (the simple average of the numbers)
 2. Then for each number: subtract the Mean and square the result (the squared difference).
 3. Then work out the average of those squared differences
- ✓ Variance is used to measure the variability in the data from the mean.

To calculate the Variance, take each difference, square it, and then average the result:

Variance =

$$\begin{aligned}\sigma^2 &= \frac{206^2 + 76^2 + (-224)^2 + 36^2 + (-94)^2}{5} \\ &= \frac{42436 + 5776 + 50176 + 1296 + 8836}{5} \\ &= \frac{108520}{5} \\ &= 21704\end{aligned}$$

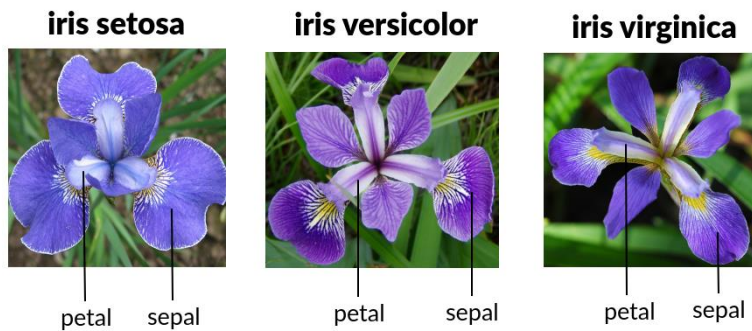
So the Variance is 21,704

And the Standard Deviation is just the square root of Variance, so:

Standard Deviation =

$$\begin{aligned}\sigma &= \sqrt{21704} \\ &= 147.32... \\ &= \mathbf{147} \text{ (to the nearest mm)}\end{aligned}$$

Finding statistical information of the iris flower dataset



Iris flower has three species - Setosa, Versicolor and Virginica

Load the dataset

```
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv'
col_name = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names = col_name)
```

Display some information of the data

```
dataset.shape
```

```
dataset.head()
```

```
dataset.info()
```

Display statistical information of the data

```
dataset.describe()
```

Output:

| | | | | |
|-------|------------|------------|------------|------------|
| count | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| mean | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| std | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| min | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| 25% | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| 50% | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| 75% | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| max | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

groupby()

A groupby operation involves some combination of splitting the object, applying a function, and combining the results. This can be used to group large amounts of data and compute operations on these groups.

Example:

```
df = pd.DataFrame({'Animal': ['Falcon', 'Falcon',  
...                          'Parrot', 'Parrot'],  
...               'Max Speed': [380., 370., 24., 26.]})  
>>> df  
   Animal  Max Speed  
0  Falcon    380.0  
1  Falcon    370.0  
2  Parrot     24.0  
3  Parrot     26.0  
>>> df.groupby(['Animal']).mean()  
           Max Speed  
Animal  
Falcon    375.0  
Parrot    25.0
```

Conclusion: Thus, we performed summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset with numeric variables grouped by the qualitative (categorical) variable.

Group A Assignment 4

Title:

Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (<https://www.kaggle.com/c/boston-housing>).

Objective:

1. To predict the value of prices of the house using the given features.

Theory:

What is linear Regression?

Linear Regression is the supervised Machine Learning model in which the model finds the best fit linear line between the independent and dependent variable i.e it finds the linear relationship between the dependent and independent variable.

Linear Regression on Boston Housing Dataset

The Housing dataset which contains information about different houses in Boston. This data was originally a part of UCI Machine Learning Repository and has been removed now. We can also access this data from the scikit-learn library. There are 506 samples and 14 feature variables in this dataset. The objective is to predict the value of prices of the house using the given features.

The problem that we are going to solve here is that given a set of features that describe a house in Boston, our machine learning model must predict the house price. To train our machine learning model with boston housing data, we will be using scikit-learn's boston dataset.

In this dataset, each row describes a boston town or suburb. There are 506 rows and 14 attributes (features) with a target column (MEDV).

Now, split the data into independent variables (X's) and dependent variable (Y) data sets. The data will be stored in df_x for the independent variables and df_y for the dependent variable.

#Transform the data set into a data frame

#NOTE: boston.data = the data we want,

boston.feature_names = the column names of the data

boston.target = Our target variable or the price of the houses

`df_x = pd.DataFrame(boston.data, columns = boston.feature_names)`

`df_y = pd.DataFrame(boston.target)`

Initialize the Linear Regression model, split the data into 67% training and 33% testing data, and then train the model with the training data set that contains the independent variables.

```
#Initialize the linear regression model
reg = linear_model.LinearRegression()#Split the data into 67% training and 33% testing data
#NOTE: We have to split the dependent variables (x) and the target or independent variable (y)
x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.33,
random_state=42)#Train our model with the training data
reg.fit(x_train, y_train)
```

Get the estimated coefficients for the linear regression model

```
print(reg.coef_)
```

Now that we are done training the linear regression model and looking at the coefficients that describe the linear function, let's print the model's predictions (what it thinks the values will be for houses) on the test data.

```
#print our price predictions on our test data
y_pred = reg.predict(x_test)
print(y_pred)
```

We want to know what was the actual values for that test data set, so I will print those values to the screen, but first I will print at least one row from the model's prediction, just to make it a little easier to compare data.

```
#Print the the prediction for the third row of our test data actual price = 13.6
y_pred[2]#print the actual price of houses from the testing data set
y_test[0]
```

To check the model's performance/accuracy I will use a metric called mean squared error (MSE). This measurement is simple to implement and easy to understand. The MSE is a measure of the quality of an estimator — it is always non-negative, and values closer to zero indicate a better fit. Usually you want to evaluate your model with other metrics as well to truly get an idea of how well your model performs. I'll do this two different ways, one using numpy and the other using sklearn.metrics.

```
# Two different ways to check model performance/accuracy using,
# mean squared error which tells you how close a regression line is to a set of points.

# 1. Mean squared error by numpy
print(np.mean((y_pred-y_test)**2))

# 2. Mean squared error by sklearn
# Resource: https://stackoverflow.com/questions/42453875/precision-score-and-accuracy-score-showing-value-error?rq=1
from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test, y_pred))
```

Conclusion: Thus, we predicted the value of prices of the house using the given features.

