ASSIGNMENT-1 - Data Wrangling 1

Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

Importing Packages

In [63]:
```
import pandas as pd
import numpy as np
```

Loading Dataset into python dataframe

In [64]:
```
df=pd.read_csv('train.csv')
```

In [65]:
```
df
```

Out[65]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	I
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	I
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	

891 rows × 12 columns

Description of Dataset

PassengerID: ID given to each passenger
Survived: Survival (0=No, 1=Yes)
Pclass: Class of passenger(1=1st, 2=2nd, 3=3rd)
Name: Name of passenger
Sex: Genger of passenger
Age: Age of passenger
SibSp: Number of Siblings/ Spouses Onboard
Parch: Number of Parents/ Children Onboard
Ticket: Ticket Number
Fare: Passenger Fare (Pounds)
Cabin: Cabin Number
Embarked: Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

In [66]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
```

```
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [67]:
```
df.describe()
```

Out[67]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [68]:
```
df.shape
```

Out[68]:
```
(891, 12)
```

In [69]:
```
df.head()
```

Out[69]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Eml
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Eml
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	

In [70]:
```
df.head(10)
```

Out[70]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Em
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	

Checking NULL Values

In [71]:
```
df.isnull()
```

Out[71]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

891 rows × 12 columns

In [72]:
```
df.isnull().sum()
```

Out[72]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Handling NULL values

Dropping Entire Column

Here, if the percentage of NULL value in a column is more than 70 percent then we are dropping it.

In [75]:
```
drop_col=df.isnull().sum()[df.isnull().sum()>(70/100 * df.shape[0])]
drop_col
```

Out[75]:
```
Cabin    687
dtype: int64
```

In [76]:
```
# Drop the columns where NULL values are more than 70 percent
df.drop(drop_col.index, inplace = True, axis=1)
```

In [77]:
```
df.head()
```

Out[77]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S

In [78]:
```
df.isnull().sum()
```

Out[78]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Embarked         2
dtype: int64
```

Replacing with mean

The NULL values in the age column will be filled with the mean age of the passengers

In [79]:
```
mean=np.mean(df['Age'])
mean
```

Out[79]:
```
29.69911764705882
```

In [80]:
```
df['Age'].fillna(mean, inplace=True)
```

In [81]:
```
df.isnull().sum()
```

Out[81]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       2
dtype: int64
```

Replacing with most frequent value

In the Embarked column, we only have a few missing entries. So it is logical that we fill it with the most frequent value.

In [82]:
```
df['Embarked'].describe()
```

Out[82]:
```
count     889
unique      3
top         S
freq      644
Name: Embarked, dtype: object
```

In [83]:
```
df['Embarked'].fillna(value='S', inplace=True)
```

In [84]:
```
df.isnull().sum()
```

Out[84]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

So, we can see that we have handled all the NULL values in the dataset

In [85]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Embarked     891 non-null    object
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

In [86]:
```
df.head()
```

Out[86]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S

Turning Categorical variable into quantitative values

Here we will change the categorical values to quantitative values. For example, in the Sex column, if it is male we will replace it with 0 and if it is female we will replace it with 1.

In [87]:
```
df['Sex'] = df['Sex'].replace(to_replace=['male','female'],value=[0,1])
```

In [88]:

```
df.head()
```

Out[88]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	0	35.0	0	0	373450	8.0500	S

ASSIGNMENT-2 -Data Wrangling 2

Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

Import libraries

In [79]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [80]:
```
df=pd.read_csv("student-por.csv")
```

In [81]:
```
df.head()
```

Out[81]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	g
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	

5 rows × 33 columns

1 school - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
2 sex - student's sex (binary: 'F' - female or 'M' - male)
3 age - student's age (numeric: from 15 to 22)
4 address - student's home address type (binary: 'U' - urban or 'R' - rural)
5 famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)
6 Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)
7 Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education)
8 Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education)
9 Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
10 Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or

police), 'at_home' or 'other')
11 reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')
12 guardian - student's guardian (nominal: 'mother', 'father' or 'other')
13 traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
14 studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
15 failures - number of past class failures (numeric: n if 1<=n<3, else 4)
16 schoolsup - extra educational support (binary: yes or no)
17 famsup - family educational support (binary: yes or no)
18 paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
19 activities - extra-curricular activities (binary: yes or no)
20 nursery - attended nursery school (binary: yes or no)
21 higher - wants to take higher education (binary: yes or no)
22 internet - Internet access at home (binary: yes or no)
23 romantic - with a romantic relationship (binary: yes or no)
24 famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
25 freetime - free time after school (numeric: from 1 - very low to 5 - very high)
26 goout - going out with friends (numeric: from 1 - very low to 5 - very high)
27 Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
28 Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
29 health - current health status (numeric: from 1 - very bad to 5 - very good)
30 absences - number of school absences (numeric: from 0 to 93)

these grades are related with the course subject, Math or Portuguese:

31 G1 - first period grade (numeric: from 0 to 20)
31 G2 - second period grade (numeric: from 0 to 20)
32 G3 - final grade (numeric: from 0 to 20, output target)

Scanning For Missing Values

In [82]:
```
df.isnull().sum()
```

Out[82]:
```
school        0
sex           0
age           0
address       0
famsize       0
Pstatus       0
Medu          0
Fedu          0
Mjob          0
```

```
Fjob          0
reason        0
guardian      0
traveltime    0
studytime     0
failures      0
schoolsup     0
famsup        0
paid          0
activities    0
nursery       0
higher        0
internet      0
romantic      0
famrel        0
freetime      0
goout         0
Dalc          0
Walc          0
health        0
absences      0
G1            0
G2            0
G3            0
dtype: int64
```

In [83]:
```
df.duplicated().sum()
```

Out[83]:
```
0
```

In [84]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 649 entries, 0 to 648
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   school      649 non-null    object
 1   sex         649 non-null    object
 2   age         649 non-null    int64 
 3   address     649 non-null    object
 4   famsize     649 non-null    object
 5   Pstatus     649 non-null    object
 6   Medu        649 non-null    int64 
 7   Fedu        649 non-null    int64 
 8   Mjob        649 non-null    object
 9   Fjob        649 non-null    object
 10  reason      649 non-null    object
 11  guardian    649 non-null    object
 12  traveltime  649 non-null    int64 
 13  studytime   649 non-null    int64 
 14  failures    649 non-null    int64 
 15  schoolsup   649 non-null    object
 16  famsup      649 non-null    object
 17  paid        649 non-null    object
 18  activities  649 non-null    object
 19  nursery     649 non-null    object
 20  higher      649 non-null    object
```

```
 21  internet    649 non-null    object
 22  romantic    649 non-null    object
 23  famrel      649 non-null    int64
 24  freetime    649 non-null    int64
 25  goout       649 non-null    int64
 26  Dalc        649 non-null    int64
 27  Walc        649 non-null    int64
 28  health      649 non-null    int64
 29  absences    649 non-null    int64
 30  G1          649 non-null    int64
 31  G2          649 non-null    int64
 32  G3          649 non-null    int64
dtypes: int64(16), object(17)
memory usage: 167.4+ KB
```

In [85]:
```
df.describe()
```

Out[85]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime
count	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000
mean	16.744222	2.514638	2.306626	1.568567	1.930663	0.221880	3.930663	3.180277
std	1.218138	1.134552	1.099931	0.748660	0.829510	0.593235	0.955717	1.051093
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000
25%	16.000000	2.000000	1.000000	1.000000	1.000000	0.000000	4.000000	3.000000
50%	17.000000	2.000000	2.000000	1.000000	2.000000	0.000000	4.000000	3.000000
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000000	4.000000
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000	5.000000

In [86]:
```
data=df.select_dtypes(include='int64')
```

In [87]:
```
data.head()
```

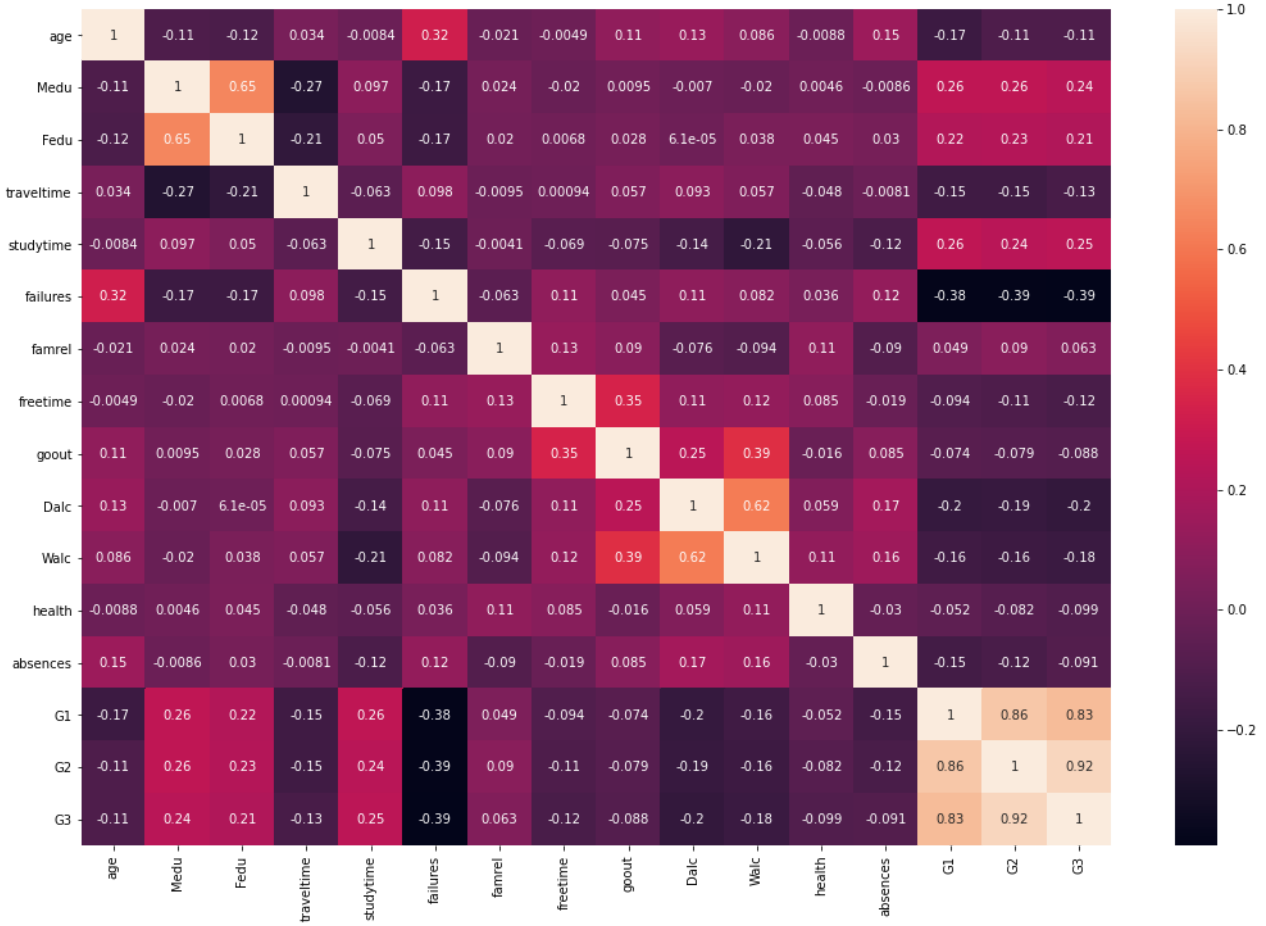
Out[87]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc	health	abs
0	18	4	4	2	2	0	4	3	4	1	1	3	
1	17	1	1	1	2	0	5	3	3	1	1	3	
2	15	1	1	1	2	0	4	3	2	2	3	3	
3	15	4	2	1	3	0	3	2	2	1	1	5	
4	16	3	3	1	2	0	4	3	2	1	2	5	

In [88]:
```
columns=data.columns
```

In [89]:

```
plt.figure(figsize=(18,12))
sns.heatmap(df.corr(),cbar=True,annot=True)
plt.show()
```



StandardScaler

StandardScaler follows Standard Normal Distribution (SND). Therefore, it makes mean = 0 and scales the data to unit variance.

In [90]:
```
from sklearn.preprocessing import StandardScaler
Scaler=StandardScaler()
```

In [91]:
```
copy_data=data.copy()
copy_data=Scaler.fit_transform(data,y=None)
copy_data=pd.DataFrame(copy_data,columns=data.columns)
copy_data.head()
```

Out[91]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	
0	1.031695	1.310216	1.540715	0.576718	0.083653	-0.374305	0.072606	-0.171647	0.693785	-0.
1	0.210137	-1.336039	-1.188832	-0.760032	0.083653	-0.374305	1.119748	-0.171647	-0.157380	-0.
2	-1.432980	-1.336039	-1.188832	-0.760032	0.083653	-0.374305	0.072606	-0.171647	-1.008546	0.

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	
3	-1.432980	1.310216	-0.278983	-0.760032	1.290114	-0.374305	-0.974536	-1.123771	-1.008546	-0.
4	-0.611422	0.428131	0.630866	-0.760032	0.083653	-0.374305	0.072606	-0.171647	-1.008546	-0.

In [92]:
```
copy_data.describe()
```

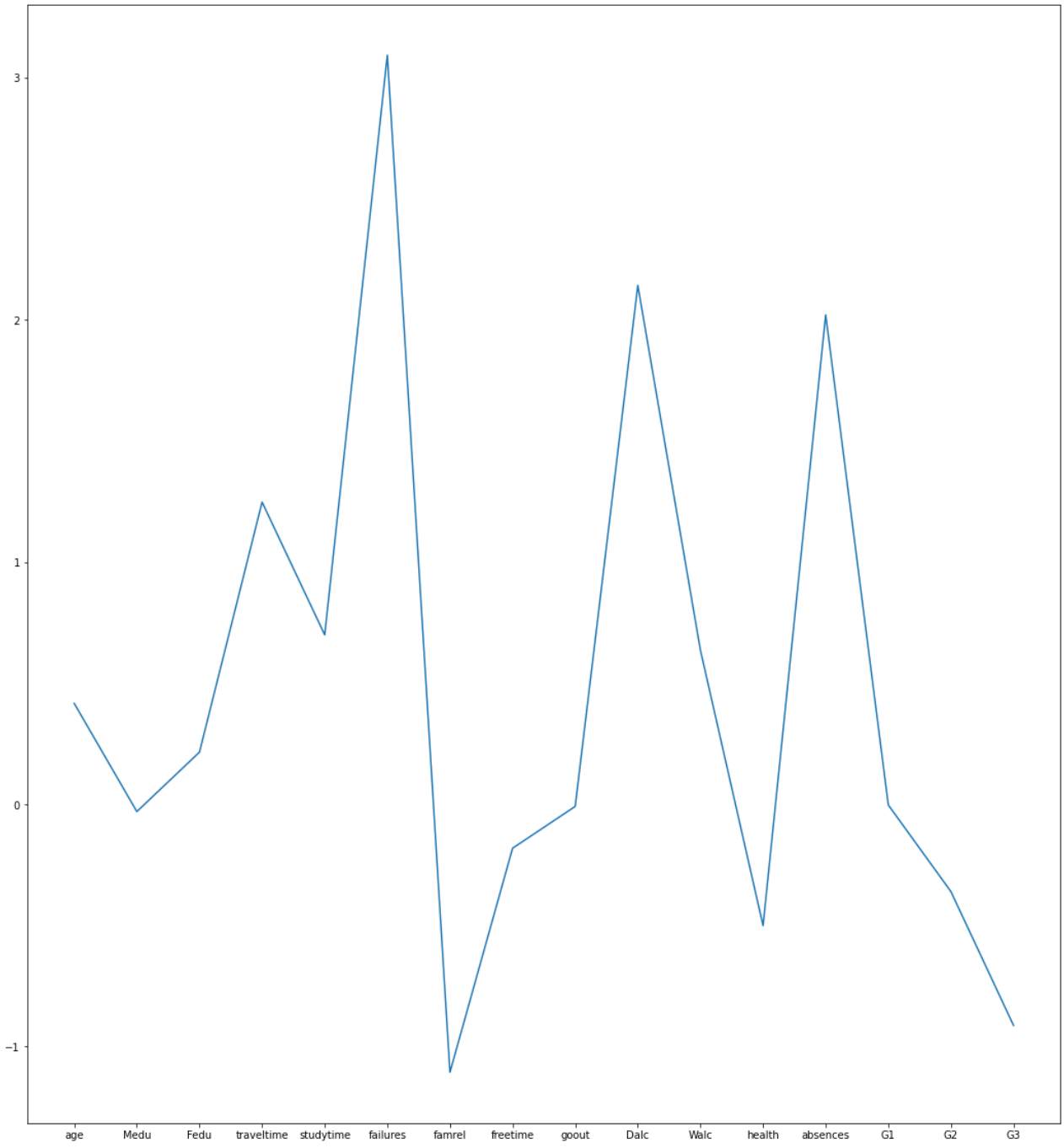
Out[92]:

	age	Medu	Fedu	traveltime	studytime	failures	
count	6.490000e+02	6.490000e+02	6.490000e+02	6.490000e+02	6.490000e+02	6.490000e+02	6.490
mean	-5.053311e-16	-5.787187e-16	2.468493e-16	-7.349026e-16	1.662769e-16	-5.109763e-16	-2.047
std	1.000771e+00	1.000771e+00	1.000771e+00	1.000771e+00	1.000771e+00	1.000771e+00	1.000
min	-1.432980e+00	-2.218124e+00	-2.098682e+00	-7.600319e-01	-1.122808e+00	-3.743051e-01	-3.068
25%	-6.114218e-01	-4.539544e-01	-1.188832e+00	-7.600319e-01	-1.122808e+00	-3.743051e-01	7.260
50%	2.101367e-01	-4.539544e-01	-2.789831e-01	-7.600319e-01	8.365295e-02	-3.743051e-01	7.260
75%	1.031695e+00	1.310216e+00	6.308662e-01	5.767180e-01	8.365295e-02	-3.743051e-01	1.119
max	4.317929e+00	1.310216e+00	1.540715e+00	3.250218e+00	2.496576e+00	4.686612e+00	1.119

Skewness

refers to a distortion or asymmetry that deviates from the symmetrical bell curve, or normal distribution, in a set of data.

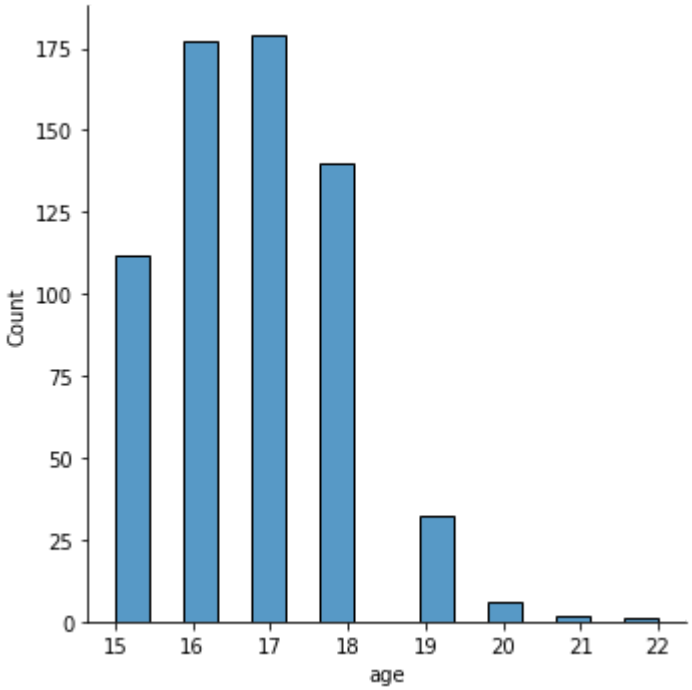
In [93]:
```
plt.figure(figsize=(18,20))
plt.plot(data.skew())
plt.show()
```

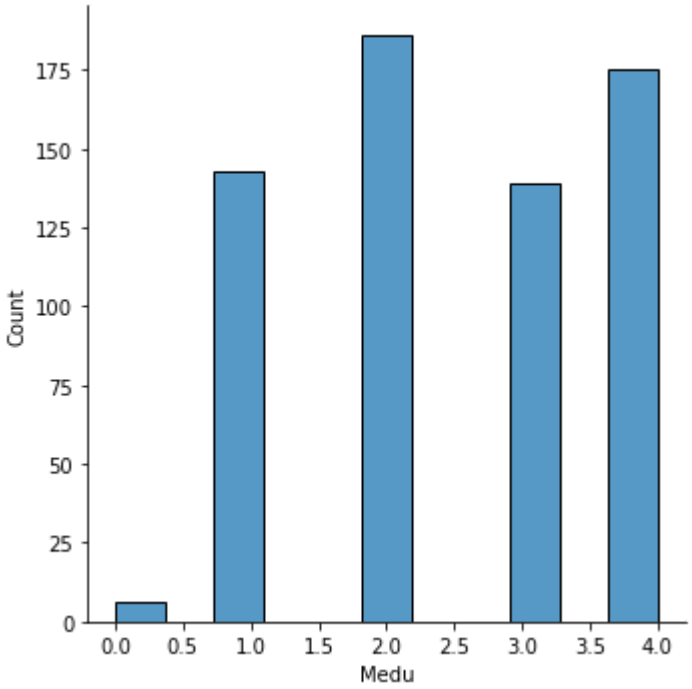


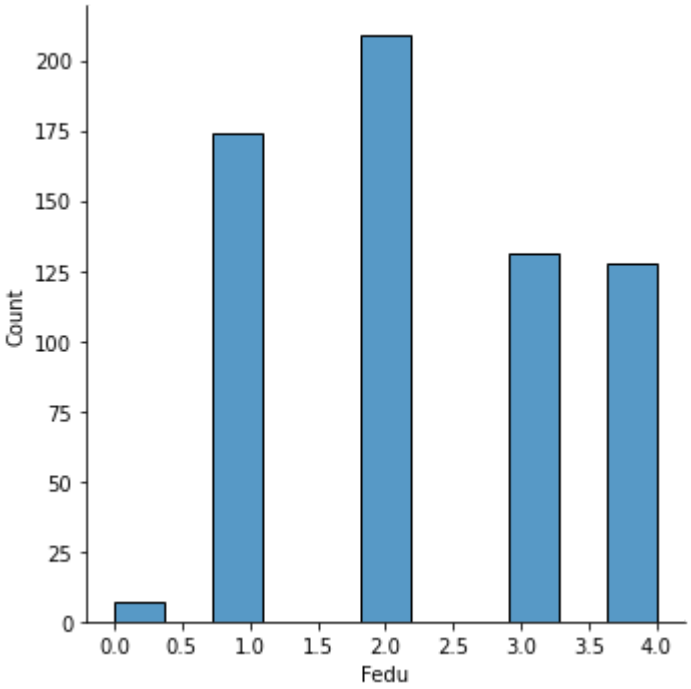
In [94]:

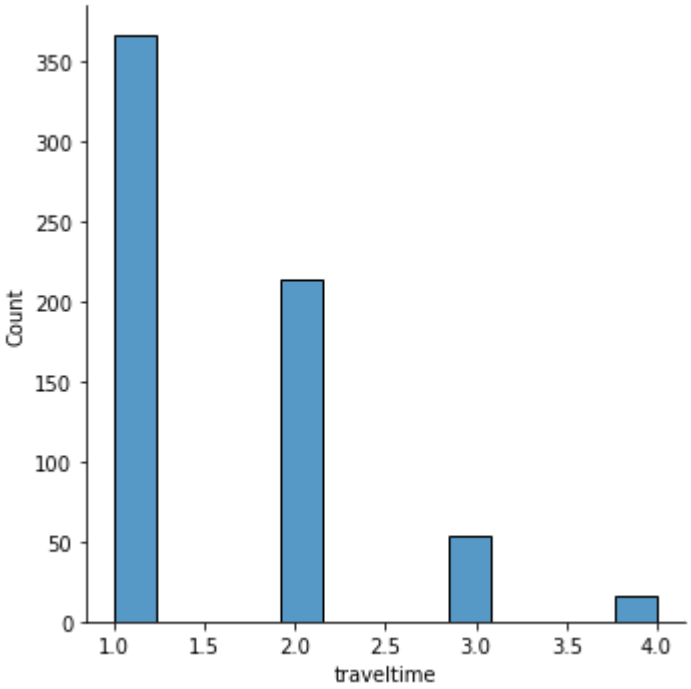
```
warnings.filterwarnings('ignore')
features_=data.columns.values[:]
fig=plt.figure(figsize=(20,10))
for columns, feature in enumerate(features_):
    sns.displot(data[feature])
plt.show()
```

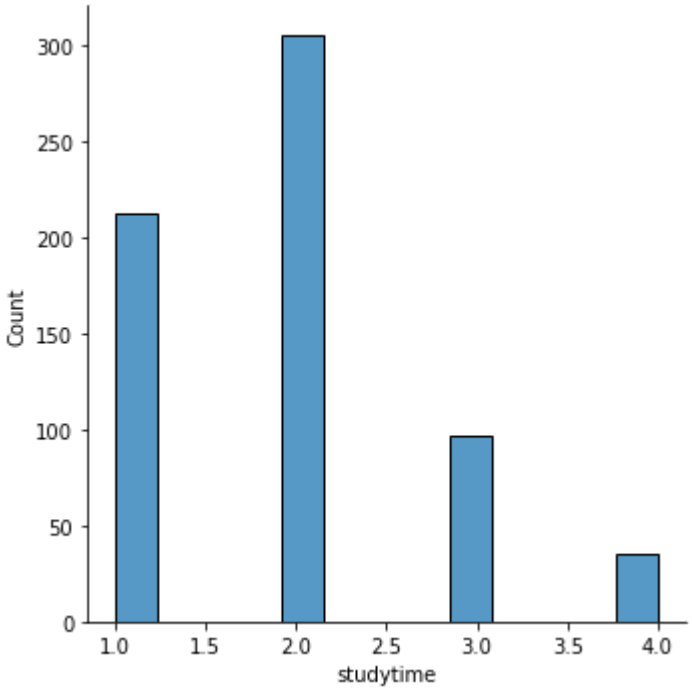
```
<Figure size 1440x720 with 0 Axes>
```

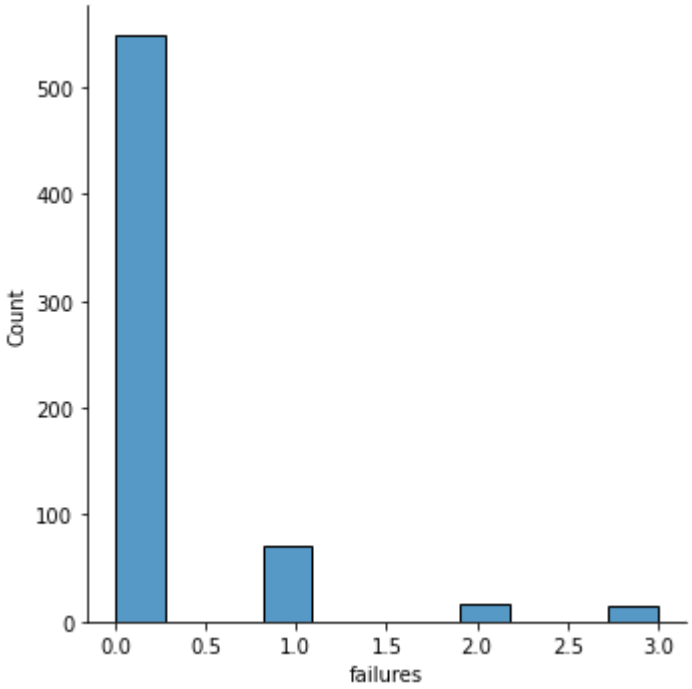

Count
175
150
125
100
75
50
25
0
15
16
17
18
19
20
21
22
age

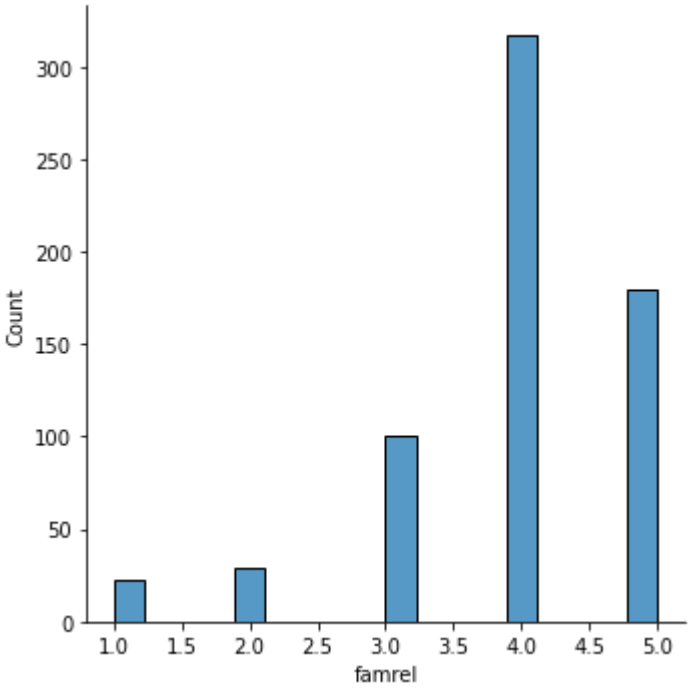

Count
175
150
125
100
75
50
25
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Medu

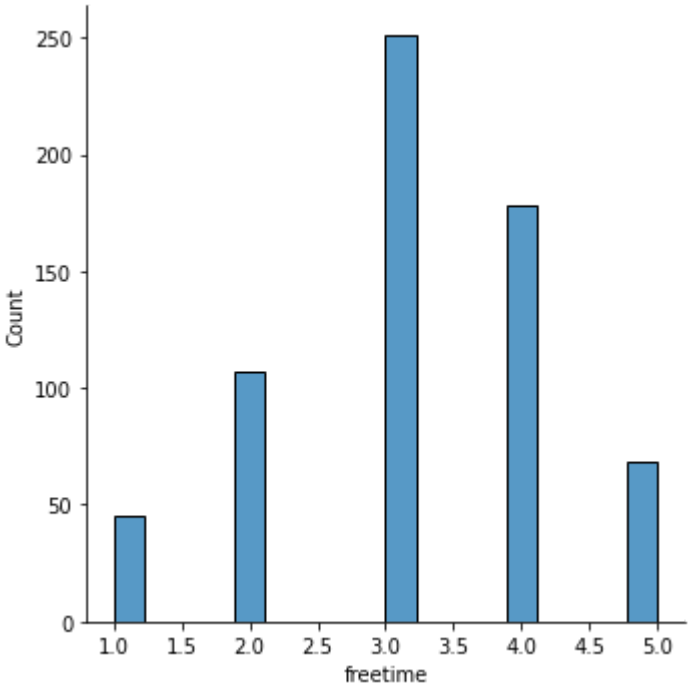

Count
200
175
150
125
100
75
50
25
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Fedu

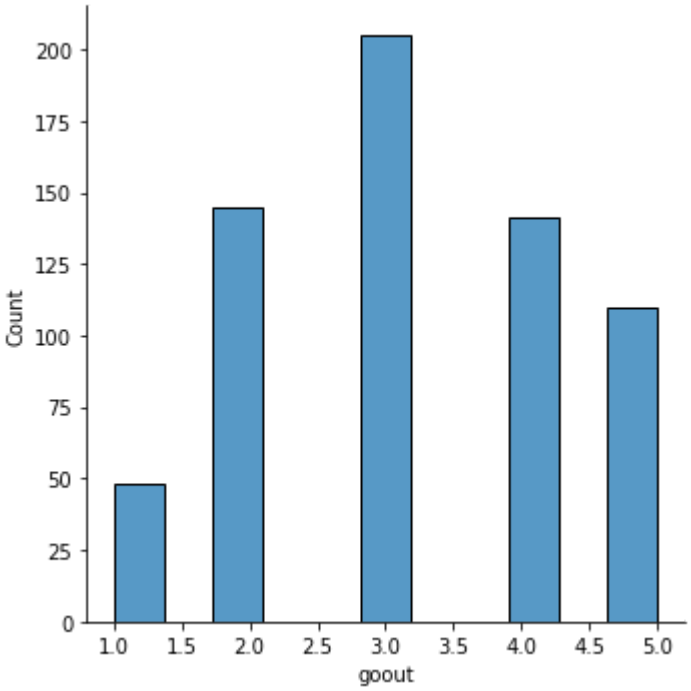

Count
350
300
250
200
150
100
50
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
traveltime

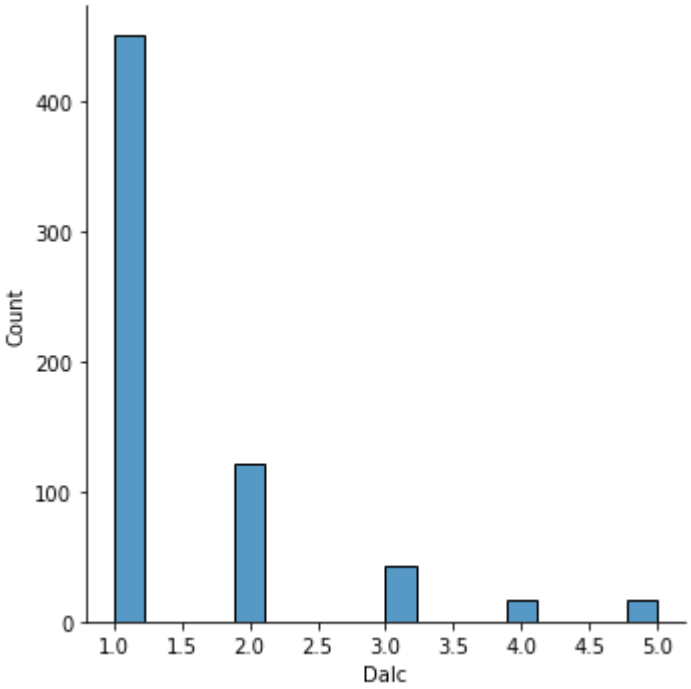

300
250
200
150
100
50
0
Count
1.0
1.5
2.0
2.5
3.0
3.5
4.0
studytime

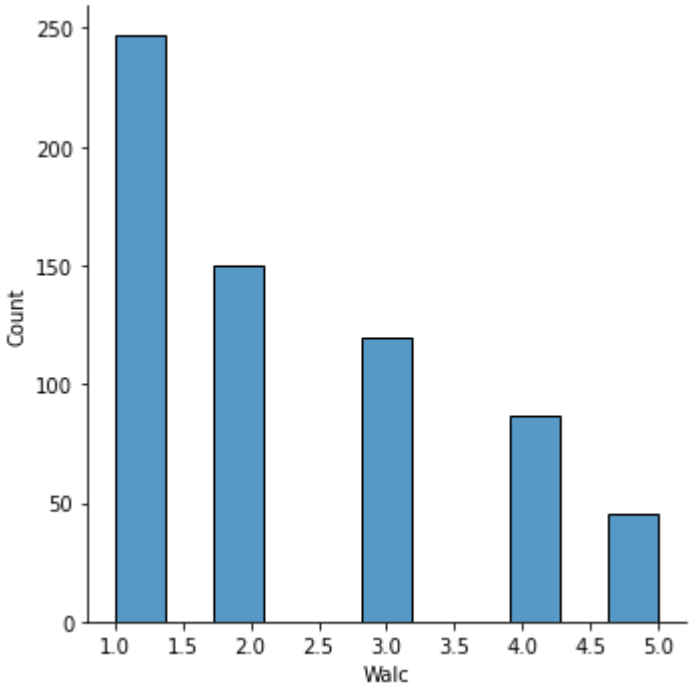

500
400
300
200
100
0
Count
0.0
0.5
1.0
1.5
2.0
2.5
3.0
failures

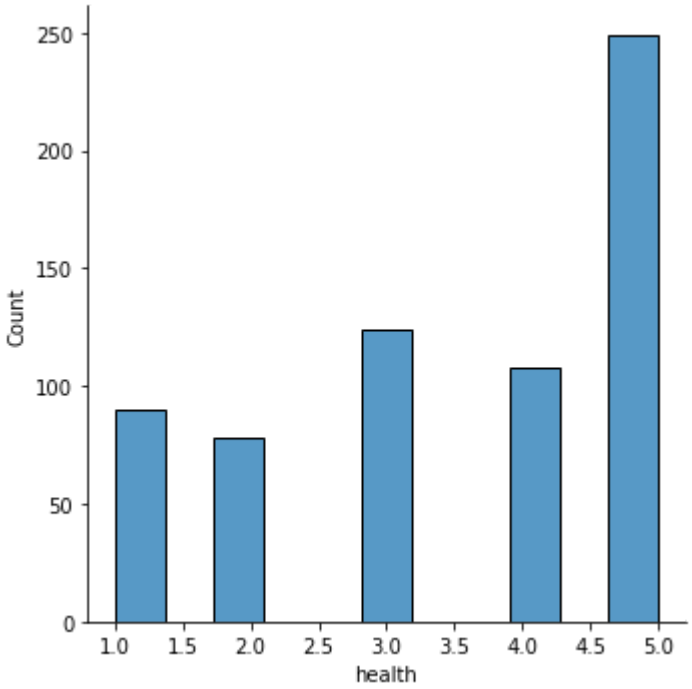

Count
300
250
200
150
100
50
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
famrel

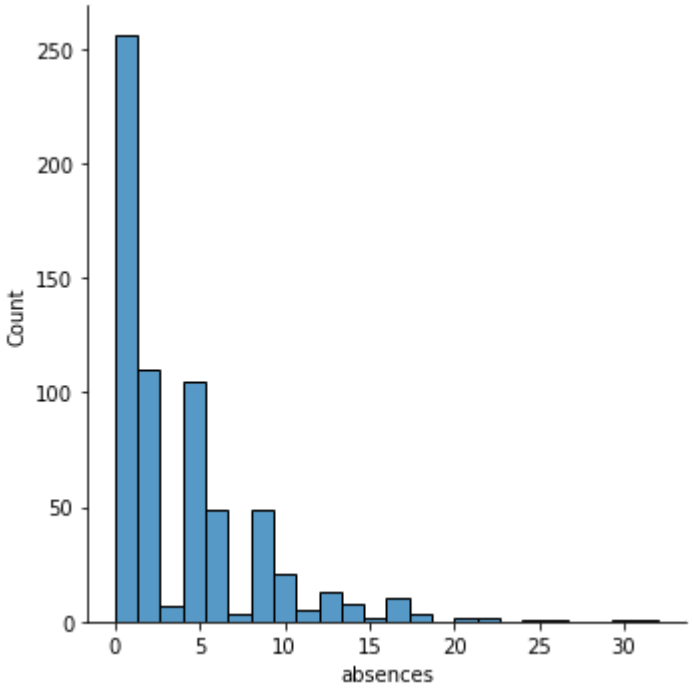

Count
250
200
150
100
50
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
freetime

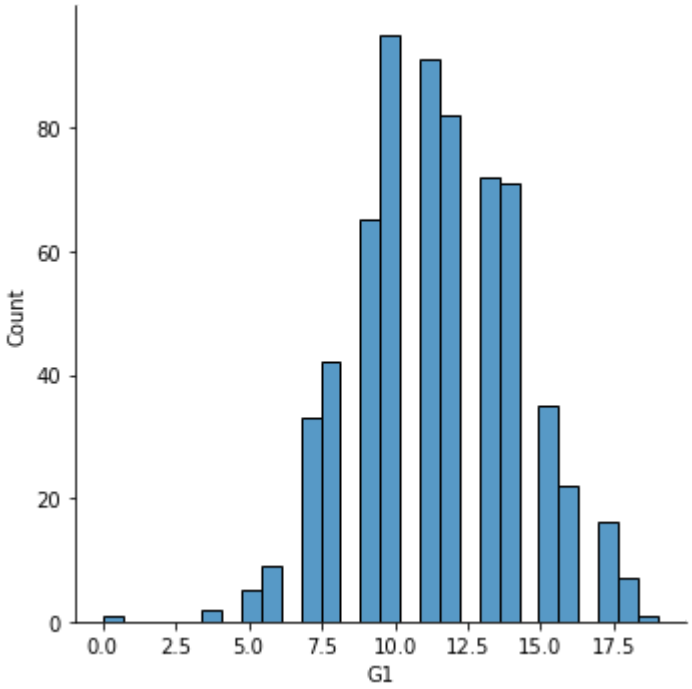

Count
goout
0
25
50
75
100
125
150
175
200
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

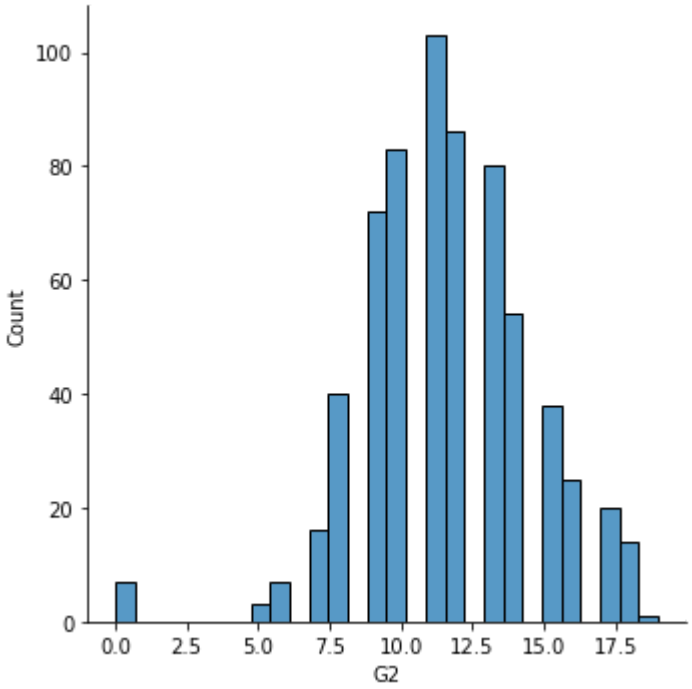

Count
Dalc
0
100
200
300
400
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

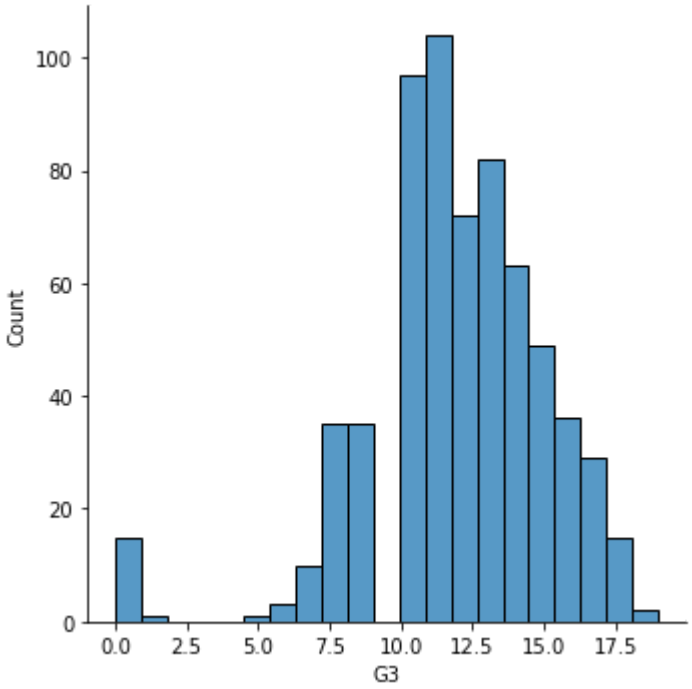

250
200
150
100
50
0
Count
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Walc


250
200
150
100
50
0
Count
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
health


250
200
150
100
50
0
Count
0
5
10
15
20
25
30
absences


80
60
40
20
0
Count
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
G1

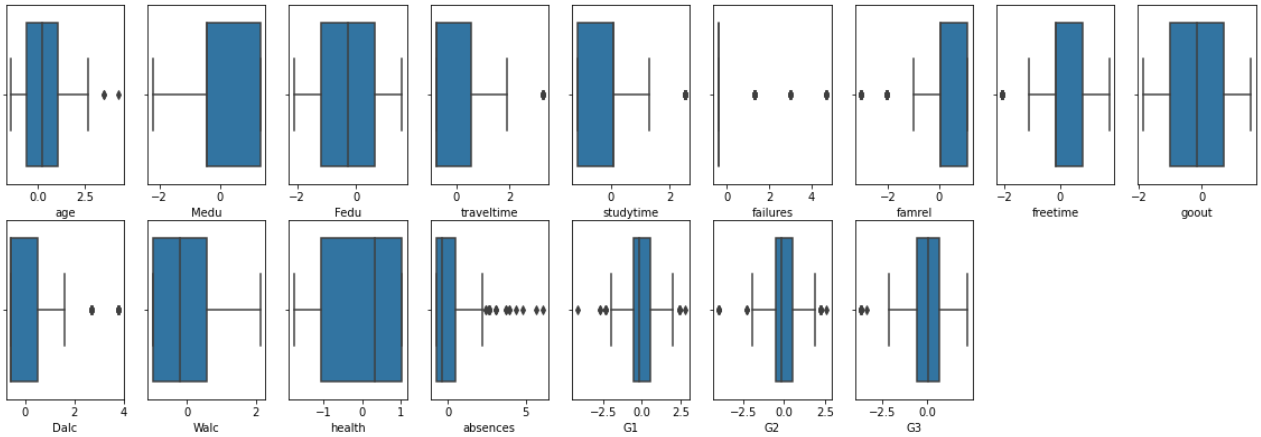




Scanning For Outliers

In [95]:
```
import warnings
```

In [97]:
```
warnings.filterwarnings('ignore')
features_=copy_data.columns.values[:]
fig=plt.figure(figsize=(20,10))
for columns, feature in enumerate(features_):
    fig.add_subplot(3,9,columns+1)
    sns.boxplot(copy_data[feature],data=copy_data)
plt.show()
```



In [99]:
```
def detect(data):
    outliers=[]
    threshhold=3
    mean=np.mean(data)
    std=np.std(data)
    for i in data:
        zscore=(i-mean)/std
        if(np.abs(zscore)>3):
            outliers.append(i)
    return outliers
```

In [100…
```
absences=detect(copy_data['absences'])
print(absences)
```