Group A Assignment 5

Title:

Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset.

Objective:

Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

Theory:

What is logistic Regression?

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

Linear Regression on Boston Housing Dataset

We will be taking data from social network ads which tell us whether a person will purchase the ad or not based on the features such as age and salary.

First, we will import all the libraries:

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd
```

Now we will import the dataset and select only age and salary as the features

```
dataset = pd.read_csv('Social_Network_Ads.csv')

dataset.head()
```

Now we will perform splitting for training and testing. We will take 75% of the data for training, and test on the remaining data

```
from sklearn.model_selection import train_test_split
```



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

Next, we scale the features to avoid variation and let the features follow a normal distribution

```
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
```

```
X_train = sc.fit_transform(X_train)
```

```
X_test = sc.transform(X_test)
```

The preprocessing part is over. It is time to fit the model

```
from sklearn.linear_model import LogisticRegression
```

```
classifier = LogisticRegression(random_state = 0)
```

```
classifier.fit(X_train, y_train)
```

We fitted the model on training data. We will predict the labels of test data.

```
y_pred = classifier.predict(X_test)
```

What is a confusion matrix?

It is a matrix of size 2×2 for binary classification with actual values on one axis and predicted on another.

| | | ACTUAL | |
|------------|----------|-------------------|-------------------|
| | | Negative | Positive |
| PREDICTION | Negative | TRUE NEGATIVE | FALSE NEGATIVE |
| | Positive | FALSE POSITIVE | TRUE POSITIVE |

Precision

Out of all the positive predicted, what percentage is truly positive.

$$Precision = \frac{TP}{TP + FP}$$

The precision value lies between 0 and 1.

Recall

Out of the total positive, what percentage are predicted positive. It is the same as TPR (true positive rate).

$$Recall = \frac{TP}{TP + FN}$$

The prediction is over. Now we will evaluate the performance of our model.

```
From sklearn.metrics import confusion_matrix, classification_report
```

```
cm = confusion_matrix(y_test, y_pred)
```

```
cl_report=classification_report(y_test,y_pred)
```

| | | precision | recall | f1-score |
|---------|--------------|-----------|--------|----------|
| support | | | | |
| | 0 | 0.89 | 0.96 | 0.92 |
| 68 | 1 | 0.89 | 0.75 | 0.81 |
| 32 | | | | |
| | accuracy | | | 0.89 |
| 100 | macro avg | 0.89 | 0.85 | 0.87 |
| 100 | weighted avg | 0.89 | 0.89 | 0.89 |
| 100 | | | | |

Conclusion: Thus, we implemented the logistic regression on social network adv dataset and identified the accuracy, precision and recall.

DS&BDL

Assignment 6

Title: Data Analytics III

Aim

1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

Objective:

Students are able to learn:

1. How to calculate the probabilities required by the Naive Bayes algorithm.
2. How to implement the Naive Bayes algorithm from scratch.
3. How to apply Naive Bayes to a real-world predictive modeling problem.

Software/Hardware Requirements: Python/R, on iris.csv dataset, Linux Operating System.

Theory:

Naive Bayes

Bayes' Theorem provides a way that we can calculate the probability of a piece of data belonging to a given class, given our prior knowledge. Bayes' Theorem is stated as:

Classification is a predictive modeling problem that involves assigning a label to a given input data sample.

The problem of classification predictive modeling can be framed as calculating the conditional probability of a class label given a data sample, for example:

$$P(\text{class}|\text{data}) = (P(\text{data}|\text{class}) * P(\text{class})) / P(\text{data})$$

Where $P(\text{class}|\text{data})$ is the probability of class given the provided data.

This calculation can be performed for each class in the problem and the class that is assigned the largest probability can be selected and assigned to the input data.

Naive Bayes is a classification algorithm for binary (two-class) and multiclass classification problems. It is called Naive Bayes or idiot Bayes because the calculations of the probabilities for each class are simplified to make their calculations tractable.

Rather than attempting to calculate the probabilities of each attribute value, they are assumed to be conditionally independent given the class value.

For this implementation we will use the Iris Flower Species Dataset.

The Iris Flower Dataset involves predicting the flower species given measurements of iris flowers.

It is a multiclass classification problem. The number of observations for each class is balanced. There are 150 observations with 4 input variables and 1 output variable. The variable names are as follows:

- Sepal length in cm.
- Sepal width in cm.
- Petal length in cm.
- Petal width in cm.
- Class

Steps:

1. Load Database
2. Separate By Class.
3. Summarize Dataset.
4. Summarize Data By Class.
5. Gaussian Probability Density Function.
6. Class Probabilities.

The first step is to load the dataset and convert the loaded data to numbers that we can use with the mean and standard deviation calculations. For this we will use the helper function `load_csv()` to load the file, `str_column_to_float()` to convert string numbers to floats and `str_column_to_int()` to convert the class column to integer values.

We will evaluate the algorithm using k-fold cross-validation with 5 folds. This means that $150/5=30$ records will be in each fold. We will use the helper functions `evaluate_algorithm()` to evaluate the algorithm with cross-validation and `accuracy_metric()` to calculate the accuracy of predictions.

A new function named `predict()` was developed to manage the calculation of the probabilities of a new row belonging to each class and selecting the class with the largest probability value.

Another new function named `naive_bayes()` was developed to manage the application of the Naive Bayes algorithm, first learning the statistics from a training dataset and using them to make predictions for a test dataset.

Confusion Matrix Definition

A confusion matrix is used to judge the performance of a classifier on the test dataset for which we already know the actual values. Confusion matrix is also termed as Error matrix. It consists of a count of correct and incorrect values broken down by each class. It not only tells us the error made by classifier but also tells us what type of error the classifier made. So, we can say that a confusion matrix is a performance measurement technique of a classifier model where output can be two classes or more. It is a table with four different groups of true and predicted values.

Terminologies in Confusion Matrix

The confusion matrix shows us how our classifier gets confused while predicting. In a confusion matrix we have four important terms which are:

1. **True Positive (TP)**- Both actual and predicted values are Positive.
2. **True Negative (TN)**- Both actual and predicted values are Negative.
3. **False Positive (FP)**- The actual value is negative but we predicted it as positive.
4. **False Negative (FN)**- The actual value is positive but we predicted it as negative.

Performance Metrics

Confusion matrix not only used for finding the errors in prediction but is also useful to find some important performance metrics like Accuracy, Recall, Precision, F-measure. We will discuss these terms one by one.

Accuracy

As the name suggests, the value of this metric suggests the accuracy of our classifier in predicting results.

It is defined as:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

A 99% accuracy can be good, average, poor or dreadful depending upon the problem.

Precision

Precision is the measure of all actual positives out of all predicted positive values.

It is defined as:

$$\text{Precision} = TP / (TP + FP)$$

Recall

Recall is the measure of positive values that are predicted correctly out of all actual positive values.

It is defined as:

$$\text{Recall} = TP / (TP + FN)$$

High Value of Recall specifies that the class is correctly known (because of a small number of False Negative).

F-measure

It is hard to compare classification models which have low precision and high recall or vice versa. So, for comparing the two classifier models we use F-measure. F-score helps to find the metrics of Recall and Precision in the same interval. Harmonic Mean is used instead of Arithmetic Mean.

F-measure is defined as:

$$\text{F-measure} = 2 * \text{Recall} * \text{Precision} / (\text{Recall} + \text{Precision})$$

The F-Measure is always closer to the Precision or Recall, whichever has a smaller value

Conclusion: In this assignment, we covered how Naïve Bayes theorem used to solve classification problem for iris flower dataset and what is confusion matrix, its need, and how to derive it in Python and R.

DS&BDL

Assignment 7

Title: Text Analytics

Aim

1. Extract Sample document and apply following document pre-processing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.
2. Create representation of document by calculating Term Frequency and Inverse Document Frequency..

Objective:

Students are able to learn:

1. Text Analytics and NLP.
2. Document Pre-Processing Methods.
3. Text Classification using TF-IDF

Software/Hardware Requirements: 64-bit Open source Linux, Python, NLTK

Theory:

Text Analytics has lots of applications in today's online world. By analyzing tweets on Twitter, we can find trending news and peoples reaction on a particular event. Amazon can understand user feedback or review on the specific product. BookMyShow can discover people's opinion about the movie. Youtube can also analyze and understand peoples viewpoints on a video.

In this assignment we are going to take sample document and will apply following pre-processing methods.

1. Text analysis Operations using nltk.

NLTK is a powerful Python package that provides a set of diverse natural languages algorithms. It is free, opensource, easy to use, large community, and well documented. NLTK consists of the most common algorithms such as tokenizing, part-of-speech tagging, stemming, sentiment analysis, topic segmentation, and named entity recognition. NLTK helps the computer to analysis, preprocess, and understand the written text.

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /home/northout/anaconda2/lib/python2.7/site-packages
```

```
Requirement already satisfied: six in /home/northout/anaconda2/lib/python2.7/site-packages  
(from nltk)
```

```
[33mYou are using pip version 9.0.1, however version 10.0.1 is available.
```

```
You should consider upgrading via the 'pip install --upgrade pip' command. [0m
```

```
#Loading NLTK
```

```
import nltk
```


2. Tokenization

Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentence is called Tokenization. Token is a single entity that is building blocks for sentence or paragraph.

Sentence Tokenization

Sentence tokenizer breaks text paragraph into sentences.

```
from nltk.tokenize import sent_tokenize
text="'''' ''''
tokenized_text=sent_tokenize(text)
print(tokenized_text)
```

Word Tokenization

Word tokenizer breaks text paragraph into words.

```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

3. Stop words

Stopwords considered as noise in the text. Text may contain stop words such as is, am, are, this, a, an, the, etc.

In NLTK for removing stopwords, you need to create a list of stopwords and filter out your list of tokens from these words.