```
[4.3863974667881624, 3.9551010565244673, 6.111583107842941, 5.6802866975792465, 3.739452
85139262, 3.092508235997078, 4.817693877051857, 3.9551010565244673, 3.092508235997078,
3.092508235997078, 3.73945285139262]
```

In [101…
```
def replace(data):
    outliers=[]
    threshhold=3
    mean=np.mean(data)
    std=np.std(data)
    for i in data:
        zscore=(i-mean)/std
        if(zscore>3):
            data.replace(i,3,inplace=True)
            outliers.append(i)
        if(zscore<-3):
            data.replace(i,-3,inplace=True)
            outliers.append(i)
    return outliers
```

In [102…
```
replace(copy_data['absences'])
```

Out[102…
```
[4.3863974667881624,
 3.9551010565244673,
 6.111583107842941,
 5.6802866975792465,
 3.73945285139262,
```

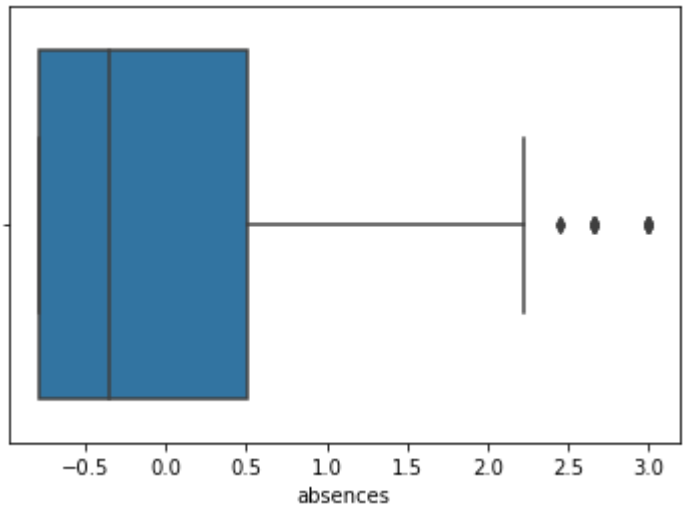
```
3.092508235997078,
4.817693877051857]
```

In [103…

```
sns.boxplot(copy_data['absences'])
```

Out[103…

```
<AxesSubplot:xlabel='absences'>
```



Assignment 3 - Descriptive Statistics

Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
df=pd.read_csv('iris flower.csv')
```

The **Iris Dataset** contains four features (length and width of sepals and petals) of 50 samples of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). These measures were used to create a linear discriminant model to classify the species

In [3]:
```
df
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [4]:
```
df.head()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa

	sepal_length	sepal_width	petal_length	petal_width	species
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Summary of data

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

Mean:

The mean is the average or a calculated central value of a set of numbers and is used to measure the central tendency of the data.

Mean = (Sum of Observations) ÷ (Total Numbers of Observations)

Percentile:

A percentile (or a centile) is a measure used in statistics indicating the value below which a given percentage of observations in a group of observations fall.

Percentile Value = $\mu + z\sigma$

where: μ: Mean

z: z-score from z table that corresponds to percentile value

σ: Standard deviation

Standard Deviation :

The standard deviation of a random variable, sample, statistical population, data set, or probability distribution is the square root of its variance.

$$s = \sqrt{s^2} = \sqrt{\frac{SS}{N-1}} = \sqrt{\frac{\sum(x_i - \bar{x})^2}{N-1}}$$

σ = population standard deviation

N = the size of the population

x_i = each value from the population

μ = the population mean

In [6]:
```
np.mean(df['sepal_length'])
```

Out[6]:
```
5.843333333333335
```

In [37]:
```
np.median(df['petal_length'])
```

Out[37]:
```
4.35
```

Group By categories

In [40]:
```
print(np.min(df['sepal_width']))
print(np.max(df['sepal_width']))
print(np.std(df['petal_width']))
```

```
2.0
4.4
0.760612618588172
```

In [41]:
```
df['species'].value_counts()
```

Out[41]:
```
Iris-setosa        50
Iris-virginica     50
Iris-versicolor    50
Name: species, dtype: int64
```

In [64]:
```
df.describe()
```

Out[64]:

	sepal_length	sepal_width	petal_length	petal_width
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

	sepal_length	sepal_width	petal_length	petal_width
max	7.900000	4.400000	6.900000	2.500000

In [65]:
```
df.groupby(['species']).mean()
```

Out[65]:

species	sepal_length	sepal_width	petal_length	petal_width
Iris-setosa	5.006	3.418	1.464	0.244
Iris-versicolor	5.936	2.770	4.260	1.326
Iris-virginica	6.588	2.974	5.552	2.026

In [66]:
```
df.groupby(['species']).min()
```

Out[66]:

species	sepal_length	sepal_width	petal_length	petal_width
Iris-setosa	4.3	2.3	1.0	0.1
Iris-versicolor	4.9	2.0	3.0	1.0
Iris-virginica	4.9	2.2	4.5	1.4

In [67]:
```
df.groupby(['species']).max()
```

Out[67]:

species	sepal_length	sepal_width	petal_length	petal_width
Iris-setosa	5.8	4.4	1.9	0.6
Iris-versicolor	7.0	3.4	5.1	1.8
Iris-virginica	7.9	3.8	6.9	2.5

In [68]:
```
df.groupby(['species']).std()
```

Out[68]:

species	sepal_length	sepal_width	petal_length	petal_width
Iris-setosa	0.352490	0.381024	0.173511	0.107210
Iris-versicolor	0.516171	0.313798	0.469911	0.197753
Iris-virginica	0.635880	0.322497	0.551895	0.274650

In [74]:
```
df.groupby(['species']).quantile(q=0.5)
```

Out[74]:

species	sepal_length	sepal_width	petal_length	petal_width
Iris-setosa	5.0	3.4	1.50	0.2
Iris-versicolor	5.9	2.8	4.35	1.3
Iris-virginica	6.5	3.0	5.55	2.0

In [75]:
```
df.groupby(['species']).quantile(q=0.75)
```

Out[75]:

species	sepal_length	sepal_width	petal_length	petal_width
Iris-setosa	5.2	3.675	1.575	0.3
Iris-versicolor	6.3	3.000	4.600	1.5
Iris-virginica	6.9	3.175	5.875	2.3

In [76]:
```
df.groupby(['species']).quantile(q=0.25)
```

Out[76]:

species	sepal_length	sepal_width	petal_length	petal_width
Iris-setosa	4.800	3.125	1.4	0.2
Iris-versicolor	5.600	2.525	4.0	1.2
Iris-virginica	6.225	2.800	5.1	1.8

In [96]:
```
df['sepal_length'].loc[(df['species']=='Iris-setosa')]
```

Out[96]:
```
0     5.1
1     4.9
2     4.7
3     4.6
4     5.0
5     5.4
6     4.6
7     5.0
8     4.4
9     4.9
10    5.4
11    4.8
12    4.8
13    4.3
14    5.8
15    5.7
16    5.4
17    5.1
18    5.7
19    5.1
20    5.4
```

```
21    5.1
22    4.6
23    5.1
24    4.8
25    5.0
26    5.0
27    5.2
28    5.2
29    4.7
30    4.8
31    5.4
32    5.2
33    5.5
34    4.9
35    5.0
36    5.5
37    4.9
38    4.4
39    5.1
40    5.0
41    4.5
42    4.4
43    5.0
44    5.1
45    4.8
46    5.1
47    4.6
48    5.3
49    5.0
Name: sepal_length, dtype: float64
```

In [97]:
```
np.mean(df['sepal_length'].loc[(df['species']=='Iris-setosa')])
```

Out[97]:
```
5.005999999999999
```

In [131…
```
data2=df[['sepal_length','species']].loc[(df['species']=='Iris-setosa') | (df['species'
data2=pd.DataFrame(data2)
data2.reset_index()
```

Out[131…

	index	sepal_length	species
0	0	5.1	Iris-setosa
1	1	4.9	Iris-setosa
2	2	4.7	Iris-setosa
3	3	4.6	Iris-setosa
4	4	5.0	Iris-setosa
...	...	...	...
95	145	6.7	Iris-virginica
96	146	6.3	Iris-virginica
97	147	6.5	Iris-virginica
98	148	6.2	Iris-virginica

	index	sepal_length	species
99	149	5.9	Iris-virginica

100 rows × 3 columns

In [120…
```
np.mean(df['sepal_length'].loc[(df['species']=='Iris-setosa') | (df['species']=='Iris-v
```

Out[120…
```
5.796999999999998
```

In [121…
```
data2['species'].value_counts()
```

Out[121…
```
Iris-setosa       50
Iris-virginica    50
Name: species, dtype: int64
```

In [129…
```
data2.iloc[99]
```

Out[129…
```
sepal_length              5.9
species         Iris-virginica
Name: 149, dtype: object
```

Assignment-4 -Data Analytics 1 _ Linear Regression

Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

1. Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (https://www.kaggle.com/c/boston-housing). The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset.
2. The objective is to predict the value of prices of the house using the given features.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

In [2]:
```
boston=pd.read_csv('boston.csv')
```

In [3]:
```
boston.head()
```

Out[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MED
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.

In [4]:
```
boston.shape
```

Out[4]:
```
(506, 14)
```

Input features in order: 1) CRIM: per capita crime rate by town
2) ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
3) INDUS: proportion of non-retail business acres per town
4) CHAS: Charles River dummy variable (1 if tract bounds river; 0 otherwise)
5) NOX: nitric oxides concentration (parts per 10 million) [parts/10M]
6) RM: average number of rooms per dwelling

7) AGE: proportion of owner-occupied units built prior to 1940

8) DIS: weighted distances to five Boston employment centres

9) RAD: index of accessibility to radial highways

10) TAX: full-value property-tax rate per $10,000[/10k]$

11) PTRATIO: pupil-teacher ratio by town

12) B: The result of the equation B=1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town

13) LSTAT: % lower status of the population

Output variable:

1) MEDV: Median value of owner-occupied homes in $1000's[k]$

In [5]:
```
boston.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64  
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

In [6]:
```
boston.describe()
```

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500

In [7]:
```
boston.isnull().sum()
```

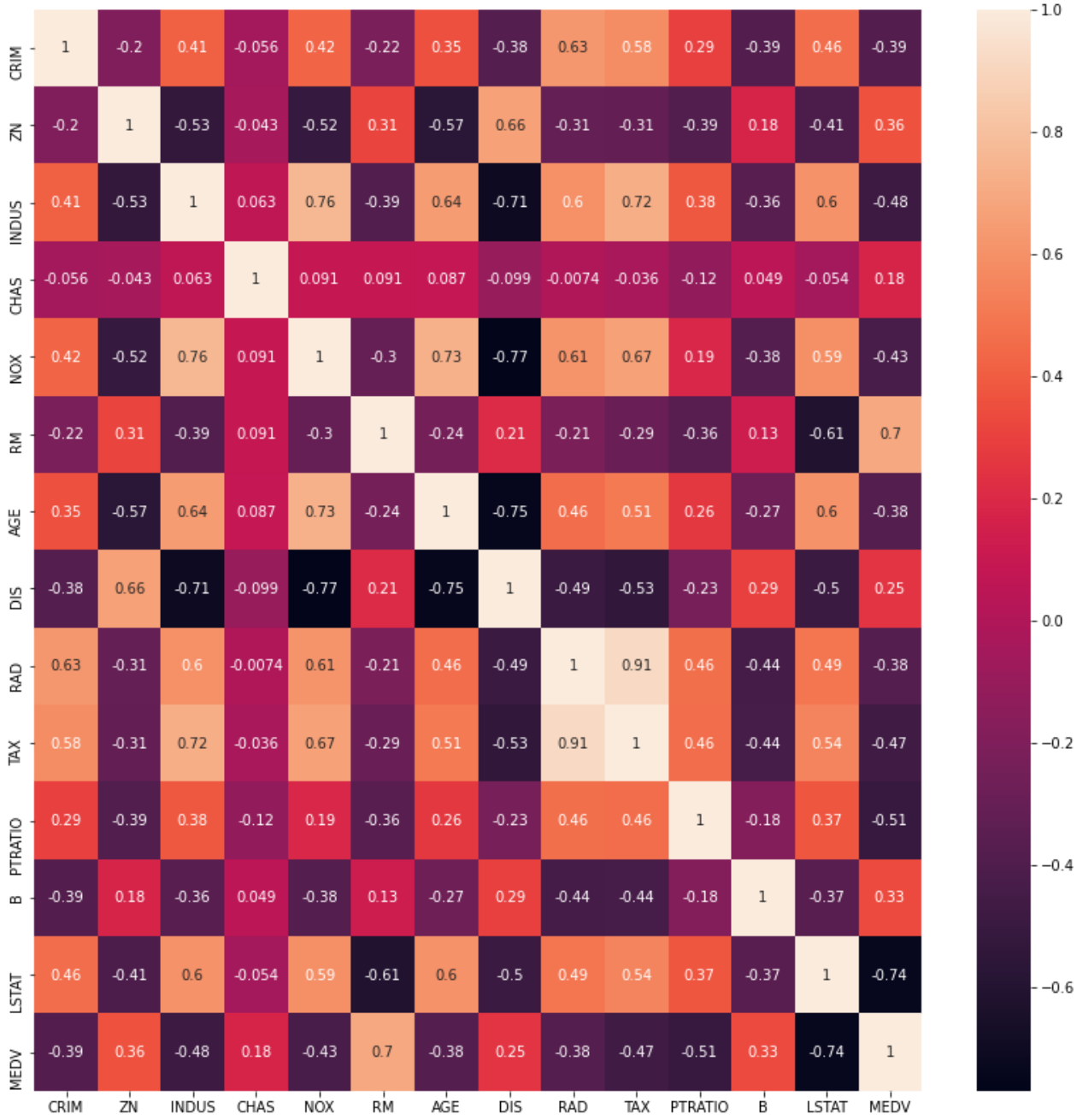
Out[7]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

In [8]:
```
boston.duplicated().sum()
```

Out[8]:
```
0
```

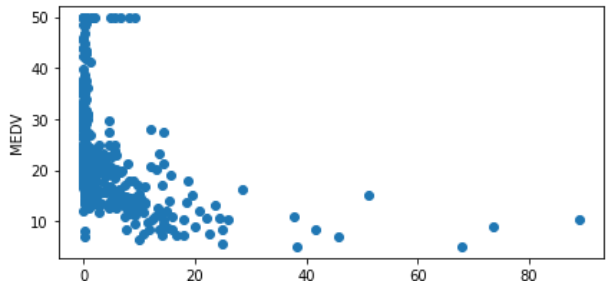
In [9]:
```
corr_m=boston.corr()
plt.figure(figsize=(14,14))
sns.heatmap(data=corr_m, annot=True)
```

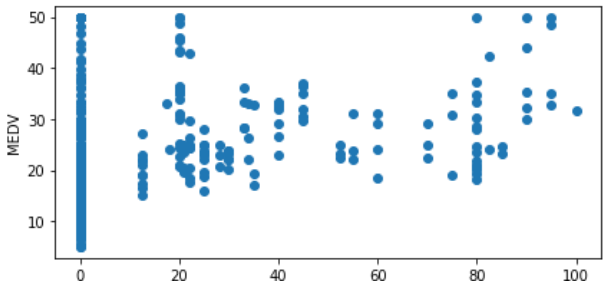
Out[9]:
```
<AxesSubplot:>
```

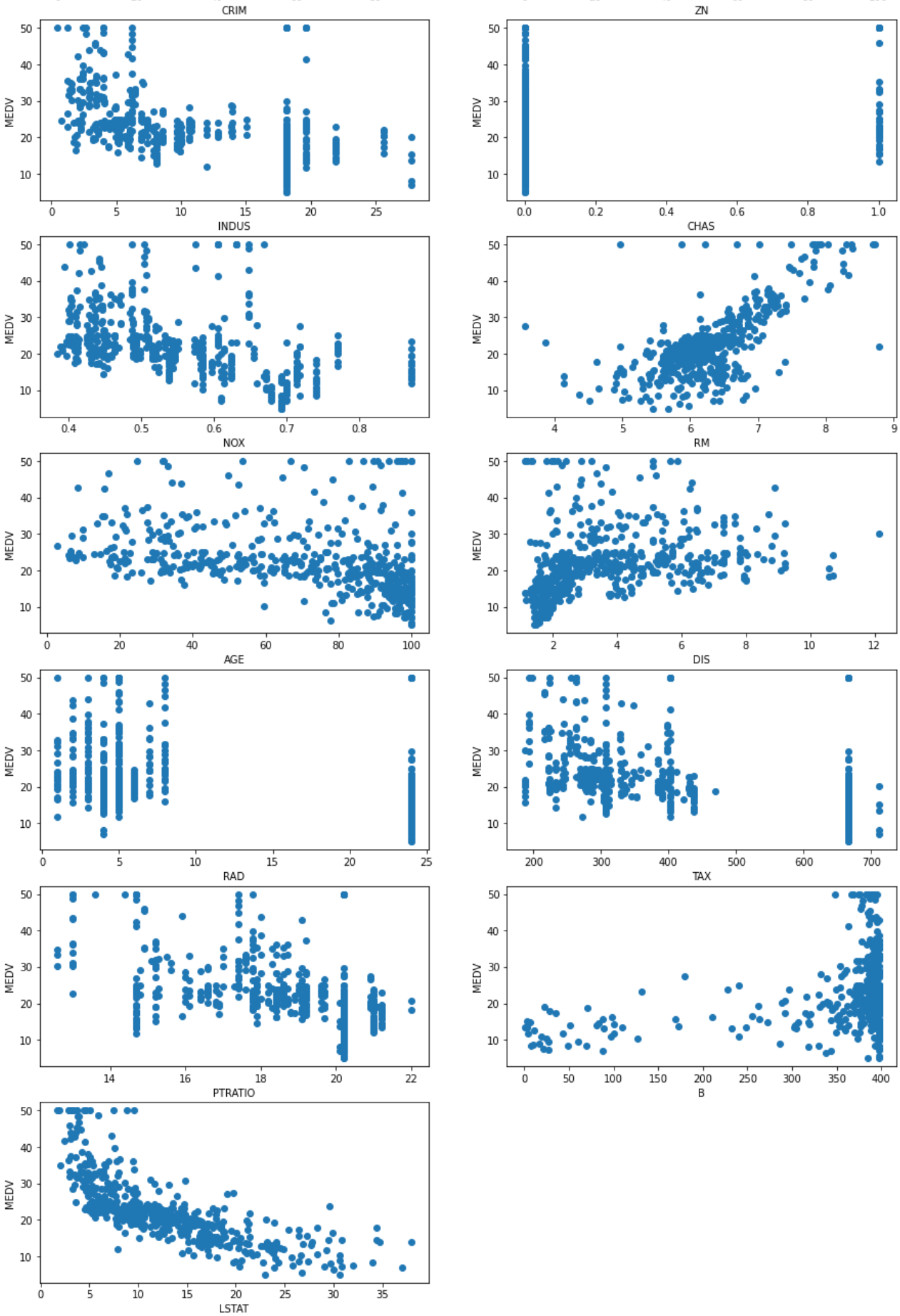


In [10]:

```
plt.figure(figsize=(15,50))
features=boston.columns[:-1]
target=boston.MEDV
for i, column in enumerate(features):
    plt.subplot(len(features),2,i+1)
    plt.scatter(x=boston[column], y=target, marker='o')
    plt.xlabel(column)
    plt.ylabel('MEDV')
```







In [11]:
```
from sklearn.preprocessing import StandardScaler
```

```
sc=StandardScaler()
# mean at 0 and std at 1
```

In [12]:
```
df=sc.fit_transform(boston)
```

In [13]:
```
df=pd.DataFrame(df)
df.head()
```

Out[13]:

	0	1	2	3	4	5	6	7	8	
0	-0.419782	0.284830	-1.287909	-0.272599	-0.144217	0.413672	-0.120013	0.140214	-0.982843	-0.666
1	-0.417339	-0.487722	-0.593381	-0.272599	-0.740262	0.194274	0.367166	0.557160	-0.867883	-0.987
2	-0.417342	-0.487722	-0.593381	-0.272599	-0.740262	1.282714	-0.265812	0.557160	-0.867883	-0.987
3	-0.416750	-0.487722	-1.306878	-0.272599	-0.835284	1.016303	-0.809889	1.077737	-0.752922	-1.106
4	-0.412482	-0.487722	-1.306878	-0.272599	-0.835284	1.228577	-0.511180	1.077737	-0.752922	-1.106

In [14]:
```
df.columns=boston.columns
```

In [15]:
```
df.head()
```

Out[15]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	T
0	-0.419782	0.284830	-1.287909	-0.272599	-0.144217	0.413672	-0.120013	0.140214	-0.982843	-0.666
1	-0.417339	-0.487722	-0.593381	-0.272599	-0.740262	0.194274	0.367166	0.557160	-0.867883	-0.987
2	-0.417342	-0.487722	-0.593381	-0.272599	-0.740262	1.282714	-0.265812	0.557160	-0.867883	-0.987
3	-0.416750	-0.487722	-1.306878	-0.272599	-0.835284	1.016303	-0.809889	1.077737	-0.752922	-1.106
4	-0.412482	-0.487722	-1.306878	-0.272599	-0.835284	1.228577	-0.511180	1.077737	-0.752922	-1.106

In [16]:
```
df.describe()
```

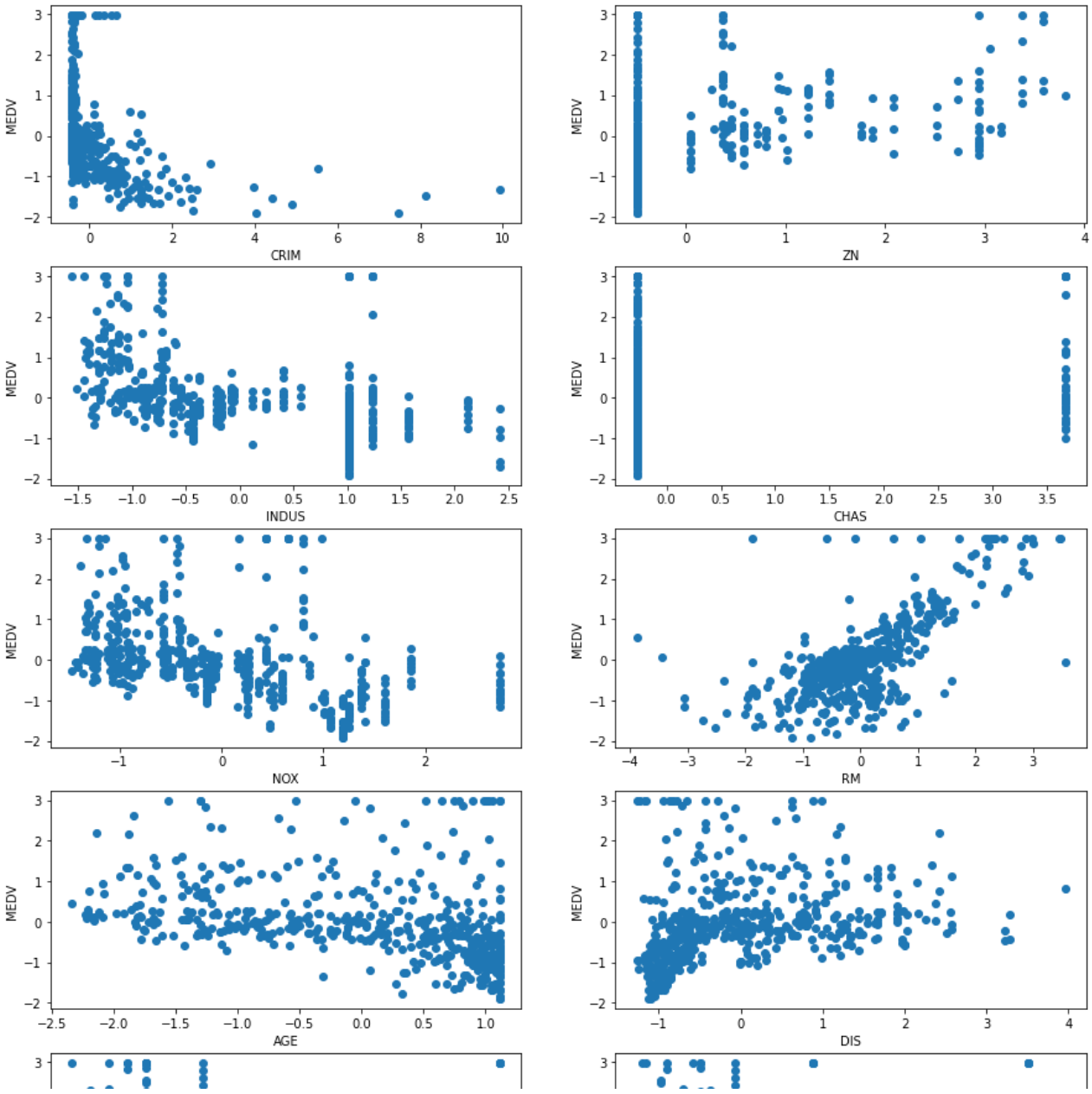
Out[16]:

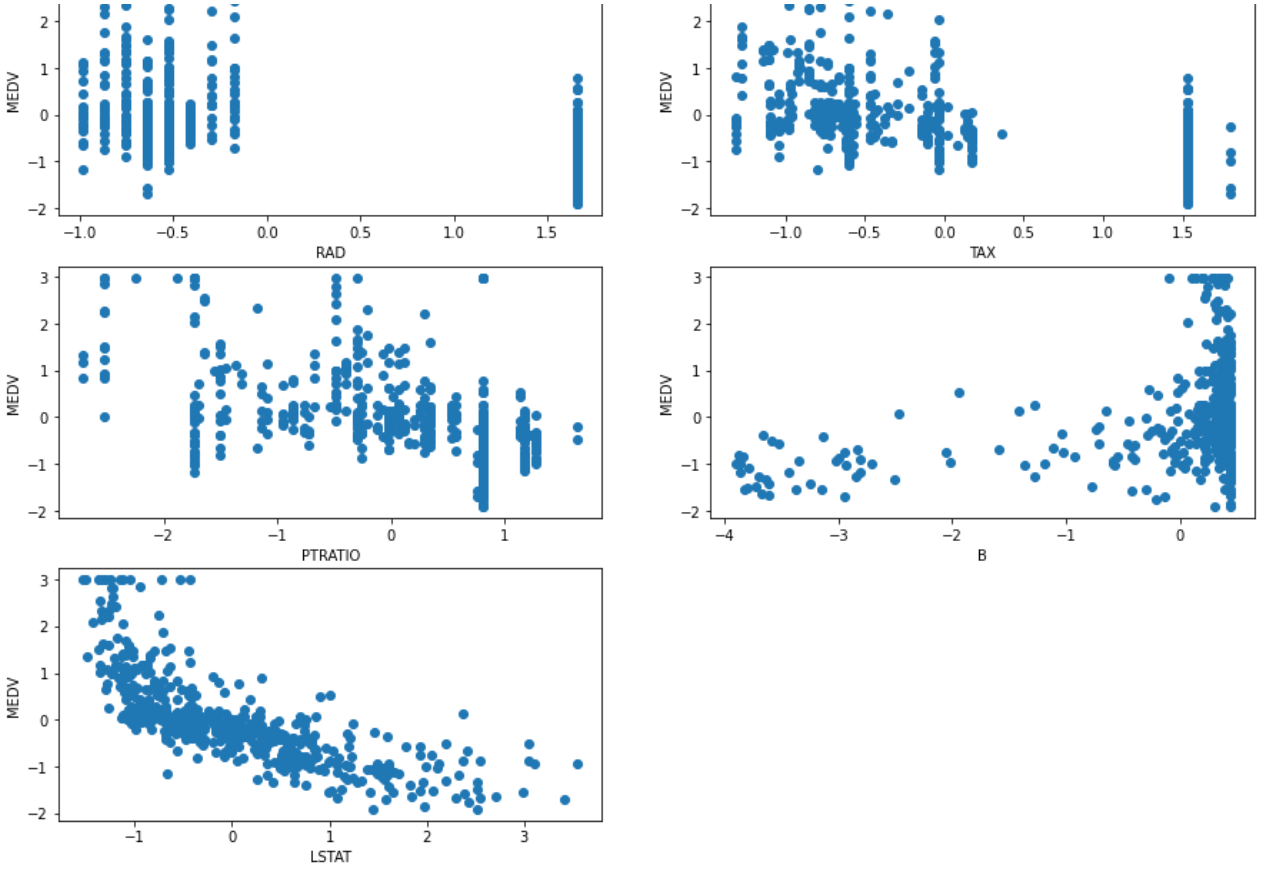
	CRIM	ZN	INDUS	CHAS	NOX	RM	
count	5.060000e+02	5.060000e+02	5.060000e+02	5.060000e+02	5.060000e+02	5.060000e+02	5.0600
mean	-8.513173e-17	3.306534e-16	2.804081e-16	-3.100287e-16	-8.071058e-16	-5.189086e-17	-2.650
std	1.000990e+00	1.000990e+00	1.000990e+00	1.000990e+00	1.000990e+00	1.000990e+00	1.0009
min	-4.197819e-01	-4.877224e-01	-1.557842e+00	-2.725986e-01	-1.465882e+00	-3.880249e+00	-2.3354
25%	-4.109696e-01	-4.877224e-01	-8.676906e-01	-2.725986e-01	-9.130288e-01	-5.686303e-01	-8.374

	CRIM	ZN	INDUS	CHAS	NOX	RM	
50%	-3.906665e-01	-4.877224e-01	-2.110985e-01	-2.725986e-01	-1.442174e-01	-1.084655e-01	3.173
75%	7.396560e-03	4.877224e-02	1.015999e+00	-2.725986e-01	5.986790e-01	4.827678e-01	9.067
max	9.933931e+00	3.804234e+00	2.422565e+00	3.668398e+00	2.732346e+00	3.555044e+00	1.1174

In [17]:

```
plt.figure(figsize=(15,50))
features=df.columns[:-1]
target=df.MEDV
for i, column in enumerate(features):
    plt.subplot(len(features),2,i+1)
    plt.scatter(x=df[column], y=target, marker='o')
    plt.xlabel(column)
    plt.ylabel('MEDV')
```





OBSERVATIONS

- **Variables LSTAT and RM have a hi correlation with the price of the house.**
- **INDUS-TAX, INDUS-DIS, INDUS-NOX, DIS-NOX , AGE-NOX, all these pairs have high correlation between them.**

In [18]:
```
X=boston[['LSTAT','RM']]
Y = boston[['MEDV']]
X.head()
```

Out[18]:

	LSTAT	RM
0	4.98	6.575
1	9.14	6.421
2	4.03	7.185
3	2.94	6.998
4	5.33	7.147

In [19]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.15, random_stat
# test_size=0.15 means 15% will be for test data
# random_state
print(X_train.shape)
print(X_test.shape)
```

```
print(Y_train.shape)
print(Y_test.shape)
```

```
(430, 2)
(76, 2)
(430, 1)
(76, 1)
```

Linear Regression

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.
Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

$$Y = \theta_1 + X\theta_0 + \epsilon$$

While training the model we are given :
x: input training data (univariate – one input variable(parameter))
y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.
θ1: intercept
θ2: coefficient of x
e:error term

Training Model

In [20]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

lin_model = LinearRegression()  # Make an instance of the model
lin_model.fit(X_train, Y_train)
```

Out[20]:
```
LinearRegression()
```

In [21]:
```
y_train_predict = lin_model.predict(X_train)
rmse = (np.sqrt(mean_squared_error(Y_train, y_train_predict)))

print("The model performance for training set")
print('RMSE is {}'.format(rmse))
print("\n")

# on testing set
y_test_predict = lin_model.predict(X_test)
```

```
rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))

print("The model performance for testing set")
print('RMSE is {}'.format(rmse))

print(lin_model.coef_.ravel())
print(lin_model.intercept_)
```

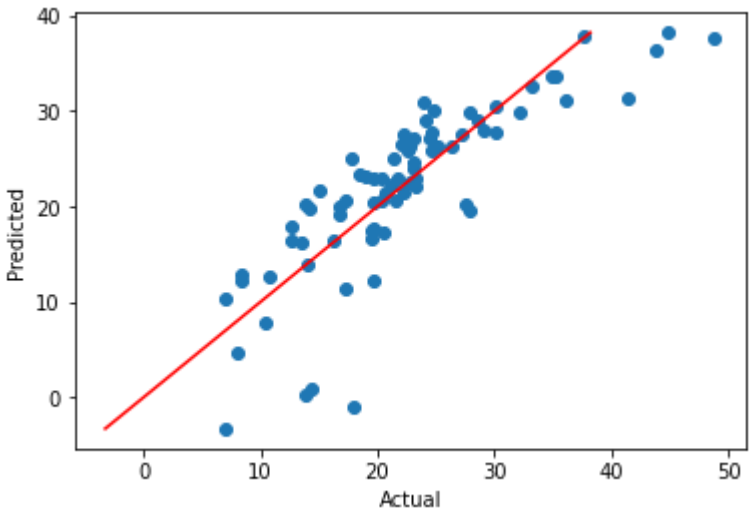
```
The model performance for training set
RMSE is 5.596970449422867


The model performance for testing set
RMSE is 5.178451251951529
[-0.70376468  4.88802288]
[0.68155642]
```

In [22]:

```
plt.scatter(Y_test, y_test_predict)

plt.plot([min(y_test_predict),max(y_test_predict)],[min(y_test_predict),max(y_test_pred
# Ploting a straight line y = x (red in color)
# For the 100% perfect fit, Predicted values will be same as Actual value
# That means for the curve below, y = x line represent 100% fit.

plt.xlabel('Actual')
plt.ylabel('Predicted')
```

Out[22]:

```
Text(0, 0.5, 'Predicted')
```



The red line shown represents y=x line (fit with 100% accuracy).

> **Now let's try to fit the linear regression model using all the variables**

In [23]:

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

In [24]:

```
X=df.drop(labels='MEDV', axis=1)
X.head()
```

Out[24]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD
0	-0.419782	0.284830	-1.287909	-0.272599	-0.144217	0.413672	-0.120013	0.140214	-0.982843
1	-0.417339	-0.487722	-0.593381	-0.272599	-0.740262	0.194274	0.367166	0.557160	-0.867883
2	-0.417342	-0.487722	-0.593381	-0.272599	-0.740262	1.282714	-0.265812	0.557160	-0.867883
3	-0.416750	-0.487722	-1.306878	-0.272599	-0.835284	1.016303	-0.809889	1.077737	-0.752922
4	-0.412482	-0.487722	-1.306878	-0.272599	-0.835284	1.228577	-0.511180	1.077737	-0.752922

In [25]:
```
X.shape
```

Out[25]:
```
(506, 13)
```

In [26]:
```
Y=df.MEDV
```

In [27]:
```
x_train,x_test,y_train,y_test=train_test_split(X, Y, test_size=0.3, random_state=2)
```

In [28]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

In [29]:
```
lin_model = LinearRegression()
lin_model.fit(x_train, y_train)
```

Out[29]:
```
LinearRegression()
```

In [30]:
```
y_train_predict = lin_model.predict(x_train)
rmse = (np.sqrt(mean_squared_error(y_train, y_train_predict)))

print("The model performance for training set")
print('RMSE is {}'.format(rmse))
print("\n")

# on testing set
y_test_predict = lin_model.predict(x_test)
rmse = (np.sqrt(mean_squared_error(y_test, y_test_predict)))

print("The model performance for testing set")
print('RMSE is {}'.format(rmse))

print(lin_model.coef_.ravel())
print(lin_model.intercept_)
```

```
The model performance for training set
RMSE is 0.5112786395135811


The model performance for testing set
RMSE is 0.5224064330994048
```

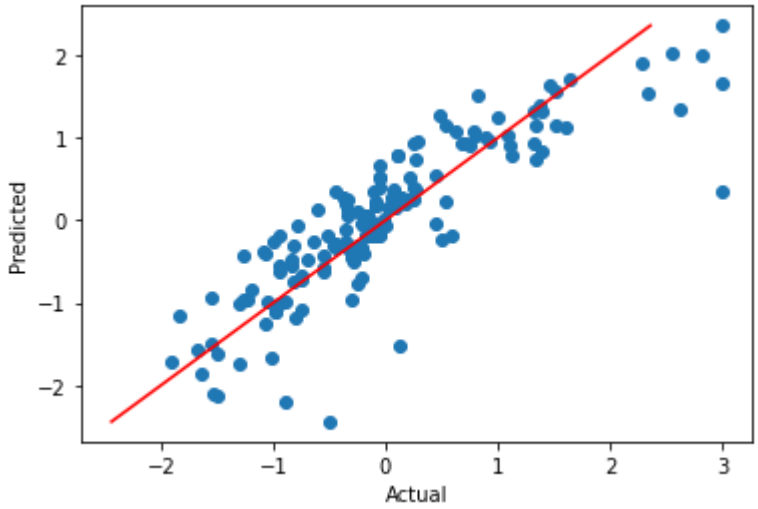
```
[-0.08725644  0.07895821 -0.01355751  0.08825089 -0.1903045   0.2674622
  0.05808119 -0.28974664  0.30499803 -0.20057974 -0.25649371  0.12447486
 -0.47178377]
0.0076784214248864485
```

In [31]:

```
plt.scatter(y_test, y_test_predict)

plt.plot([min(y_test_predict),max(y_test_predict)],[min(y_test_predict),max(y_test_pred
# Ploting a straight line y = x (red in color)
# For the 100% perfect fit, Predicted values will be same as Actual value
# That means for the curve below, y = x line represent 100% fit.

plt.xlabel('Actual')
plt.ylabel('Predicted')
```

Out[31]:

```
Text(0, 0.5, 'Predicted')
```



In [ ]:

Assignment - 5 _ Data Analytics 2 -Logestic Regression

Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

1. Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

Loading Social_Network_Ads.csv dataset

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

df = pd.read_csv('Social_Network_Ads.csv')
df.head()
```

Out[5]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

Purchased is the binary data thus it is called the Dependable variable and Age, EstimatedSalary are independent variables

In [6]:
```
df['Purchased'].value_counts()
```

Out[6]:
```
0    257
1    143
Name: Purchased, dtype: int64
```

In [7]:
```
df = df.drop('User ID', axis = 1)
df.head()
```

Out[7]:

	Gender	Age	EstimatedSalary	Purchased
0	Male	19	19000	0
1	Male	35	20000	0
2	Female	26	43000	0
3	Female	27	57000	0
4	Male	19	76000	0

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Gender           400 non-null    object
 1   Age              400 non-null    int64 
 2   EstimatedSalary  400 non-null    int64 
 3   Purchased        400 non-null    int64 
dtypes: int64(3), object(1)
memory usage: 12.6+ KB
```

In [9]:

```
from sklearn.preprocessing import OneHotEncoder
df_onehot = pd.get_dummies(df, columns=['Gender'], prefix = ['Gender'])
df_onehot.head(10)
```

Out[9]:

	Age	EstimatedSalary	Purchased	Gender_Female	Gender_Male
0	19	19000	0	0	1
1	35	20000	0	0	1
2	26	43000	0	1	0
3	27	57000	0	1	0
4	19	76000	0	0	1
5	27	58000	0	0	1
6	27	84000	0	1	0
7	32	150000	1	1	0
8	25	33000	0	0	1
9	35	65000	0	1	0

Data Transformation of Gender into Male and Female gender using 0 and 1

In [10]:

```
from sklearn.preprocessing import MinMaxScaler

scaler=MinMaxScaler(feature_range=(0,1))
```

```
df_onehot["EstimatedSalary"]=scaler.fit_transform(df_onehot[["EstimatedSalary"]])
Column_loc = ['Age','EstimatedSalary','Gender_Female','Gender_Male','Purchased']
df_onehot = df_onehot[Column_loc]
df_onehot.head(5)
```

Out[10]:

	Age	EstimatedSalary	Gender_Female	Gender_Male	Purchased
0	19	0.029630	0	1	0
1	35	0.037037	0	1	0
2	26	0.207407	1	0	0
3	27	0.311111	1	0	0
4	19	0.451852	0	1	0

Initializaing dependent and independent variables

In [11]:

```
X = df_onehot.iloc[:,:-1].values
print(X)
y = df_onehot.iloc[:,-1].values
```

```
[[1.90000000e+01 2.96296296e-02 0.00000000e+00 1.00000000e+00]
 [3.50000000e+01 3.70370370e-02 0.00000000e+00 1.00000000e+00]
 [2.60000000e+01 2.07407407e-01 1.00000000e+00 0.00000000e+00]
 ...
 [5.00000000e+01 3.70370370e-02 1.00000000e+00 0.00000000e+00]
 [3.60000000e+01 1.33333333e-01 0.00000000e+00 1.00000000e+00]
 [4.90000000e+01 1.55555556e-01 1.00000000e+00 0.00000000e+00]]
```

Applying train, test, split operatoin on dataset

In [12]:

```
from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size= 0.25, random_state
print(train_X.shape,train_y.shape,test_X.shape,test_y.shape)
```

```
(300, 4) (300,) (100, 4) (100,)
```

Logistic Regression

Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical.
For example,

- To predict whether an email is spam (1) or (0)
- Whether the tumor is malignant (1) or not (0)

Applying logistic regression on trained data

In [13]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(train_X,train_y)    #fitting trainable data
```

Out[13]:
```
LogisticRegression()
```

In [14]:
```
#pridicting Y value using test_X
predict_y = lr.predict(test_X)
predict_y
```

Out[14]:
```
array([0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1], dtype=int64)
```

Calculating accuracy

In [15]:
```
from sklearn.metrics import accuracy_score
accuracy_score(test_y, predict_y)
```

Out[15]:
```
0.86
```

Calulating - Precision and Recall

In [16]:
```
from sklearn.metrics import confusion_matrix, precision_score, recall_score
print(precision_score(test_y, predict_y))
print(recall_score(test_y, predict_y))
```

```
0.8076923076923077
0.7
```

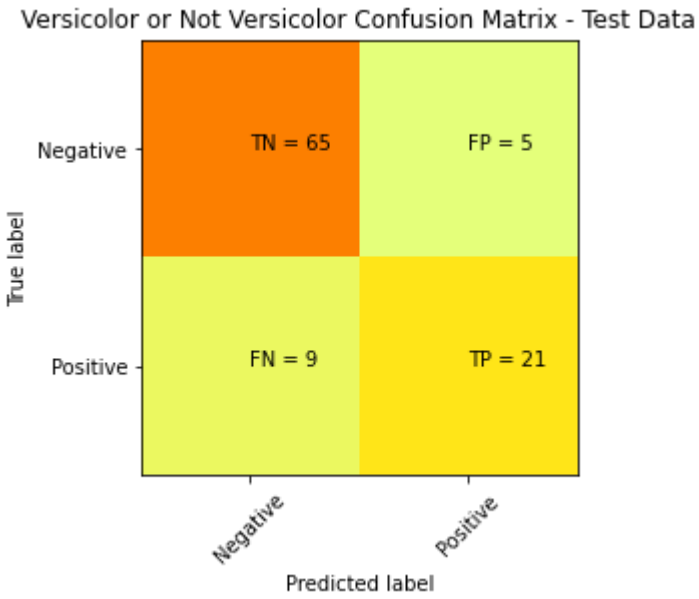
Confusion Matrix

In [17]:
```
print(confusion_matrix(test_y, predict_y))
tp,fp, fn, tn = confusion_matrix(test_y, predict_y).ravel()
```

```
[[65  5]
 [ 9 21]]
```

In [24]:
```
plt.clf()
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative','Positive']
plt.title('Versicolor or Not Versicolor Confusion Matrix - Test Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN','FP'], ['FN', 'TP']]
for i in range(2):
    for j in range(2):
```

```
        plt.text(j,i, str(s[i][j])+" = "+str(cm[i][j]))
plt.show()
```



TP FP FN TN

In [19]:
```
print(f'Correctly Predicted made Purchase {tp}')
print(f'Falsely Predicted made Purchase {fp}')
print(f'Falsely Predicted made did NOT made Purchase {fn}')
print(f'Correctly Predicted made did NOT made Purchase {tn}')
```

```
Correctly Predicted made Purchase 65
Falsely Predicted made Purchase 5
Falsely Predicted made did NOT made Purchase 9
Correctly Predicted made did NOT made Purchase 21
```

In [20]:
```
Result_Test_X = pd.DataFrame(test_X, columns =['Age','EstimatedSalary','Gender_Female',
Result_Predict_y = pd.DataFrame(predict_y, columns= ['Pred_Purchase'])
pd.concat([Result_Test_X, Result_Predict_y], axis=1)
```

Out[20]:

	Age	EstimatedSalary	Gender_Female	Gender_Male	Pred_Purchase
0	33.0	0.333333	1.0	0.0	0
1	27.0	0.511111	1.0	0.0	0
2	25.0	0.533333	0.0	1.0	0
3	38.0	0.259259	1.0	0.0	0
4	27.0	0.540741	0.0	1.0	0
...	...	...	...	...	...
95	45.0	0.222222	1.0	0.0	0
96	32.0	0.777778	0.0	1.0	0
97	21.0	0.422222	0.0	1.0	0

	Age	EstimatedSalary	Gender_Female	Gender_Male	Pred_Purchase
98	45.0	0.125926	0.0	1.0	0
99	48.0	0.911111	1.0	0.0	1

100 rows × 5 columns

In [21]:
```
Result_test_y = pd.DataFrame(test_y, columns= ['Purchased'])
Result = pd.concat([Result_Test_X, Result_test_y, Result_Predict_y], axis=1)
Result
```

Out[21]:

	Age	EstimatedSalary	Gender_Female	Gender_Male	Purchased	Pred_Purchase
0	33.0	0.333333	1.0	0.0	0	0
1	27.0	0.511111	1.0	0.0	0	0
2	25.0	0.533333	0.0	1.0	0	0
3	38.0	0.259259	1.0	0.0	0	0
4	27.0	0.540741	0.0	1.0	0	0
...	...	...	...	...	...	...
95	45.0	0.222222	1.0	0.0	1	0
96	32.0	0.777778	0.0	1.0	1	0
97	21.0	0.422222	0.0	1.0	0	0
98	45.0	0.125926	0.0	1.0	1	0
99	48.0	0.911111	1.0	0.0	1	1

100 rows × 6 columns

In [22]:
```
# Comparing test_y and predicted_y values and saving
Result.to_csv('./Final_Prediction_test_values.csv')
```

In [ ]:
```

```

Assignment -6 -Data Analytics 3 - Naive Bayes Classification

Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

1. Implement Simple Naive 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.

In [2]:
```
# Description

# It includes three iris species with 50 samples each as well as some properties about
#One flower species is linearly separable from the other two, but the other two are not

# The columns in this dataset are:

# Id
# SepalLengthCm
# SepalWidthCm
# PetalLengthCm
# PetalWidthCm
# Species
```

In [3]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns

iris = pd.read_csv('Iris.csv')
iris.head()
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [4]:
```
iris['Species'].unique()
```

Out[4]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [5]:

```
iris.describe(include='all')
```

Out[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	150.000000	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	NaN	Iris-versicolor
freq	NaN	NaN	NaN	NaN	NaN	50
mean	75.500000	5.843333	3.054000	3.758667	1.198667	NaN
std	43.445368	0.828066	0.433594	1.764420	0.763161	NaN
min	1.000000	4.300000	2.000000	1.000000	0.100000	NaN
25%	38.250000	5.100000	2.800000	1.600000	0.300000	NaN
50%	75.500000	5.800000	3.000000	4.350000	1.300000	NaN
75%	112.750000	6.400000	3.300000	5.100000	1.800000	NaN
max	150.000000	7.900000	4.400000	6.900000	2.500000	NaN

In [6]:

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [7]:

```
#Removing unecessary columns
iris.drop(columns="Id",inplace=True)
```

In [8]:

```
#Checking for null values
iris.isnull().sum()
```

Out[8]:

```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Correlations

In [9]:

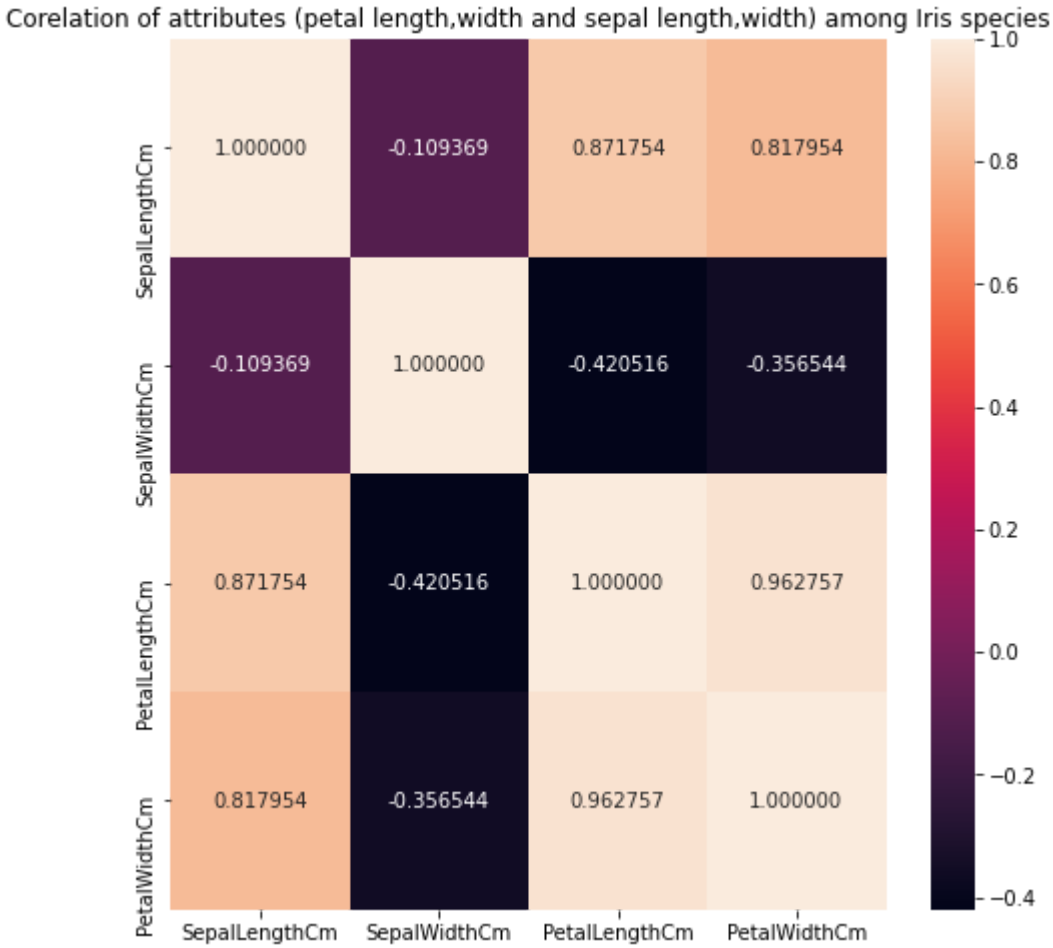
```
iris.corr()
```

Out[9]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

In [10]:

```
plt.subplots(figsize = (8,8))
sns.heatmap(iris.corr(),annot=True,fmt="f").set_title("Corelation of attributes (petal
plt.show()
```



Splitting The Data into Training And Testing Dataset

In [11]:

```
X=iris.iloc[:,0:4].values
y=iris.iloc[:,4].values

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

Naive Bayes Classifiers

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.It is mainly used in text classification that includes a high-dimensional training dataset.Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.It is a **probabilistic classifier**, which means it predicts on the basis of the probability of an object.Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Bayes' Theorem:

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability. The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{(P(B|A)P(A))}{P(B)}$$

Where,

- **P(A|B) is Posterior probability**: Probability of hypothesis A on the observed event B.
- **P(B|A) is Likelihood probability**: Probability of the evidence given that the probability of a hypothesis is true.
- **P(A) is Prior Probability**: Probability of hypothesis before observing the evidence.
- **P(B) is Marginal Probability**: Probability of Evidence.

In [12]:
```
#Metrics
from sklearn.metrics import make_scorer, accuracy_score,precision_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score ,precision_score,recall_score,f1_score

#Model Select
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
```

In [13]:
```
#Train and Test split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=0)
```

In [14]:
```
gaussian = GaussianNB()
gaussian.fit(X_train, y_train)
Y_pred = gaussian.predict(X_test)
accuracy_nb=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_gaussian = round(gaussian.score(X_train, y_train) * 100, 2)
```

Confusion Matrix

In [20]:
```
cm = confusion_matrix(y_test, Y_pred)
print('Confusion matrix for Naive Bayes\n',cm)
```

```
Confusion matrix for Naive Bayes
 [[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]
```

In [21]:
```
FP = cm.sum(axis=0) - np.diag(cm)
FN = cm.sum(axis=1) - np.diag(cm)
TP = np.diag(cm)
TN = cm.sum() - (FP + FN + TP)
FP = FP.astype(float)
FN = FN.astype(float)
TP = TP.astype(float)
TN = TN.astype(float)

print("['SetosaTP','VersicolorTP','VirginicaTP']",TP)
print("['SetosaFP','VersicolorFP','VirginicaFP']",FP)
print("['SetosaFN','VersicolorFN','VirginicaFN']",FN)
print("['SetosaTN','VersicolorTN','VirginicaTN']",TN)
```

```
['SetosaTP','VersicolorTP','VirginicaTP'] [16. 18. 11.]
['SetosaFP','VersicolorFP','VirginicaFP'] [0. 0. 0.]
['SetosaFN','VersicolorFN','VirginicaFN'] [0. 0. 0.]
['SetosaTN','VersicolorTN','VirginicaTN'] [29. 27. 34.]
```

Accuracy

In [22]:
```
accuracy = accuracy_score(y_test,Y_pred)
print('accuracy_Naive Bayes: %.3f' %accuracy)
```

```
accuracy_Naive Bayes: 1.000
```

In [23]:
```
precision =precision_score(y_test, Y_pred,average='micro')
print('precision_Naive Bayes: %.3f' %precision)
```

```
precision_Naive Bayes: 1.000
```

Recall

In [24]:
```
recall =  recall_score(y_test, Y_pred,average='micro')
print('recall_Naive Bayes: %.3f' %recall)
```

```
recall_Naive Bayes: 1.000
```

Assignment 7 - Text Analytics

Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

Text Analytics

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

In [1]:
```
import pandas as pd
```

In [2]:
```
text = '''It was a Thursday, but it felt like a Monday to John. And John loved Mondays.

I should probably get another latte. I've just been sitting here with this empty cup. B

John was always impatient on the weekends; he missed the formal structure of the busine

Jesus, I've written another loser. '''
```

Tokenization of text

In [3]:
```
text_split = text.split()
```

In [4]:
```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\asus\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\asus\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\asus\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[4]:
```
True
```

In [5]:
```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
stop_words = stopwords.words('english')
```

In [6]:
```
#stop_words
```

In [7]:
```
tokenized = sent_tokenize(text)
for i in tokenized:

    wordsList = nltk.word_tokenize(i)

    # removing stop words from wordList
    wordsList = [w for w in wordsList if not w in stop_words]

    # Using a Tagger. Which is part-of-speech
    # tagger or POS-tagger.
    tagged = nltk.pos_tag(wordsList)

    print(tagged)
```

```
[('It', 'PRP'), ('Thursday', 'NNP'), (',', ','), ('felt', 'VBD'), ('like', 'IN'), ('Mond
ay', 'NNP'), ('John', 'NNP'), ('.', '.')]
[('And', 'CC'), ('John', 'NNP'), ('loved', 'VBD'), ('Mondays', 'NNP'), ('.', '.')]
[('He', 'PRP'), ('thrived', 'VBD'), ('work', 'NN'), ('.', '.')]
[('He', 'PRP'), ('dismissed', 'VBD'), ('old', 'JJ'), ('cliché', 'NN'), ('dreading', 'VB
G'), ('Monday', 'NNP'), ('mornings', 'NNS'), ('refused', 'VBD'), ('engage', 'JJ'), ('wat
er-cooler', 'JJ'), ('complaints', 'NNS'), ('“', 'JJ'), ('grind', 'VBP'), ('”', 'JJ'),
('empty', 'JJ'), ('conversations', 'NNS'), ('included', 'VBD'), ('familiar', 'JJ'), ('pa
rry', 'NN'), ('“', 'NNP'), ('How', 'NNP'), ('weekend', 'NN'), ('?', '.'), ('”', 'JJ'),
('“', 'NNP'), ('Too', 'NNP'), ('short', 'JJ'), ('!', '.'), ('”', 'NN'), ('.', '.')]
[('Yes', 'UH'), (',', ','), ('John', 'NNP'), ('liked', 'VBD'), ('work', 'NN'), ('unasham
ed', 'NN'), ('.', '.')]
[('I', 'PRP'), ('probably', 'RB'), ('get', 'VB'), ('another', 'DT'), ('latte', 'NN'),
('.', '.')]
[('I', 'PRP'), ('’', 'VBP'), ('sitting', 'VBG'), ('empty', 'JJ'), ('cup', 'NN'), ('.',
'.')]
[('But', 'CC'), ('I', 'PRP'), ('’', 'VBP'), ('start', 'JJ'), ('get', 'VB'), ('jittery',
'NN'), ('.', '.')]
[('I', 'PRP'), ('’', 'VBP'), ('get', 'VB'), ('decaf', 'NN'), ('.', '.')]
[('No', 'DT'), (',', ','), ('’', 'FW'), ('stupid', 'JJ'), (',', ','), ('feels', 'JJ'),
('stupid', 'JJ'), ('pay', 'NN'), ('decaf', 'NN'), ('.', '.')]
[('I', 'PRP'), ('’', 'VBP'), ('justify', 'NN'), ('.', '.')]
[('John', 'NNP'), ('always', 'RB'), ('impatient', 'JJ'), ('weekends', 'NNS'), (';',
':'), ('missed', 'VBN'), ('formal', 'JJ'), ('structure', 'NN'), ('business', 'NN'), ('we
ek', 'NN'), ('.', '.')]
[('When', 'WRB'), ('younger', 'JJR'), ('used', 'VBD'), ('stay', 'NN'), ('late', 'JJ'),
('school', 'NN'), ('Fridays', 'NNP'), ('come', 'VBP'), ('early', 'JJ'), ('Mondays', 'NN
P'), (',', ','), ('pattern', 'NN'), ('mother', 'NN'), ('referred', 'VBD'), ('equal', 'J
J'), ('parts', 'NNS'), ('admiration', 'NN'), ('disdain', 'VBP'), ('“', 'JJ'), ('studyin
g', 'VBG'), ('overtime.', 'JJ'), ('”', 'NNP'), ('Jesus', 'NNP'), (',', ','), ('I', 'PR
P'), ('’', 'VBP'), ('written', 'VBN'), ('another', 'DT'), ('loser', 'NN'), ('.', '.')]
```

In [8]:
```
tokenized
```

Out[8]:
```
['It was a Thursday, but it felt like a Monday to John.',
 'And John loved Mondays.',
 'He thrived at work.',
 'He dismissed the old cliché of dreading Monday mornings and refused to engage in water
-cooler complaints about “the grind” and empty conversations that included the familiar
parry “How was your weekend?” “Too short!”.',
```