```
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

Removing Stop Words

```
filtered_sent=[]
for w in tokenized_sent:
    if w not in stop_words:
        filtered_sent.append(w)
print("Tokenized Sentence:",tokenized_sent)
print("Filterd Sentence:",filtered_sent)
```

4. Stemming and Lemmatization.

Stemming

Stemming is a process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes. For example, connection, connected, connecting word reduce to a common word "connect".

```
# Stemming

from nltk.stem import PorterStemmer

from nltk.tokenize import sent_tokenize, word_tokenize
```

```

ps = PorterStemmer()

stemmed_words=[]

for w in filtered_sent:

    stemmed_words.append(ps.stem(w))

print("Filtered Sentence:",filtered_sent)

print("Stemmed Sentence:",stemmed_words)

```

Lemmatization

Lemmatization reduces words to their base word, which is linguistically correct lemmas. It transforms root word with the use of vocabulary and morphological analysis. Lemmatization is usually more sophisticated than stemming. Stemmer works on an individual word without knowledge of the context. For example, The word "better" has "good" as its lemma. This thing will miss by stemming because it requires a dictionary look-up.

```

#Lexicon Normalization

#performing stemming and Lemmatization

from nltk.stem.wordnet import WordNetLemmatizer

lem = WordNetLemmatizer()

from nltk.stem.porter import PorterStemmer

stem = PorterStemmer()

word = "flying"

print("Lemmatized Word:",lem.lemmatize(word,"v"))

print("Stemmed Word:",stem.stem(word))

```

5. POS Tagging

The primary target of Part-of-Speech(POS) tagging is to identify the grammatical group of a given word. Whether it is a NOUN, PRONOUN, ADJECTIVE, VERB, ADVERBS, etc. based on the context. POS Tagging looks for relationships within the sentence and assigns a corresponding tag to the word.

```

sent = " "
tokens=nltk.word_tokenize(sent)
print(tokens)
['Albert', 'Einstein', 'was', 'born', 'in', 'Ulm', ',', 'Germany', 'in', '1879', '.']
nltk.pos_tag(tokens)

```

6. Text Classification

Text Classification Model using TF-IDF. First, import the MultinomialNB module and create a Multinomial Naive Bayes classifier object using MultinomialNB() function. Then, fit your model on a train set using fit() and perform prediction on the test set using predict().

```

from sklearn.naive_bayes import MultinomialNB

#Import scikit-learn metrics module for accuracy calculation

from sklearn import metrics

# Model Generation Using Multinomial Naive Bayes

clf = MultinomialNB().fit(X_train, y_train)

predicted= clf.predict(X_test)

print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test, predicted))

```

Feature Generation using TF-IDF

In Term Frequency(TF), you just count the number of words occurred in each document. The main issue with this Term Frequency is that it will give more weight to longer documents. Term frequency is basically the output of the BoW model.

IDF(Inverse Document Frequency) measures the amount of information a given word provides across the document. IDF is the logarithmically scaled inverse ratio of the number of documents that contain the word and the total number of documents.

$$\text{idf}(W) = \log \frac{\#(\text{documents})}{\#(\text{documents containing word } W)}$$

TF-IDF(Term Frequency-Inverse Document Frequency) normalizes the document term matrix. It is the product of TF and IDF. Word with high tf-idf in a document, it is most of the times occurred in given documents and must be absent in the other documents. So the words must be a signature word.

```

from sklearn.feature_extraction.text import TfidfVectorizer
tf=TfidfVectorizer()
text_tf= tf.fit_transform(data['Phrase'])

```

Conclusion: In this assignment, we have learned what Text Analytics is, NLP and text mining, basics of text analytics operations using NLTK such as Tokenization, Normalization, Stemming, Lemmatization and POS tagging. What is Text Classification Model using TF-IDF.

Group A

Assignment No: 8

Title of the Assignment: Data Visualization I

1. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to see if we can find any patterns in the data.
2. Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.

Objective of the Assignment: Students should be able to perform the data Visualization operation using Python on any open source dataset

Prerequisite:

1. Basic of Python Programming
 2. Seaborn Library, Concept of Data Visualization.
-

Theory:

Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

Seaborn offers the following functionalities:

1. Dataset oriented API to determine the relationship between variables.
2. Automatic estimation and plotting of linear regression plots.
3. It supports high-level abstractions for multi-plot grids.
4. Visualizing univariate and bivariate distribution.

To initialize the Seaborn library, the command used is:

```
import seaborn as sns
```

Using Seaborn we can plot wide varieties of plots like:

1. **Distribution Plots:** The most common approach to visualizing a distribution is the *histogram*. This is the default approach in `displot()`, which uses the same underlying code as `histplot()`.

2. Pie Chart & Bar Chart:

Pie Chart is generally used to analyze the data on how a numeric variable changes across different categories. A familiar style of plot that accomplishes this goal is a bar plot. In seaborn, the `barplot()` function operates on a full dataset and applies a function to obtain the estimate

3. Scatter Plots

Scatter Plot is used when we want to plot the relationship between any two numeric columns from a dataset. These plots are the most powerful visualization tools that are being used in the field of machine learning

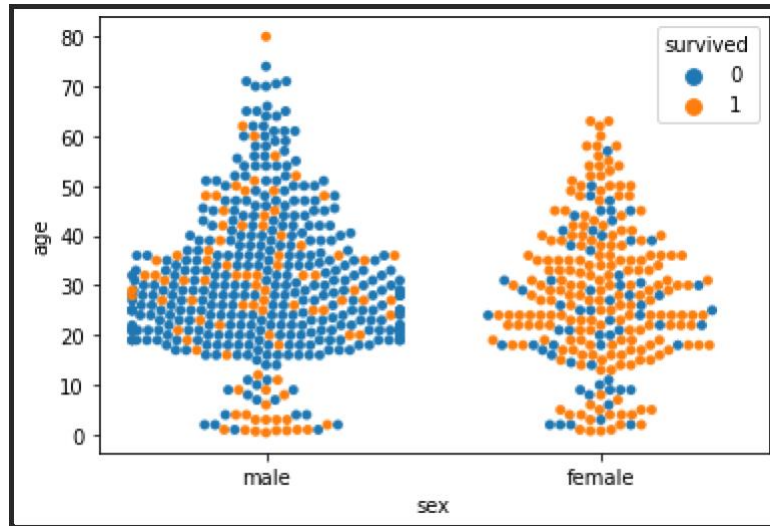
4. **Pair Plots:** Pair Plots are used when we want to see the relationship pattern among more than 3 different numeric variables

5. Heat maps:

The heatmap represents the data in a 2-dimensional form. The ultimate goal of the heatmap is to show the summary of information in a colored graph. It utilizes the concept of using colors and color intensities to visualize a range of values.