```
 'Yes, John liked his work and was unashamed.',
 'I should probably get another latte.',
 'I've just been sitting here with this empty cup.',
 'But then I'll start to get jittery.',
 'I'll get a decaf.',
 'No, that's stupid, it feels stupid to pay for a decaf.',
 'I can't justify that.',
 'John was always impatient on the weekends; he missed the formal structure of the busin
ess week.',
 'When he was younger he used to stay late after school on Fridays and come in early on
Mondays, a pattern his mother referred to with equal parts admiration and disdain as "st
udying overtime."\n\nJesus, I've written another loser.']
```

Stemming and Lemmatization

In [9]:
```
from nltk.stem.porter import PorterStemmer
```

In [10]:
```
porter_stemmer = PorterStemmer()
```

In [11]:
```
nltk_token = nltk.word_tokenize(text)
```

In [12]:
```
for w in nltk_token:
  print("Actual : %s , Stem: %s" %(w, porter_stemmer.stem(w)))
```

```
Actual : It , Stem: it
Actual : was , Stem: wa
Actual : a , Stem: a
Actual : Thursday , Stem: thursday
Actual : , , Stem: ,
Actual : but , Stem: but
Actual : it , Stem: it
Actual : felt , Stem: felt
Actual : like , Stem: like
Actual : a , Stem: a
Actual : Monday , Stem: monday
Actual : to , Stem: to
Actual : John , Stem: john
Actual : . , Stem: .
Actual : And , Stem: and
Actual : John , Stem: john
Actual : loved , Stem: love
Actual : Mondays , Stem: monday
Actual : . , Stem: .
Actual : He , Stem: he
Actual : thrived , Stem: thrive
Actual : at , Stem: at
Actual : work , Stem: work
Actual : . , Stem: .
Actual : He , Stem: he
Actual : dismissed , Stem: dismiss
Actual : the , Stem: the
Actual : old , Stem: old
Actual : cliché , Stem: cliché
Actual : of , Stem: of
Actual : dreading , Stem: dread
```

```
Actual : Monday , Stem: monday
Actual : mornings , Stem: morn
Actual : and , Stem: and
Actual : refused , Stem: refus
Actual : to , Stem: to
Actual : engage , Stem: engag
Actual : in , Stem: in
Actual : water-cooler , Stem: water-cool
Actual : complaints , Stem: complaint
Actual : about , Stem: about
Actual : “ , Stem: “
Actual : the , Stem: the
Actual : grind , Stem: grind
Actual : ” , Stem: ”
Actual : and , Stem: and
Actual : empty , Stem: empti
Actual : conversations , Stem: convers
Actual : that , Stem: that
Actual : included , Stem: includ
Actual : the , Stem: the
Actual : familiar , Stem: familiar
Actual : parry , Stem: parri
Actual : “ , Stem: “
Actual : How , Stem: how
Actual : was , Stem: wa
Actual : your , Stem: your
Actual : weekend , Stem: weekend
Actual : ? , Stem: ?
Actual : ” , Stem: ”
Actual : “ , Stem: “
Actual : Too , Stem: too
Actual : short , Stem: short
Actual : ! , Stem: !
Actual : ” , Stem: ”
Actual : . , Stem: .
Actual : Yes , Stem: ye
Actual : , , Stem: ,
Actual : John , Stem: john
Actual : liked , Stem: like
Actual : his , Stem: hi
Actual : work , Stem: work
Actual : and , Stem: and
Actual : was , Stem: wa
Actual : unashamed , Stem: unasham
Actual : . , Stem: .
Actual : I , Stem: i
Actual : should , Stem: should
Actual : probably , Stem: probabl
Actual : get , Stem: get
Actual : another , Stem: anoth
Actual : latte , Stem: latt
Actual : . , Stem: .
Actual : I , Stem: i
Actual : ’ , Stem: ’
Actual : ve , Stem: ve
Actual : just , Stem: just
Actual : been , Stem: been
Actual : sitting , Stem: sit
Actual : here , Stem: here
Actual : with , Stem: with
```

```
Actual : this , Stem: thi
Actual : empty , Stem: empti
Actual : cup , Stem: cup
Actual : . , Stem: .
Actual : But , Stem: but
Actual : then , Stem: then
Actual : I , Stem: i
Actual : ’ , Stem: ’
Actual : ll , Stem: ll
Actual : start , Stem: start
Actual : to , Stem: to
Actual : get , Stem: get
Actual : jittery , Stem: jitteri
Actual : . , Stem: .
Actual : I , Stem: i
Actual : ’ , Stem: ’
Actual : ll , Stem: ll
Actual : get , Stem: get
Actual : a , Stem: a
Actual : decaf , Stem: decaf
Actual : . , Stem: .
Actual : No , Stem: no
Actual : , , Stem: ,
Actual : that , Stem: that
Actual : ’ , Stem: ’
Actual : s , Stem: s
Actual : stupid , Stem: stupid
Actual : , , Stem: ,
Actual : it , Stem: it
Actual : feels , Stem: feel
Actual : stupid , Stem: stupid
Actual : to , Stem: to
Actual : pay , Stem: pay
Actual : for , Stem: for
Actual : a , Stem: a
Actual : decaf , Stem: decaf
Actual : . , Stem: .
Actual : I , Stem: i
Actual : can , Stem: can
Actual : ’ , Stem: ’
Actual : t , Stem: t
Actual : justify , Stem: justifi
Actual : that , Stem: that
Actual : . , Stem: .
Actual : John , Stem: john
Actual : was , Stem: wa
Actual : always , Stem: alway
Actual : impatient , Stem: impati
Actual : on , Stem: on
Actual : the , Stem: the
Actual : weekends , Stem: weekend
Actual : ; , Stem: ;
Actual : he , Stem: he
Actual : missed , Stem: miss
Actual : the , Stem: the
Actual : formal , Stem: formal
Actual : structure , Stem: structur
Actual : of , Stem: of
Actual : the , Stem: the
Actual : business , Stem: busi
```

```
Actual : week , Stem: week
Actual : . , Stem: .
Actual : When , Stem: when
Actual : he , Stem: he
Actual : was , Stem: wa
Actual : younger , Stem: younger
Actual : he , Stem: he
Actual : used , Stem: use
Actual : to , Stem: to
Actual : stay , Stem: stay
Actual : late , Stem: late
Actual : after , Stem: after
Actual : school , Stem: school
Actual : on , Stem: on
Actual : Fridays , Stem: friday
Actual : and , Stem: and
Actual : come , Stem: come
Actual : in , Stem: in
Actual : early , Stem: earli
Actual : on , Stem: on
Actual : Mondays , Stem: monday
Actual : , , Stem: ,
Actual : a , Stem: a
Actual : pattern , Stem: pattern
Actual : his , Stem: hi
Actual : mother , Stem: mother
Actual : referred , Stem: refer
Actual : to , Stem: to
Actual : with , Stem: with
Actual : equal , Stem: equal
Actual : parts , Stem: part
Actual : admiration , Stem: admir
Actual : and , Stem: and
Actual : disdain , Stem: disdain
Actual : as , Stem: as
Actual : “ , Stem: “
Actual : studying , Stem: studi
Actual : overtime. , Stem: overtime.
Actual : ” , Stem: ”
Actual : Jesus , Stem: jesu
Actual : , , Stem: ,
Actual : I , Stem: i
Actual : ’ , Stem: ’
Actual : ve , Stem: ve
Actual : written , Stem: written
Actual : another , Stem: anoth
Actual : loser , Stem: loser
Actual : . , Stem: .
```

Lemmatization

In [13]:
```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
```

In [14]:
```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
```

```
[nltk_data]     C:\Users\ORIONORIGINAL\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\wordnet.zip.
```

Out[14]:
```
True
```

In [15]:
```
for w in nltk_token:
  print("Actual : %s , Lemme: %s" %(w, wordnet_lemmatizer.lemmatize(w)))
```

```
Actual : It , Lemme: It
Actual : was , Lemme: wa
Actual : a , Lemme: a
Actual : Thursday , Lemme: Thursday
Actual : , , Lemme: ,
Actual : but , Lemme: but
Actual : it , Lemme: it
Actual : felt , Lemme: felt
Actual : like , Lemme: like
Actual : a , Lemme: a
Actual : Monday , Lemme: Monday
Actual : to , Lemme: to
Actual : John , Lemme: John
Actual : . , Lemme: .
Actual : And , Lemme: And
Actual : John , Lemme: John
Actual : loved , Lemme: loved
Actual : Mondays , Lemme: Mondays
Actual : . , Lemme: .
Actual : He , Lemme: He
Actual : thrived , Lemme: thrived
Actual : at , Lemme: at
Actual : work , Lemme: work
Actual : . , Lemme: .
Actual : He , Lemme: He
Actual : dismissed , Lemme: dismissed
Actual : the , Lemme: the
Actual : old , Lemme: old
Actual : cliché , Lemme: cliché
Actual : of , Lemme: of
Actual : dreading , Lemme: dreading
Actual : Monday , Lemme: Monday
Actual : mornings , Lemme: morning
Actual : and , Lemme: and
Actual : refused , Lemme: refused
Actual : to , Lemme: to
Actual : engage , Lemme: engage
Actual : in , Lemme: in
Actual : water-cooler , Lemme: water-cooler
Actual : complaints , Lemme: complaint
Actual : about , Lemme: about
Actual : “ , Lemme: “
Actual : the , Lemme: the
Actual : grind , Lemme: grind
Actual : ” , Lemme: ”
Actual : and , Lemme: and
Actual : empty , Lemme: empty
Actual : conversations , Lemme: conversation
Actual : that , Lemme: that
Actual : included , Lemme: included
Actual : the , Lemme: the
```

```
Actual : familiar , Lemme: familiar
Actual : parry , Lemme: parry
Actual : “ , Lemme: “
Actual : How , Lemme: How
Actual : was , Lemme: wa
Actual : your , Lemme: your
Actual : weekend , Lemme: weekend
Actual : ? , Lemme: ?
Actual : ” , Lemme: ”
Actual : “ , Lemme: “
Actual : Too , Lemme: Too
Actual : short , Lemme: short
Actual : ! , Lemme: !
Actual : ” , Lemme: ”
Actual : . , Lemme: .
Actual : Yes , Lemme: Yes
Actual : , , Lemme: ,
Actual : John , Lemme: John
Actual : liked , Lemme: liked
Actual : his , Lemme: his
Actual : work , Lemme: work
Actual : and , Lemme: and
Actual : was , Lemme: wa
Actual : unashamed , Lemme: unashamed
Actual : . , Lemme: .
Actual : I , Lemme: I
Actual : should , Lemme: should
Actual : probably , Lemme: probably
Actual : get , Lemme: get
Actual : another , Lemme: another
Actual : latte , Lemme: latte
Actual : . , Lemme: .
Actual : I , Lemme: I
Actual : ’ , Lemme: ’
Actual : ve , Lemme: ve
Actual : just , Lemme: just
Actual : been , Lemme: been
Actual : sitting , Lemme: sitting
Actual : here , Lemme: here
Actual : with , Lemme: with
Actual : this , Lemme: this
Actual : empty , Lemme: empty
Actual : cup , Lemme: cup
Actual : . , Lemme: .
Actual : But , Lemme: But
Actual : then , Lemme: then
Actual : I , Lemme: I
Actual : ’ , Lemme: ’
Actual : ll , Lemme: ll
Actual : start , Lemme: start
Actual : to , Lemme: to
Actual : get , Lemme: get
Actual : jittery , Lemme: jittery
Actual : . , Lemme: .
Actual : I , Lemme: I
Actual : ’ , Lemme: ’
Actual : ll , Lemme: ll
Actual : get , Lemme: get
Actual : a , Lemme: a
Actual : decaf , Lemme: decaf
```

```
Actual : . , Lemme: .
Actual : No , Lemme: No
Actual : , , Lemme: ,
Actual : that , Lemme: that
Actual : ’ , Lemme: ’
Actual : s , Lemme: s
Actual : stupid , Lemme: stupid
Actual : , , Lemme: ,
Actual : it , Lemme: it
Actual : feels , Lemme: feel
Actual : stupid , Lemme: stupid
Actual : to , Lemme: to
Actual : pay , Lemme: pay
Actual : for , Lemme: for
Actual : a , Lemme: a
Actual : decaf , Lemme: decaf
Actual : . , Lemme: .
Actual : I , Lemme: I
Actual : can , Lemme: can
Actual : ’ , Lemme: ’
Actual : t , Lemme: t
Actual : justify , Lemme: justify
Actual : that , Lemme: that
Actual : . , Lemme: .
Actual : John , Lemme: John
Actual : was , Lemme: wa
Actual : always , Lemme: always
Actual : impatient , Lemme: impatient
Actual : on , Lemme: on
Actual : the , Lemme: the
Actual : weekends , Lemme: weekend
Actual : ; , Lemme: ;
Actual : he , Lemme: he
Actual : missed , Lemme: missed
Actual : the , Lemme: the
Actual : formal , Lemme: formal
Actual : structure , Lemme: structure
Actual : of , Lemme: of
Actual : the , Lemme: the
Actual : business , Lemme: business
Actual : week , Lemme: week
Actual : . , Lemme: .
Actual : When , Lemme: When
Actual : he , Lemme: he
Actual : was , Lemme: wa
Actual : younger , Lemme: younger
Actual : he , Lemme: he
Actual : used , Lemme: used
Actual : to , Lemme: to
Actual : stay , Lemme: stay
Actual : late , Lemme: late
Actual : after , Lemme: after
Actual : school , Lemme: school
Actual : on , Lemme: on
Actual : Fridays , Lemme: Fridays
Actual : and , Lemme: and
Actual : come , Lemme: come
Actual : in , Lemme: in
Actual : early , Lemme: early
Actual : on , Lemme: on
```

```
Actual : Mondays , Lemme: Mondays
Actual : , , Lemme: ,
Actual : a , Lemme: a
Actual : pattern , Lemme: pattern
Actual : his , Lemme: his
Actual : mother , Lemme: mother
Actual : referred , Lemme: referred
Actual : to , Lemme: to
Actual : with , Lemme: with
Actual : equal , Lemme: equal
Actual : parts , Lemme: part
Actual : admiration , Lemme: admiration
Actual : and , Lemme: and
Actual : disdain , Lemme: disdain
Actual : as , Lemme: a
Actual : “ , Lemme: “
Actual : studying , Lemme: studying
Actual : overtime. , Lemme: overtime.
Actual : ” , Lemme: ”
Actual : Jesus , Lemme: Jesus
Actual : , , Lemme: ,
Actual : I , Lemme: I
Actual : ’ , Lemme: ’
Actual : ve , Lemme: ve
Actual : written , Lemme: written
Actual : another , Lemme: another
Actual : loser , Lemme: loser
Actual : . , Lemme: .
```

2. Word count

Term Frequency (TF)

Formula: **tf(t,d)** = count of t in d / number of words in d

In [16]:
```
sentence1 = "Data Science is the best job of the 21st century"
sentence2 = "machine learning is the key for data science"
```

In [17]:
```
# Spliting both sentences
sentence1 = sentence1.split(" ")
sentence2 = sentence2.split(" ")
```

In [18]:
```
join = set(sentence1).union(set(sentence2))
```

In [19]:
```
join
```

Out[19]:
```
{'21st',
 'Data',
 'Science',
 'best',
 'century',
 'data',
 'for',
```

```
'is',
'job',
'key',
'learning',
'machine',
'of',
'science',
'the'}
```

In [20]:
```
wordDict1 = dict.fromkeys(join, 0)
wordDict2 = dict.fromkeys(join, 0)

for word in sentence1:
  wordDict1[word] += 1


for word in sentence2:
  wordDict2[word] += 1
```

In [21]:
```
pd.DataFrame([wordDict1, wordDict2])
```

Out[21]:

	the	is	learning	for	century	Science	job	key	of	data	science	machine	21st	best	Data
0	2	1	0	0	1	1	1	0	1	0	0	0	1	1	1
1	1	1	1	1	0	0	0	1	0	1	1	1	0	0	0

In [22]:
```
def getTF(wordDict, data):
  res = {}
  corpusCount = len(data)
  for word, count in wordDict.items():
    res[word] = count/float(corpusCount)
  return res

tf1 = getTF(wordDict1, sentence1)
tf2 = getTF(wordDict2, sentence2)
```

In [23]:
```
tf2
```

Out[23]:
```
{'the': 0.125,
 'is': 0.125,
 'learning': 0.125,
 'for': 0.125,
 'century': 0.0,
 'Science': 0.0,
 'job': 0.0,
 'key': 0.125,
 'of': 0.0,
 'data': 0.125,
 'science': 0.125,
 'machine': 0.125,
 '21st': 0.0,
 'best': 0.0,
 'Data': 0.0}
```

In [24]:
```
tf = pd.DataFrame([tf1, tf2])
```

In [25]:
```
tf
```

Out[25]:

	the	is	learning	for	century	Science	job	key	of	data	science	machine	21st	best
0	0.200	0.100	0.000	0.000	0.1	0.1	0.1	0.000	0.1	0.000	0.000	0.000	0.1	0.1
1	0.125	0.125	0.125	0.125	0.0	0.0	0.0	0.125	0.0	0.125	0.125	0.125	0.0	0.0

In [26]:
```
filtered_sentence = [w for w in wordDict1 if w not in stop_words]
filtered_sentence
```

Out[26]:
```
['learning',
 'century',
 'Science',
 'job',
 'key',
 'data',
 'science',
 'machine',
 '21st',
 'best',
 'Data']
```

Inverse Document Frequency (IDF)

Formula: idf(t) = log(N/(df + 1))

In [27]:
```
import math
def getIDF(documents):
  n = len(documents)
  res = {}
  res = dict.fromkeys(documents[0].keys(), 0)

  for word, count in res.items():
    res[word] = math.log10(n / (float(count) + 1))
  return res
```

In [28]:
```
idfs = getIDF([wordDict1, wordDict2])
idfs
```

Out[28]:
```
{'the': 0.3010299956639812,
 'is': 0.3010299956639812,
 'learning': 0.3010299956639812,
 'for': 0.3010299956639812,
 'century': 0.3010299956639812,
 'Science': 0.3010299956639812,
 'job': 0.3010299956639812,
 'key': 0.3010299956639812,
 'of': 0.3010299956639812,
```

```
'data': 0.3010299956639812,
'science': 0.3010299956639812,
'machine': 0.3010299956639812,
'21st': 0.3010299956639812,
'best': 0.3010299956639812,
'Data': 0.3010299956639812}
```

In [29]:
```
def getTFIDF(tf, idf):
  tfidf = {}
  for word, count in tf.items():
    tfidf[word] = count*idf[word]
  return tfidf
```

In [30]:
```
tfidf1 = getTFIDF(tf1, idfs)
tfidf2 = getTFIDF(tf2, idfs)

pdTFIDF = pd.DataFrame([tfidf1, tfidf2])
pdTFIDF
```

Out[30]:

	the	is	learning	for	century	Science	job	key	of	data	s
0	0.060206	0.030103	0.000000	0.000000	0.030103	0.030103	0.030103	0.000000	0.030103	0.000000	0.
1	0.037629	0.037629	0.037629	0.037629	0.000000	0.000000	0.000000	0.037629	0.000000	0.037629	0.

TFIDF using sklearn

In [31]:
```
from sklearn.feature_extraction.text import TfidfVectorizer


firstV= "Data Science is the sexiest job of the 21st century"
secondV= "machine learning is the key for data science"

vectorize= TfidfVectorizer()

response= vectorize.fit_transform([firstV, secondV])

# get idf values
print('\nIdf values:')
for ele1, ele2 in zip(vectorize.get_feature_names(), vectorize.idf_):
    print(ele1, ':', ele2)
```

```
Idf values:
21st : 1.4054651081081644
century : 1.4054651081081644
data : 1.0
for : 1.4054651081081644
is : 1.0
job : 1.4054651081081644
key : 1.4054651081081644
learning : 1.4054651081081644
machine : 1.4054651081081644
of : 1.4054651081081644
science : 1.0
```

```
sexiest : 1.4054651081081644
the : 1.0
```

In [32]:

```
print('\nTf-Idf values:')
print(response)
```

```
Tf-Idf values:
  (0, 1)        0.34211869506421816
  (0, 0)        0.34211869506421816
  (0, 9)        0.34211869506421816
  (0, 5)        0.34211869506421816
  (0, 11)       0.34211869506421816
  (0, 12)       0.48684053853849035
  (0, 4)        0.24342026926924518
  (0, 10)       0.24342026926924518
  (0, 2)        0.24342026926924518
  (1, 3)        0.40740123733358447
  (1, 6)        0.40740123733358447
  (1, 7)        0.40740123733358447
  (1, 8)        0.40740123733358447
  (1, 12)       0.28986933576883284
  (1, 4)        0.28986933576883284
  (1, 10)       0.28986933576883284
  (1, 2)        0.28986933576883284
```

In [ ]:

Assignment 8 - Data Visualization 1

Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

- Use the inbuild 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.
- Write a code to check how the price of the ticket for each passenger is distributed by plotting a histogram.

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

In [3]:
```
data=pd.read_csv('titanic.csv')
```

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [5]:
```
data.shape
```

Out[5]:
```
(891, 12)
```

In [6]:
```
data.describe()
```

Out[6]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [7]:
```
data
```

Out[7]:

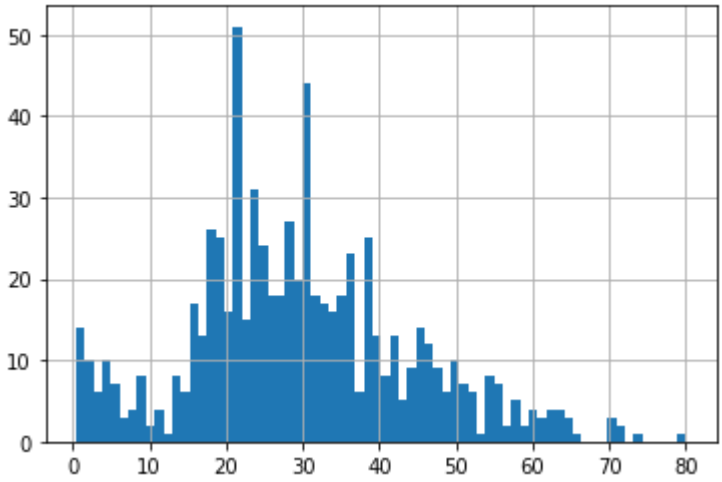
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	I
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	I
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	

891 rows × 12 columns

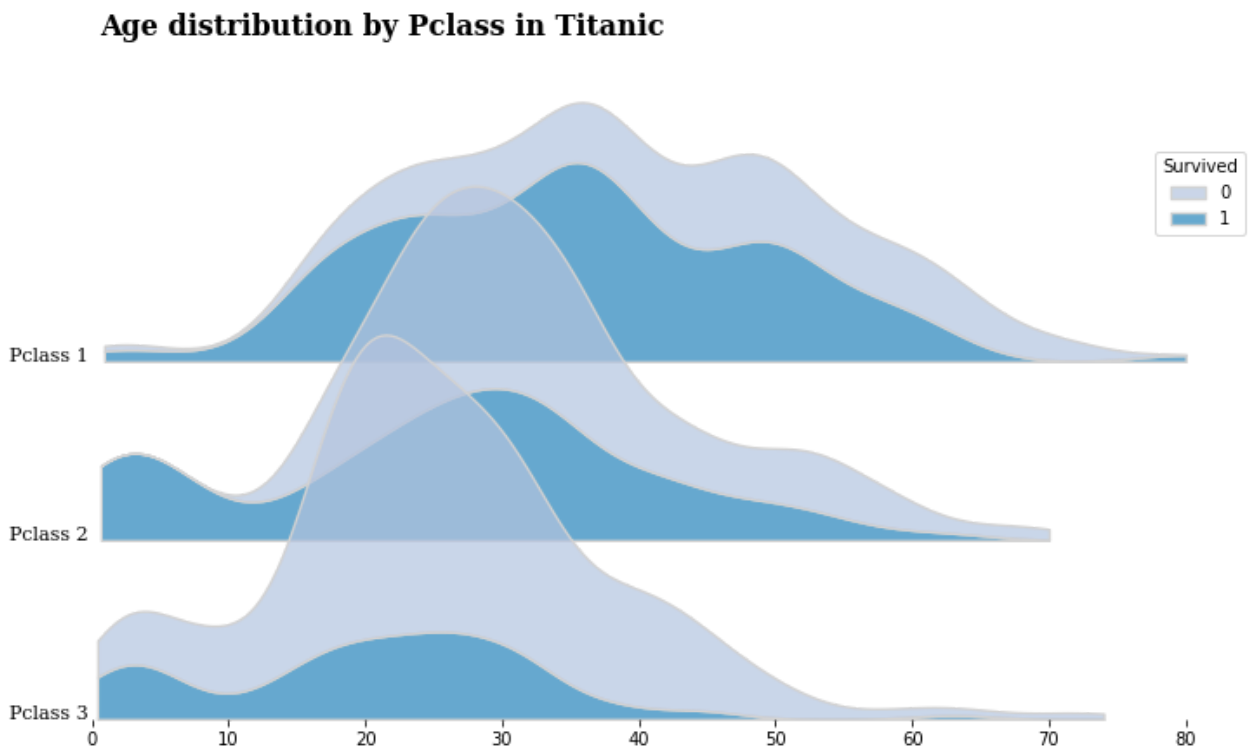
In [8]:
```
###Now let us look at the ages of the passengers

data['Age'].hist(bins=70)
```

Out[8]:
```
<AxesSubplot:>
```



In [13]:
```
fig = plt.figure(figsize=(12, 8))
gs = fig.add_gridspec(3,1)
gs.update(hspace= -0.55)

axes = list()
colors = ["#022133", "#5c693b", "#51371c"]

for idx, cls, c in zip(range(3), sorted(data['Pclass'].unique()), colors):
    axes.append(fig.add_subplot(gs[idx, 0]))

    # you can also draw density plot with matplotlib + scipy.
    sns.kdeplot(x='Age', data=data[data['Pclass']==cls],
                fill=True, ax=axes[idx], cut=0, bw_method=0.25,
                lw=1.4, edgecolor='lightgray', hue='Survived',
                multiple="stack", palette='PuBu', alpha=0.7
               )
```

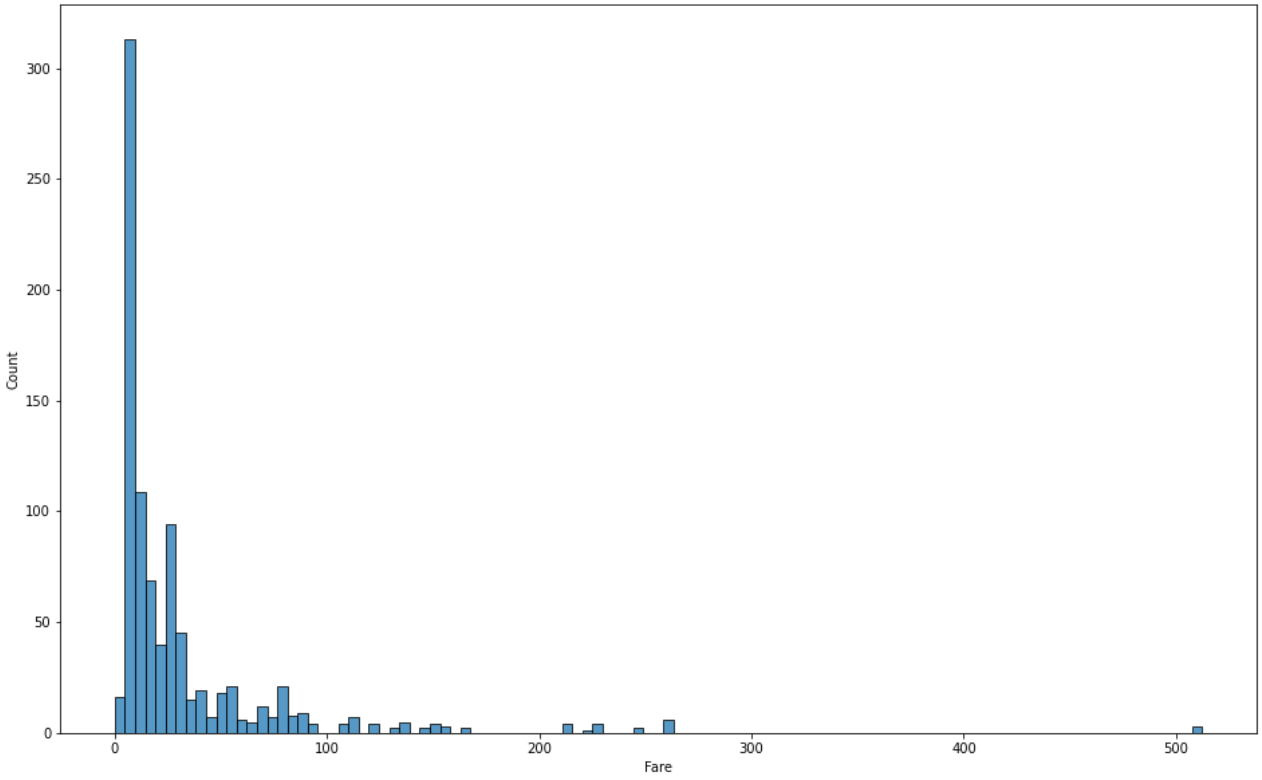
```
    axes[idx].set_ylim(0, 0.04)
    axes[idx].set_xlim(0, 85)

    axes[idx].set_yticks([])
    if idx != 2 : axes[idx].set_xticks([])
    axes[idx].set_ylabel('')
    axes[idx].set_xlabel('')

    spines = ["top","right","left","bottom"]
    for s in spines:
        axes[idx].spines[s].set_visible(False)

    axes[idx].patch.set_alpha(0)
    axes[idx].text(-0.2,0,f'Pclass {cls}',fontweight="light", fontfamily='serif', fonts
    if idx != 1 : axes[idx].get_legend().remove()

fig.text(0.13,0.81,"Age distribution by Pclass in Titanic", fontweight="bold", fontfami

plt.show()
```



From above graph, we can infer that there are less numbers of survivers form class 2 & 3 and their age group is between 10-30 years

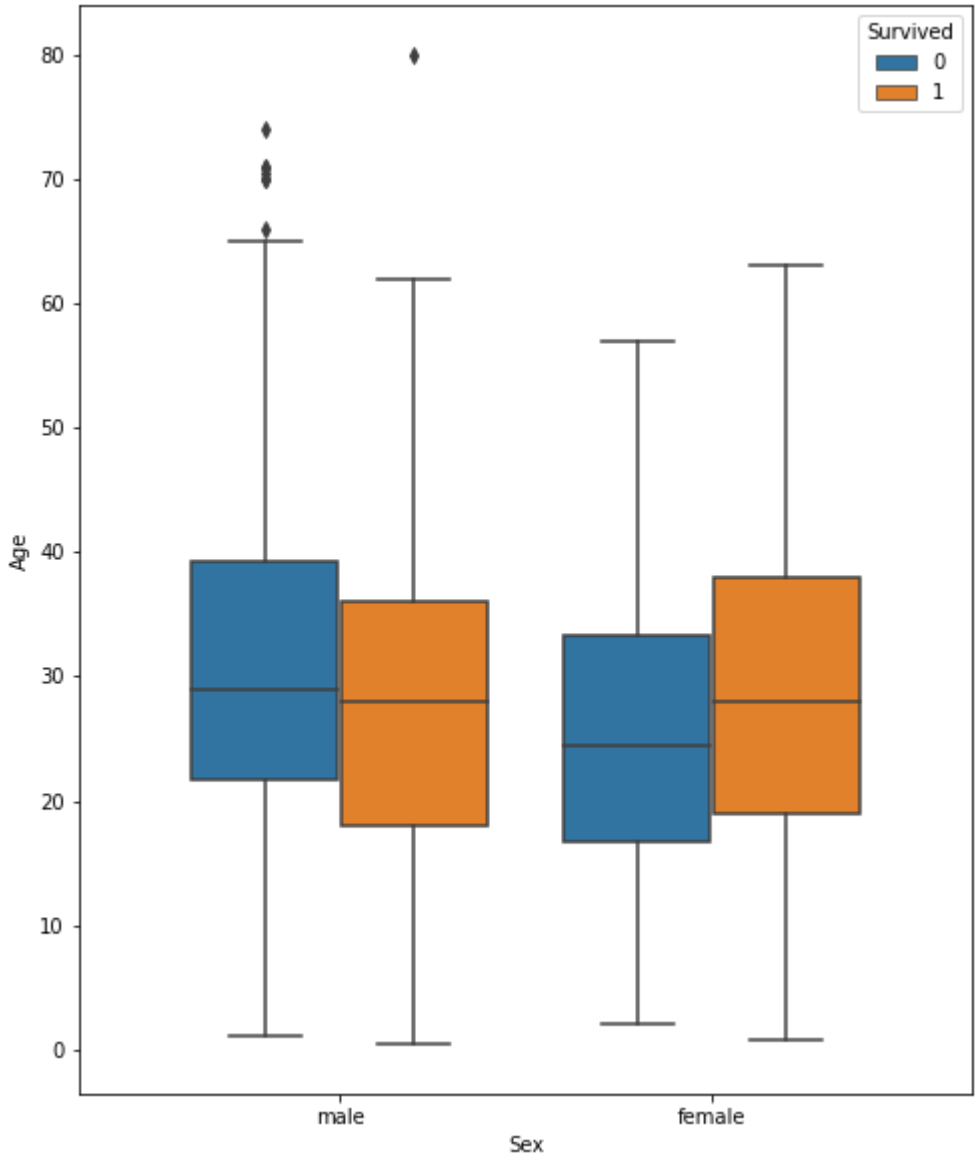
In [10]:

```
plt.figure(figsize=(16,10))
sns.histplot(data=data['Fare'])
plt.show()
```



In [11]:
```
plt.figure(figsize=(8,10))
sns.boxplot(y=data['Age'], x=data['Sex'], hue=data['Survived'])
plt.plot()
```

Out[11]:
```
[]
```



1) Number of females survived is more than female casualties.

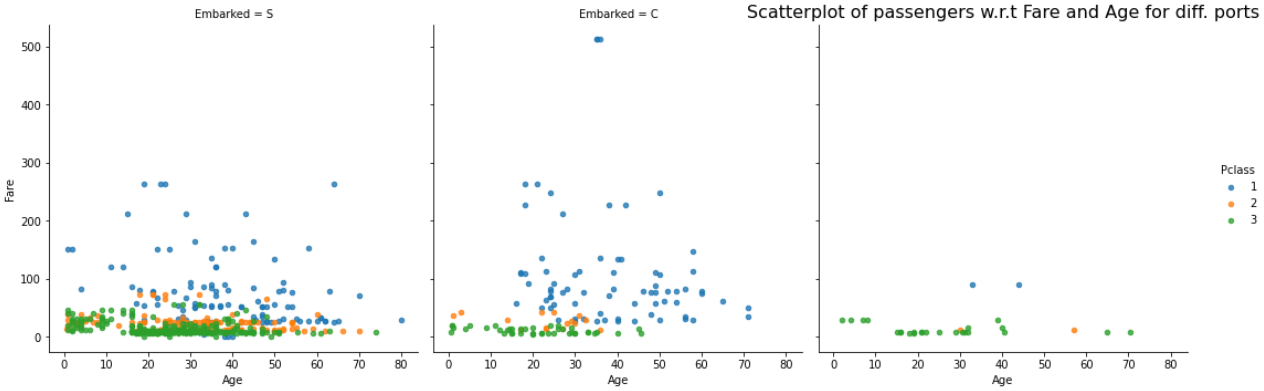
2) Number of males survived is less than the male casualties.

3) Hence, first priority was to save females then males.

In [12]:
```
sns.lmplot('Age', 'Fare', data=data, fit_reg=False, hue="Pclass", col="Embarked", scatt
plt.subplots_adjust(top=0.9)
plt.title('Scatterplot of passengers w.r.t Fare and Age for diff. ports', fontsize=16)
```

```
c:\users\orionoriginal\appdata\local\programs\python\python39\lib\site-packages\seaborn
\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. F
rom version 0.12, the only valid positional argument will be `data`, and passing other a
rguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[12]:
```
Text(0.5, 1.0, 'Scatterplot of passengers w.r.t Fare and Age for diff. ports')
```



From above plot we get that more number of passengers Embarked at port S & C

Assignment 9 - Data Visualization 2

Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

- Use the inbuild dataset 'titanic'. Plot a box plot for distribution of age with respect to each gender along with the information about wheather they survived or not.
- Write observations on the inference form the above statistics.

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


data = pd.read_csv("train.csv")
data.head()
```

Out[1]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Eml
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	

In [2]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
```

```
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

VARIABLE DESCRIPTIONS

- Pclass Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
- survival Survival (0 = No; 1 = Yes)
- name Name
- sex Sex
- age Age
- sibsp Number of Siblings/Spouses Aboard
- parch Number of Parents/Children Aboard
- ticket Ticket Number
- fare Passenger Fare (British pound)
- cabin Cabin
- embarked Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

In [3]:
```
data.shape
```

Out[3]:
```
(891, 12)
```

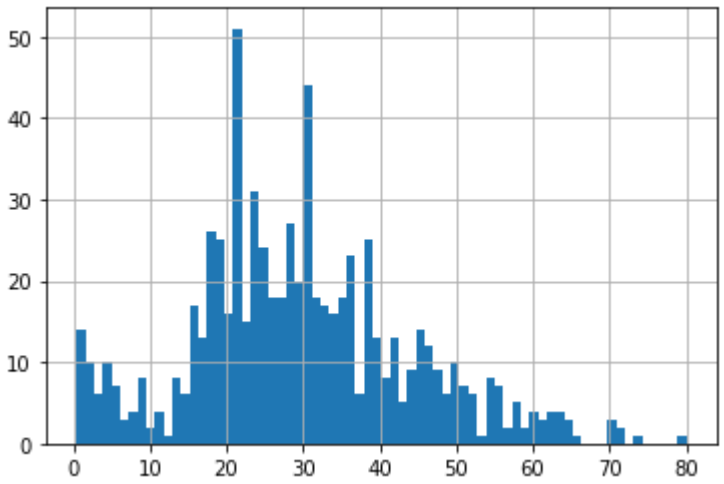
In [4]:
```
data.describe()
```

Out[4]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [5]:
```
###Now let us look at the ages of the passengers

data['Age'].hist(bins=70)
```

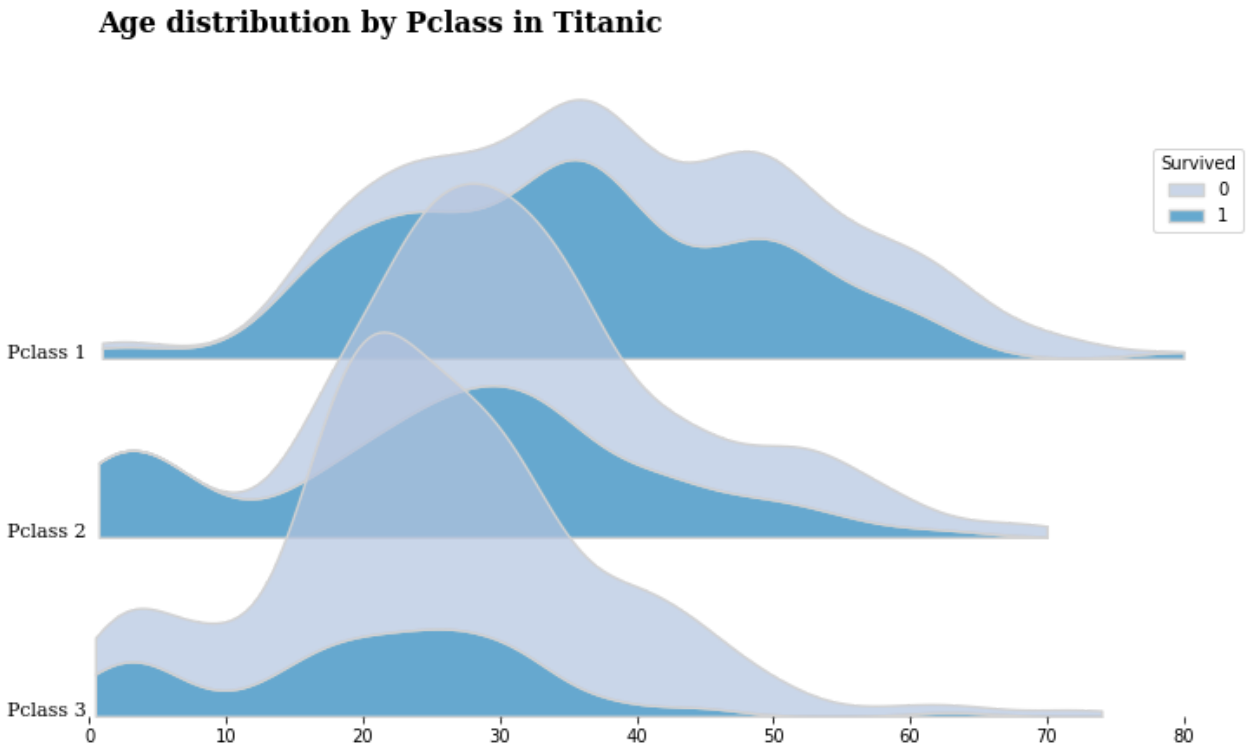
Out[5]:
```
<AxesSubplot:>
```



There are more number of people from age group 15-35 years old.

In [6]:
```
fig = plt.figure(figsize=(12, 8))
gs = fig.add_gridspec(3,1)
gs.update(hspace= -0.55)

axes = list()
colors = ["#022133", "#5c693b", "#51371c"]

for idx, cls, c in zip(range(3), sorted(data['Pclass'].unique()), colors):
    axes.append(fig.add_subplot(gs[idx, 0]))

    # you can also draw density plot with matplotlib + scipy.
    sns.kdeplot(x='Age', data=data[data['Pclass']==cls],
                fill=True, ax=axes[idx], cut=0, bw_method=0.25,
                lw=1.4, edgecolor='lightgray', hue='Survived',
                multiple="stack", palette='PuBu', alpha=0.7
               )

    axes[idx].set_ylim(0, 0.04)
    axes[idx].set_xlim(0, 85)

    axes[idx].set_yticks([])
    if idx != 2 : axes[idx].set_xticks([])
    axes[idx].set_ylabel('')
    axes[idx].set_xlabel('')

    spines = ["top","right","left","bottom"]
    for s in spines:
        axes[idx].spines[s].set_visible(False)

    axes[idx].patch.set_alpha(0)
    axes[idx].text(-0.2,0,f'Pclass {cls}',fontweight="light", fontfamily='serif', fonts
    if idx != 1 : axes[idx].get_legend().remove()

fig.text(0.13,0.81,"Age distribution by Pclass in Titanic", fontweight="bold", fontfami
```