Assignment Questions

1. List out different types of plot to find patterns of data
2. Explain when you will use distribution plots and when you will use categorical plots.
3. Write the conclusion from the following swarm plot (consider titanic dataset)



4. Which parameter is used to add another categorical variable to the violin plot, Explain with syntax and example.

DS&BDL

Assignment 11

Title: Hadoop MapReduce Framework- WordCount application

Aim:

Write a code in JAVA for a simple WordCount application that counts the number of occurrences of each word in a given input set using the Hadoop MapReduce framework on local-standalone set-up.

Objective:

By completing this task, students will learn the following

1. Hadoop Distributed File System.
2. MapReduce Framework.

Software/Hardware Requirements: 64-bit Open source OS-Linux, Java, Hadoop.

Theory:

Map and Reduce tasks in Hadoop-With in a MapReduce job there are two separate tasks map task and reduce task.

Map task- A MapReduce job splits the input dataset into independent chunks known as input splits in Hadoop which are processed by the map tasks in a completely parallel manner. Hadoop framework creates separate map task for each input split.

Reduce task- The output of the maps is sorted by the Hadoop framework which then becomes input to the reduce tasks.

Hadoop MapReduce framework operates exclusively on <key, value> pairs. In a MapReduce job, the input to the Map function is a set of <key, value> pairs and output is also a set of <key, value> pairs. The output <key, value> pair may have different type from the input <key, value> pair.

$\langle K1, V1 \rangle \rightarrow \text{map} \rightarrow (K2, V2)$

The output from the map tasks is sorted by the Hadoop framework. MapReduce guarantees that the input to every reducer is sorted by key. Input and output of the reduce task can be represented as follows.

$\langle K2, \text{list}(V2) \rangle \rightarrow \text{reduce} \rightarrow \langle K3, V3 \rangle$

1. Creating and copying input file to HDFS

If you already have a file in HDFS which you want to use as input then you can skip this step.

First thing is to create a file which will be used as input and copy it to HDFS.

Let's say you have a file wordcount.txt with the following content.

Hello wordcount MapReduce Hadoop program.

This is my first MapReduce program.

You want to copy this file to /user/process directory with in HDFS. If that path doesn't exist then you need to create those directories first.

HDFS Command Reference List Link- <https://www.netjstech.com/2018/02/hdfs-commands-reference-list.html>

```
hdfs dfs -mkdir -p /user/process
```

Then copy the file wordcount.txt to this directory.

```
hdfs dfs -put /netjs/MapReduce/wordcount.txt  
/user/process
```

2. Write your wordcount mapreduce code.

WordCount example reads text files and counts the frequency of the words. Each mapper takes a line of the input file as input and breaks it into words. It then emits a key/value pair of the word (In the form of (word, 1)) and each reducer sums the counts for each word and emits a single key/value with the word and sum.

In the word count MapReduce code there is a Mapper class (MyMapper) with map function and a Reducer class (MyReducer) with a reduce function.

1. Map function

From the wordcount.txt file Each line will be passed to the map function in the following format.

```
<0, Hello wordcount MapReduce Hadoop program.>
```

```
<41, This is my first MapReduce program.>
```

In the map function the line is split on space and each word is written to the context along with the value as 1.

So the output from the map function for the two lines will be as follows.

Line 1 <key, value> output

```
(Hello, 1)
```

```
(wordcount, 1)
```

```
(MapReduce, 1)
```

```
(Hadoop, 1)
```

```
(program., 1)
```

Line 2 <key, value> output

```
(This, 1)
```

```
(is, 1)
```

```
(my, 1)
```

```
(first, 1)
```

```
(MapReduce, 1)
```

```
(program., 1)
```

2. Shuffling and sorting by Hadoop Framework

The output of map function doesn't become input of the reduce function directly. It goes through shuffling and sorting by Hadoop framework. In this processing the data is sorted and grouped by keys.

After the internal processing the data will be in the following format. This is the input to reduce function.

```
<Hadoop, (1)>
```



```
<Hello, (1)>
<MapReduce, (1, 1)>
<This, (1)>
<first, (1)>
<is, (1)>
<my, (1)>
<program., (1, 1)>
<wordcount, (1)>
```

3. How reduce task works in Hadoop

As we just saw the input to the reduce task is in the format (key, list<values>). In the reduce function, for each input <key, value> pair, just iterate the list of values for each key and add the values that will give the count for each key.

Write that key and sum of values to context, that <key, value> pair is the output of the reduce function.

```
Hadoop 1
Hello 1
MapReduce 2
This 1
first 1
is 1
my 1
program. 2
wordcount 1
```

3. Creating jar of your wordcount MapReduce code

You will also need to add at least the following Hadoop jars so that your code can compile. You will find these jars inside the /share/hadoop directory of your Hadoop installation. With in /share/hadoop path look in hdfs, mapreduce and common directories for required jars.

```
hadoop-common-2.9.0.jar
hadoop-hdfs-2.9.0.jar
hadoop-hdfs-client-2.9.0.jar
hadoop-mapreduce-client-core-2.9.0.jar
hadoop-mapreduce-client-common-2.9.0.jar
hadoop-mapreduce-client-jobclient-2.9.0.jar
hadoop-mapreduce-client-hs-2.9.0.jar
hadoop-mapreduce-client-app-2.9.0.jar
commons-io-2.4.jar
```

Once you are able to compile your code you need to create jar file.

In the eclipse IDE right click on your Java program and select Export – Java – jar file.

4. Running the MapReduce code

You can use the following command to run the program. Assuming you are in your hadoop installation directory.

```
bin/hadoop jar /netjs/MapReduce/wordcount.jar
org.netjs.WordCount /user/process /user/out
```

Explanation for the arguments passed is as follows-

/netjs/MapReduce/wordcount.jar is the path to your jar file.

org.netjs.WordCount is the fully qualified path to your Java program class.

/user/process – path to input directory.

/user/out – path to output directory.

Once your word count MapReduce program is successfully executed you can verify the output file.

```
hdfs dfs -ls /user/out
```

Found 2 items

```
-rw-r--r-- 1 netjs supergroup      0 2018-02-27 13:37 /user/out/_SUCCESS
```

```
-rw-r--r-- 1 netjs supergroup    77 2018-02-27 13:37 /user/out/part-r-00000
```

As you can see Hadoop framework creates output files using part-r-xxxx format. Since only one reducer is used here so there is only one output file part-r-00000. You can see the content of the file using the following command.

```
hdfs dfs -cat /user/out/part-r-00000
```

```
Hadoop      1
Hello       1
MapReduce   2
This        1
first       1
is          1
my          1
program.    2
wordcount   1
```

Conclusion: In this assignment, we have learned what is HDFS and How Hadoop MapReduce framework is used to count the number of occurrences of each word in a given input set.

DS&BDL

Assignment 12

Title: Design a distributed application using MapReduce which

Processes a log file of a system.

Aim:

Write a code in JAVA to design a distributed application using MapReduce which processes a log file of a system. List out the users who have logged for maximum period on the system. Use simple log file from the Internet and process it using a pseudo distribution mode on Hadoop platform.

Objective:

By completing this task, students will learn the following

1. Hadoop Distributed File System.
2. MapReduce Framework.

Software/Hardware Requirements: 64-bit Open source OS-Linux, Java, Hadoop.

Theory:

Map and Reduce tasks in Hadoop-Within a MapReduce job there are two separate tasks map task and reduce task.

Map task- A MapReduce job splits the input dataset into independent chunks known as input splits in Hadoop which are processed by the map tasks in a completely parallel manner. Hadoop framework creates separate map task for each input split.

Reduce task- The output of the maps is sorted by the Hadoop framework which then becomes input to the reduce tasks.

Hadoop MapReduce framework operates exclusively on <key, value> pairs. In a MapReduce job, the input to the Map function is a set of <key, value> pairs and output is also a set of <key, value> pairs. The output <key, value> pair may have different type from the input <key, value> pair.

$\langle K1, V1 \rangle \rightarrow \text{map} \rightarrow (K2, V2)$

The output from the map tasks is sorted by the Hadoop framework. MapReduce guarantees that the input to every reducer is sorted by key. Input and output of the reduce task can be represented as follows.

$\langle K2, \text{list}(V2) \rangle \rightarrow \text{reduce} \rightarrow \langle K3, V3 \rangle$

1. Input Data

Input to the MapReduce comes from HDFS where log files are stored on the processing cluster. By dividing log files into small blocks we can distribute them over nodes of Hadoop cluster. The format of input files to MapReduce is arbitrary but it is line-based for log files. As each line is considered as a one record as we can say one log.

2. MapReduce Algorithm

MapReduce is a simple programming model which is easily scalable over multiple nodes in a Hadoop cluster. MapReduce job is written in Java consisting of Map and Reduce function. MapReduce takes log file as an input and feeds each record in the log file to the Mapper. Mapper processes all the records in the log file and Reducer processes all the outputs from the Mapper and gives final reduced results.

Map Function: Input to the map method is the InputSplit of log file. It produces intermediate results in (key, value) pairs. For each occurrence of key it emits (key, „1“) pair. If there are n occurrences of key, then it produces n (key, „1“) pairs. OutputCollector is the utility provided by MapReduce framework to collect output from mapper and reducer and reporter is to report a progress of application.

```
Map(LongWritable key, Text value, OutputCollector output, Reporter reporter) { For each key in the value; EmitIntermediate(key, „1“); }
```

Reduce Function: Input to reduce method is (key, values) pairs. It sums together all counts emitted by map method. If input to the reduce method is (key, (1,1,1,...n times)) then it aggregates all the values for that key producing output (key, n) pair. OutputCollector and Reporter works in similar way as in map method.

```
reduce(Text key, Iterator values, OutputCollector output, Reporter reporter)
```

```
{  
    int sum = 0;  
    for each v in values;  
        sum += ParseInt(v);  
    output.collect(key,(sum));  
}
```

3. Creating Pig Query

Pig queries are written in Pig Latin language. Pig Latin statements are generally organized in the following manner:

A LOAD statement reads data from the Hadoop file system.

A series of "transformation" statements process the data.

A STORE statement writes output to the Hadoop file system.

Conclusion:

In this assignment, we have learned what is HDFS and How Hadoop MapReduce framework is used to process a log file of system

DS&BDL

Assignment 13

Title: Locate dataset (e.g., sample_weather.txt) for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed.

Aim:

Write a code in JAVA for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed.

Objective:

By completing this task, students will learn the following

1. Hadoop Distributed File System.
2. MapReduce Framework.

Software/Hardware Requirements: 64-bit Open source OS-Linux, Java, Hadoop.

Theory:

Here, we will write a Map-Reduce program for analyzing weather datasets to understand its data processing programming model. Weather sensors are collecting weather information across the globe in a large volume of log data. This weather data is semi-structured and record-oriented. This data is stored in a line-oriented ASCII format, where each row represents a single record. Each row has lots of fields like longitude, latitude, daily max-min temperature, daily average temperature, etc. for easiness, we will focus on the main element, i.e. temperature. We will use the data from the National Centres for Environmental Information(NCEI). It has a massive amount of historical weather data that we can use for our data analysis.

1. Input Data

We can download the dataset for various cities in different years. choose the year of your choice and select any one of the data text-file for analyzing. We can get information about data from *README.txt* file available on the NCEI website.

2. Make a project in Eclipse with below steps:

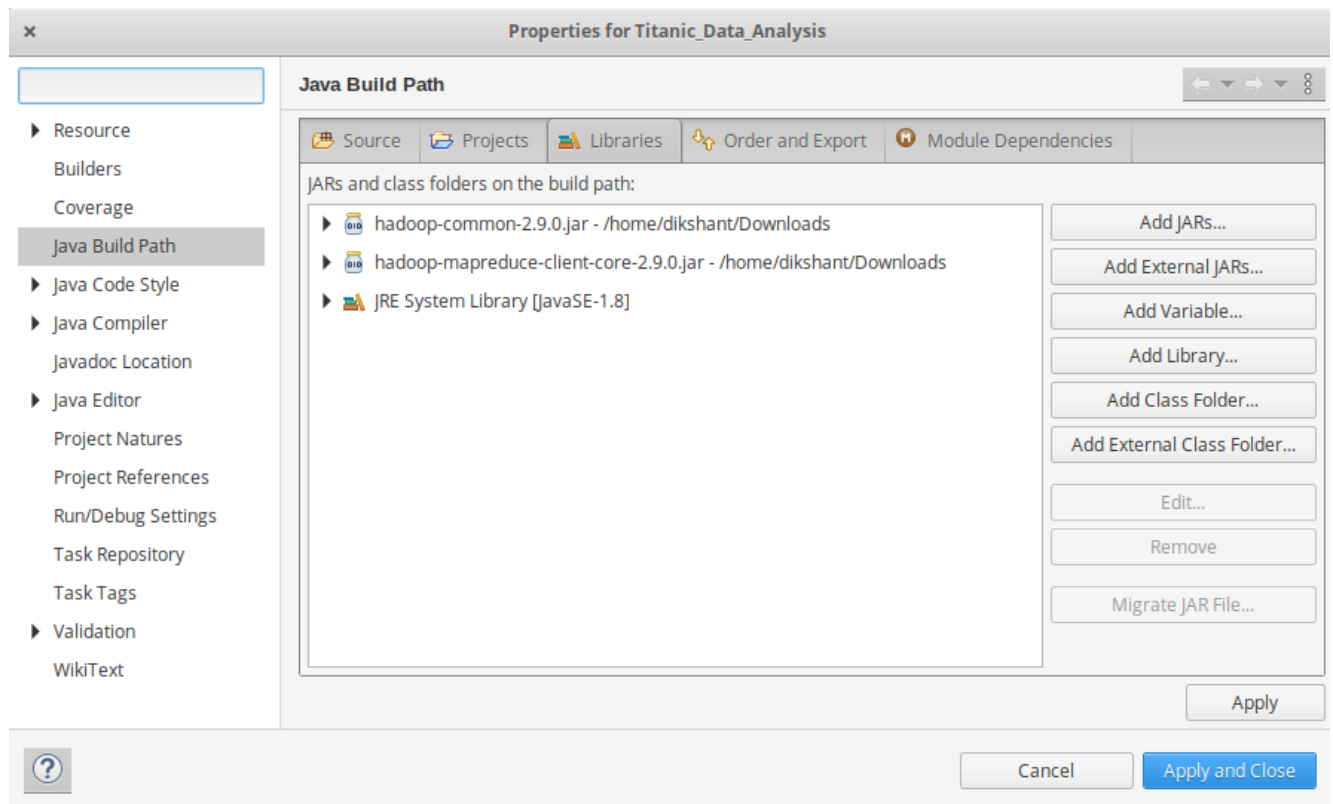
First Open Eclipse -> then select File -> New -> Java Project ->Name it MyProject -> then select use an execution environment -> choose JavaSE-1.8 then next -> Finish.

In this Project Create Java class with name **MyMaxMin** -> then click **Finish**

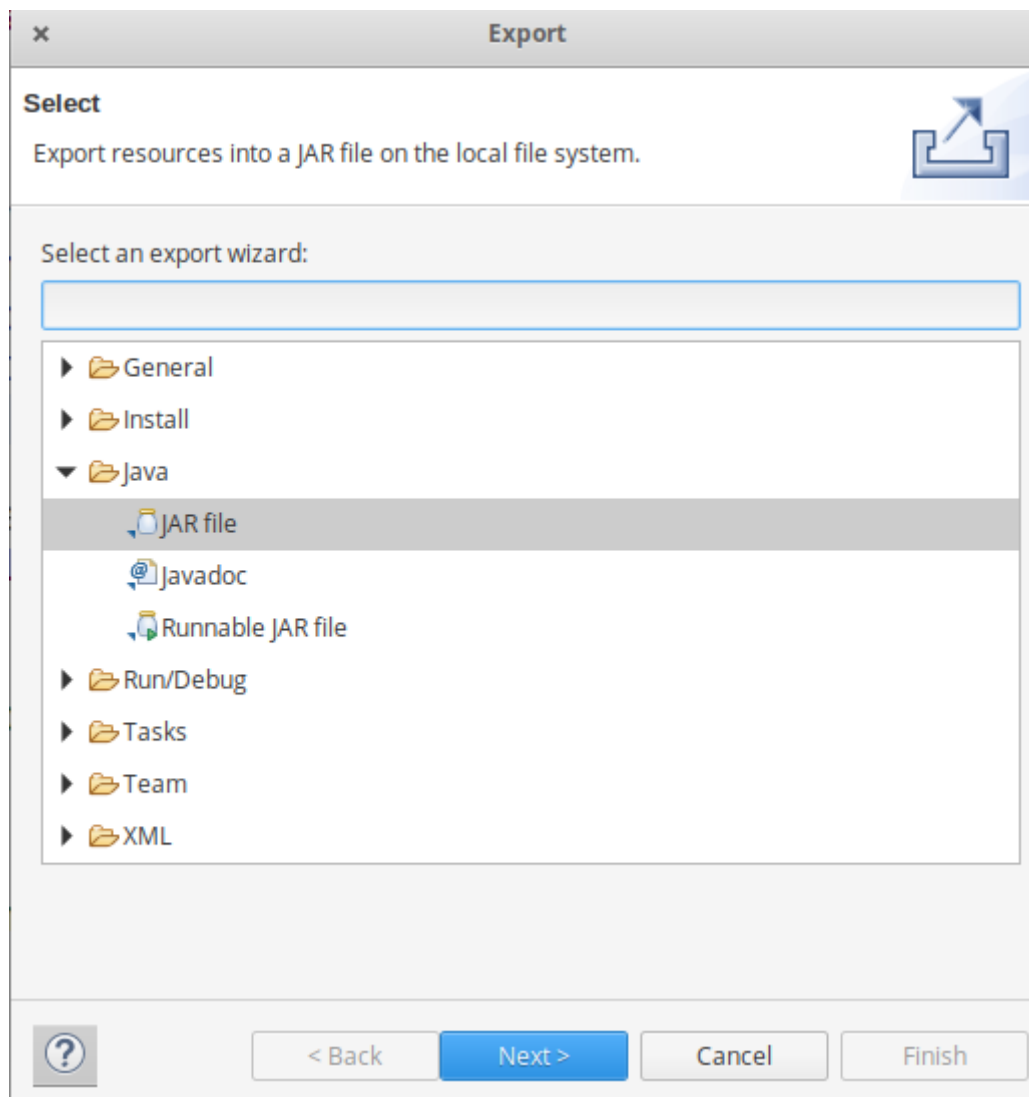
Write code for mapper and reducer in MyMaxMin class.

Now we need to add external jar for the packages that we have import. Download the jar package Hadoop Common and Hadoop MapReduce Core according to your Hadoop version.

3 . Now we add these external jars to our MyProject. Right Click on MyProject -> then select Build Path-> Click on Configure Build Path and select Add External jars.... and add jars from it's download location then click -> Apply and Close.



4. Now export the project as jar file. Right-click on MyProject choose Export.. and go to Java -> JAR file click -> Next and choose your export destination then click -> Next. choose Main Class as MyMaxMin by clicking -> Browse and then click -> Finish -> Ok.



5. Start our Hadoop Daemons

start-dfs.sh

start-yarn.sh

6. Move your dataset to the Hadoop HDFS.

hdfs dfs -put /file_path /destination

7. Now Run your Jar File with below command and produce the output in **MyOutput** File.

hadoop jar /jar_file_location /dataset_location_in_HDFS /output-file_name

8. Now Move to *localhost:50070/*, under utilities select *Browse the file system* and download **part-r-00000** in **/MyOutput** directory to see result.

| <input type="checkbox"/> | | Permission | | Owner | | Group | | Size | | Last Modified | | Replication | | Block Size | | Name | |
|--------------------------|--|-------------|--|----------|--|------------|--|----------|--|---------------|--|-------------|--|------------|--|--------------------------------------|--|
| <input type="checkbox"/> | | -rw-r--r-- | | dikshant | | supergroup | | 38.78 KB | | Jul 04 09:39 | | 1 | | 122.07 MB | | CRND0103-2020-AK_Fairbanks_11_NE.txt | |
| <input type="checkbox"/> | | drwxrwxr-x+ | | dikshant | | supergroup | | 0 B | | Jun 23 14:23 | | 0 | | 0 B | | Hadoop_File | |
| <input type="checkbox"/> | | drwxr-xr-x | | dikshant | | supergroup | | 0 B | | Jul 04 09:44 | | 0 | | 0 B | | MyOutput | |
| <input type="checkbox"/> | | drwxrwxrwx | | dikshant | | supergroup | | 0 B | | Jun 14 21:43 | | 0 | | 0 B | | tmp | |
| <input type="checkbox"/> | | drwxr-xr-x | | dikshant | | supergroup | | 0 B | | Jun 14 21:43 | | 0 | | 0 B | | user | |

Show entries Search:

| <input type="checkbox"/> | | Permission | | Owner | | Group | | Size | | Last Modified | | Replication | | Block Size | | Name | |
|--------------------------|--|------------|--|----------|--|------------|--|---------|--|---------------|--|-------------|--|------------|--|--------------|--|
| <input type="checkbox"/> | | -rw-r--r-- | | dikshant | | supergroup | | 0 B | | Jul 04 09:44 | | 1 | | 122.07 MB | | _SUCCESS | |
| <input type="checkbox"/> | | -rw-r--r-- | | dikshant | | supergroup | | 3.85 KB | | Jul 04 09:44 | | 1 | | 122.07 MB | | part-r-00000 | |

9. See the result in the Downloaded File.

For Example, **20200101** means year = 2020

month = 01

Date = 01

Conclusion:

In this assignment, we have learned how to locate dataset in Hadoop through jar files.