```
plt.show()
```



From above graph, we can infer that there are less numbers of survivers form class 2 & 3 and their age group is between 10-30 years

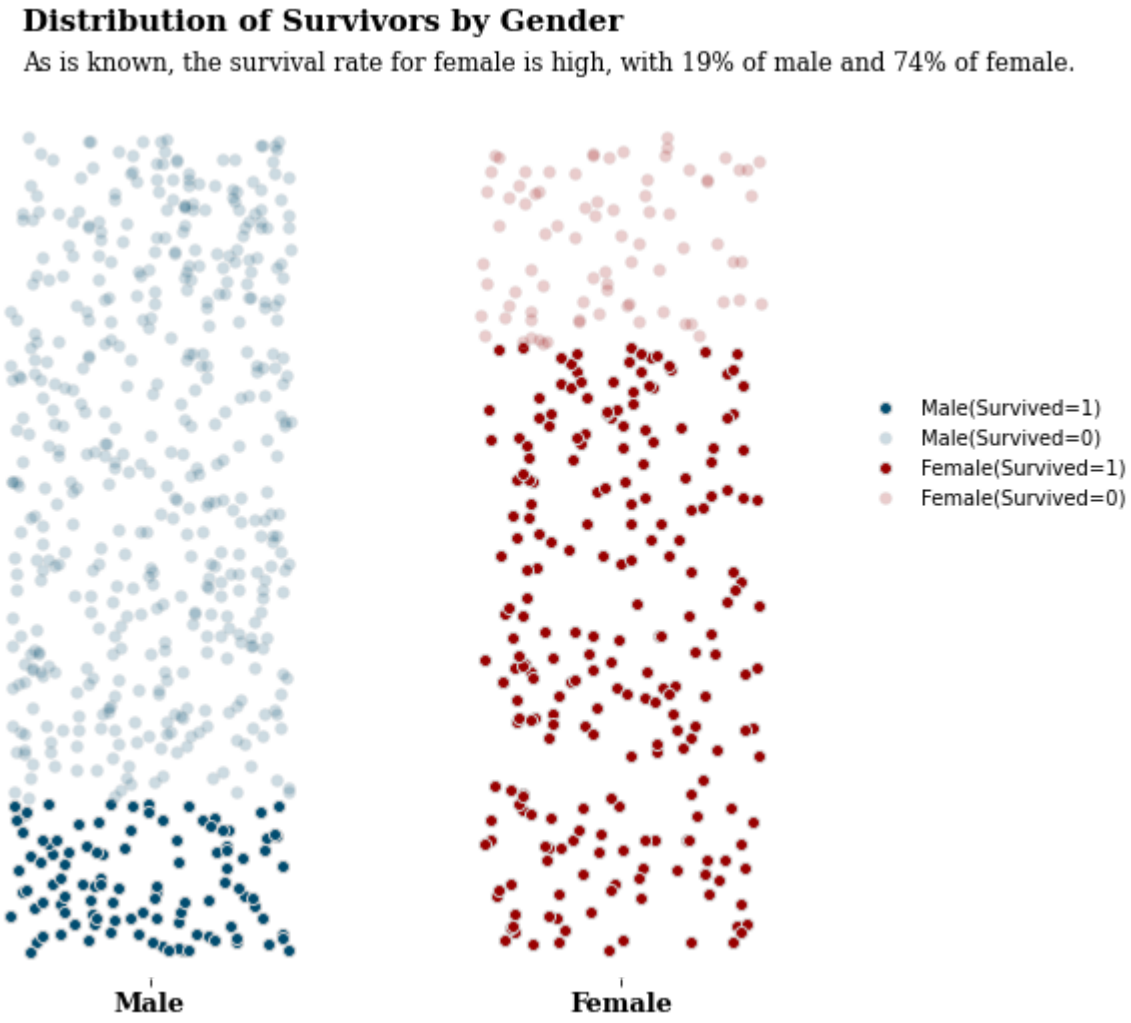
In [7]:
```
survival_rate = data.groupby(['Sex']).mean()[['Survived']]
male_rate = survival_rate.loc['male']
female_rate = survival_rate.loc['female']
display(survival_rate)
```

Sex	Survived
female	0.742038
male	0.188908

In [8]:
```
male_pos = np.random.uniform(0, male_rate, len(data[(data['Sex']=='male') & (data['Surv
male_neg = np.random.uniform(male_rate, 1, len(data[(data['Sex']=='male') & (data['Surv
female_pos = np.random.uniform(0, female_rate, len(data[(data['Sex']=='female') & (data
female_neg = np.random.uniform(female_rate, 1, len(data[(data['Sex']=='female') & (data
```

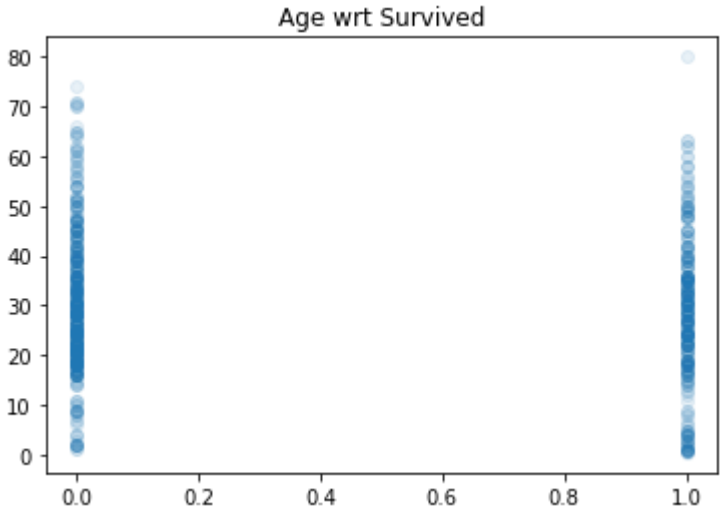
In [9]:
```
fig, ax = plt.subplots(1, 1, figsize=(9, 7))

np.random.seed(42)

# Male Stripplot
ax.scatter(np.random.uniform(-0.3, 0.3, len(male_pos)), male_pos, color='#004c70', edge
```

```
ax.scatter(np.random.uniform(-0.3, 0.3, len(male_neg)), male_neg, color='#004c70', edge

# Female Stripplot
ax.scatter(1+np.random.uniform(-0.3, 0.3, len(female_pos)), female_pos, color='#990000'
ax.scatter(1+np.random.uniform(-0.3, 0.3, len(female_neg)), female_neg, color='#990000'

# Set Figure & Axes
ax.set_xlim(-0.5, 2.0)
ax.set_ylim(-0.03, 1.1)

# Ticks
ax.set_xticks([0, 1])
ax.set_xticklabels(['Male', 'Female'], fontweight='bold', fontfamily='serif', fontsize=
ax.set_yticks([], minor=False)
ax.set_ylabel('')

# Spines
for s in ["top","right","left", 'bottom']:
    ax.spines[s].set_visible(False)


# Title & Explanation
fig.text(0.1, 1, 'Distribution of Survivors by Gender', fontweight='bold', fontfamily='
fig.text(0.1, 0.96, 'As is known, the survival rate for female is high, with 19% of mal

ax.legend(loc=(0.8, 0.5), edgecolor='None')
plt.tight_layout()
plt.show()
```



In [10]:
```
plt.scatter(data.Survived, data.Age, alpha=0.1)   ## here the plot has to be transparen
plt.title("Age wrt Survived")
```

Out[10]:
```
Text(0.5, 1.0, 'Age wrt Survived')
```



So From the above we can understand that, some of the older people died (between 50-70) and some of the younger people (between 20-40) survived more.

In [13]:

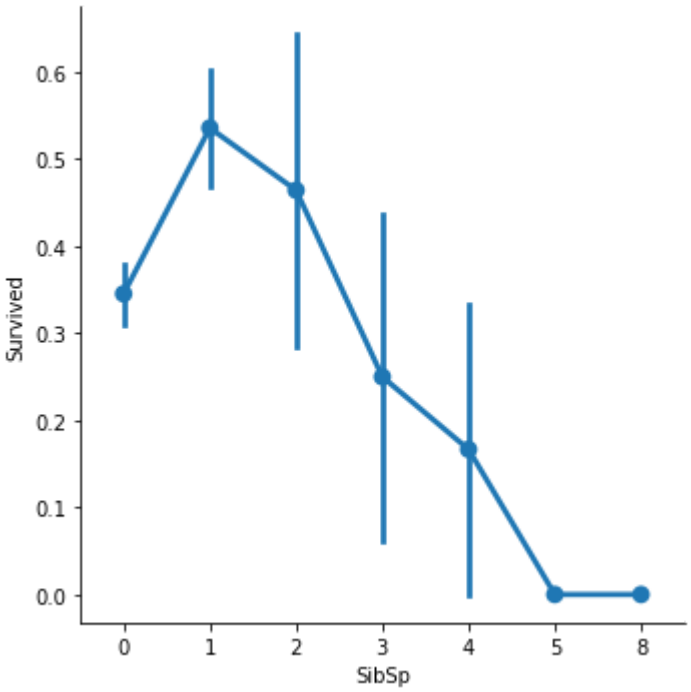
```
g = sns.FacetGrid(data=df,col="Sex",margin_titles=True)
g.map(sns.boxplot,"Survived","Age",order=[False,True])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_19964/1993416722.py in <module>
----> 1 g = sns.FacetGrid(data=df,col="Sex",margin_titles=True)
      2 g.map(sns.boxplot,"Survived","Age",order=[False,True])

NameError: name 'df' is not defined
```

In [14]:

```
sns.factorplot(x="SibSp", y="Survived", data=data);
```

```
C:\Users\asus\anaconda3\lib\site-packages\seaborn\categorical.py:3717: UserWarning: The
`factorplot` function has been renamed to `catplot`. The original name will be removed i
n a future release. Please update your code. Note that the default `kind` in `factorplot
` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
```



More people have survived with family size 2 and less people from family size 5

Assignment -10 -Data Visualization 3

Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

Download the Iris flower dataset or any other dataset into a DataFrame. (e.g., https://archive.ics.uci.edu/ml/datasets/Iris). Scan the dataset and give the inference as:

1. List down the features and their types (e.g., numeric, nominal) available in the dataset.
2. Create a histogram for each feature in the dataset to illustrate the feature distributions.
3. Create a box plot for each feature in the dataset.
4. Compare distributions and identify outliers.

In [19]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [20]:
```
data=pd.read_csv('iris flower.csv')
```

In [21]:
```
data.head()
```

Out[21]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [22]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

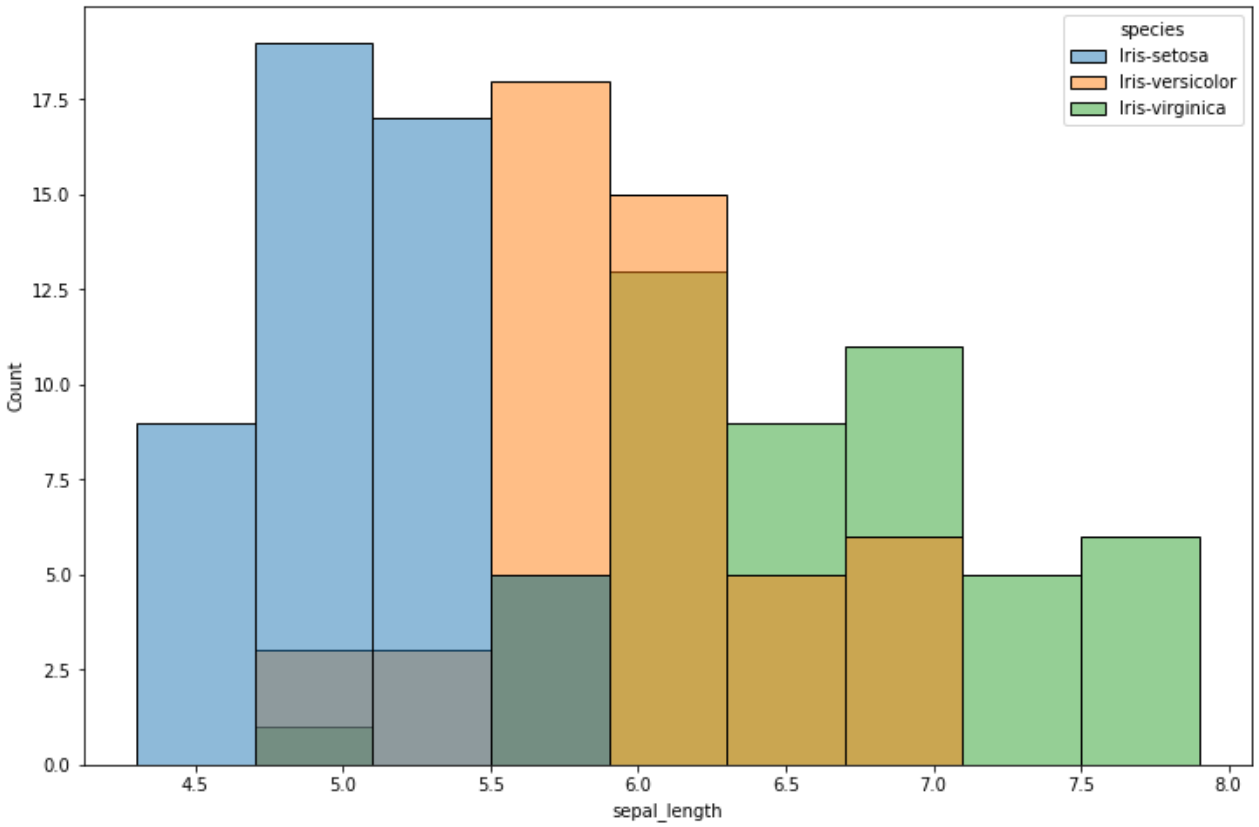
In [23]:
```
data.describe()
```

Out[23]:

	sepal_length	sepal_width	petal_length	petal_width
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

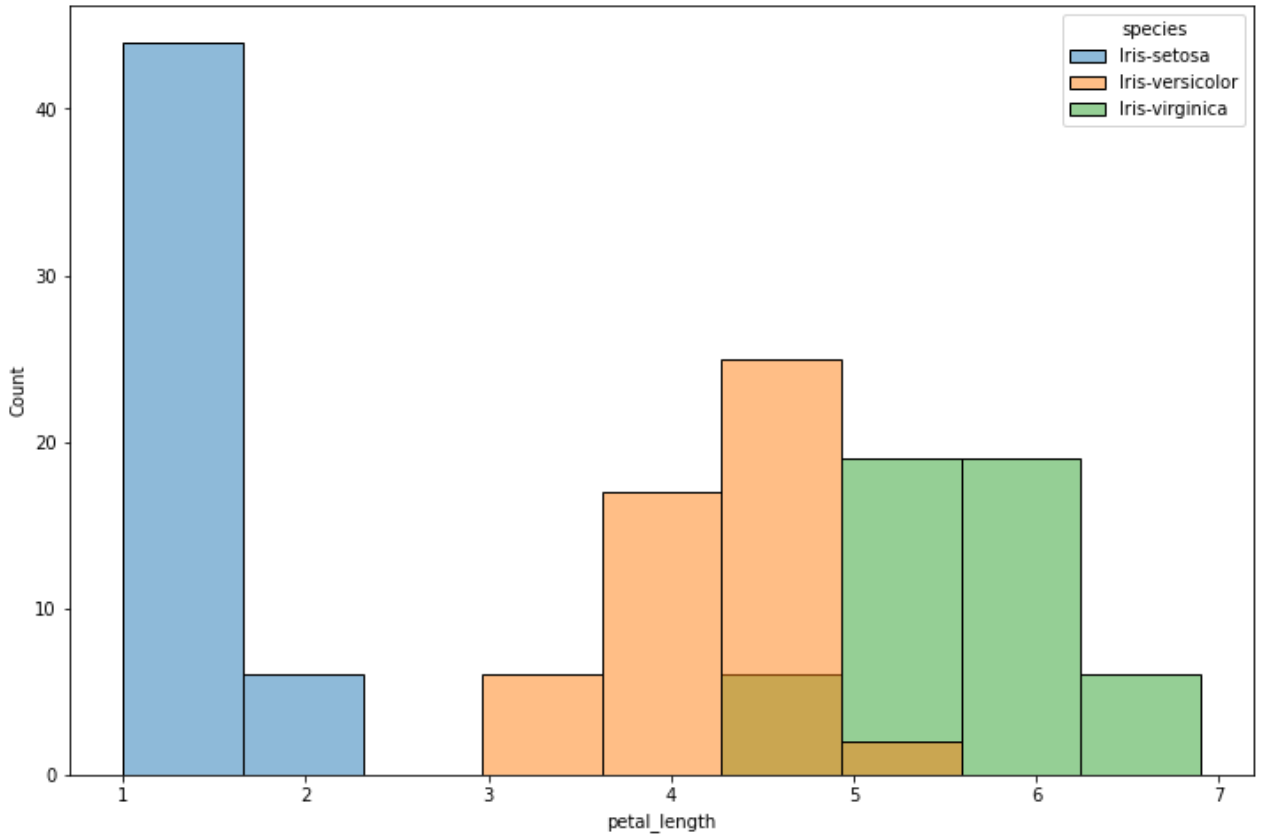
In [24]:
```
plt.figure(figsize=(12,8))
sns.histplot(x=data['sepal_length'], hue=data['species'])
plt.plot()
```

Out[24]:
```
[]
```



In [25]:
```
plt.figure(figsize=(12,8))
sns.histplot(x=data['petal_length'], hue=data['species'])
plt.plot()
```

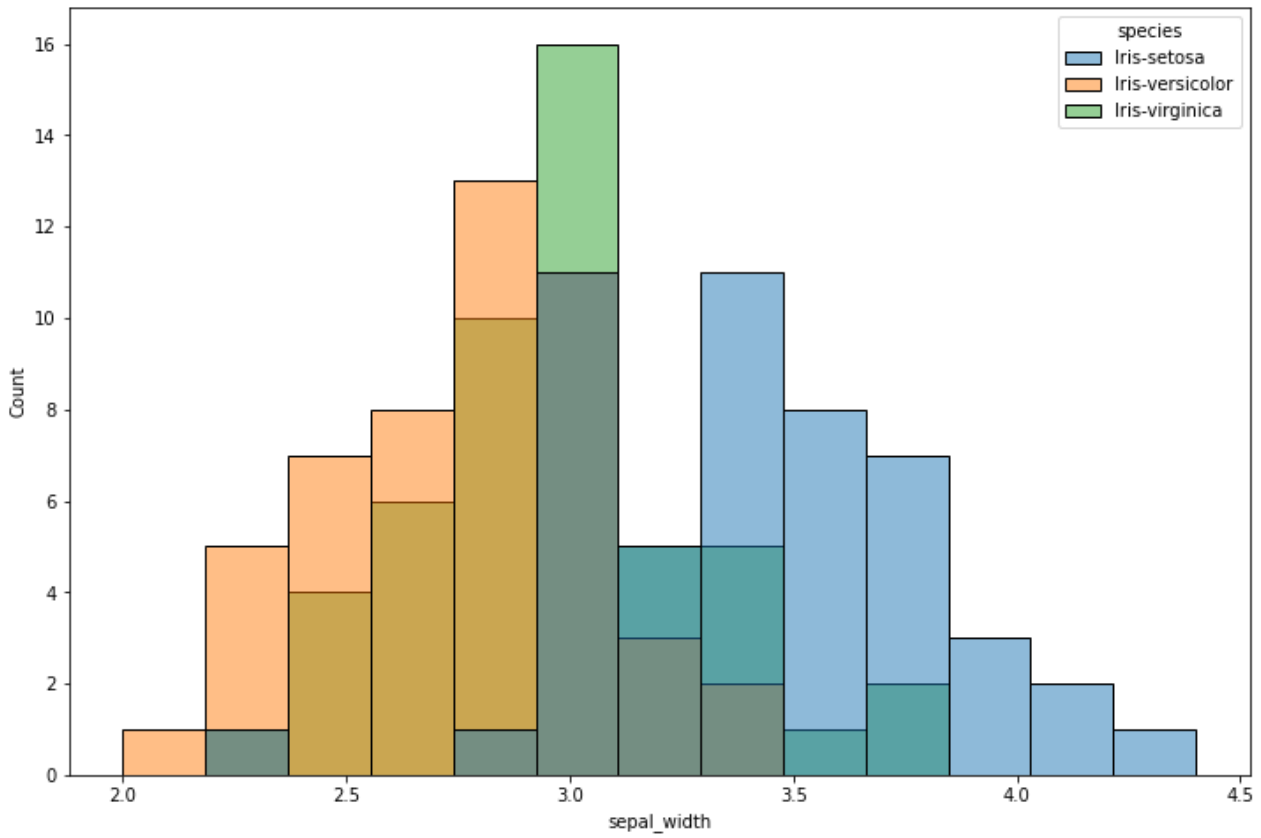
Out[25]:
```
[]
```



In [26]:

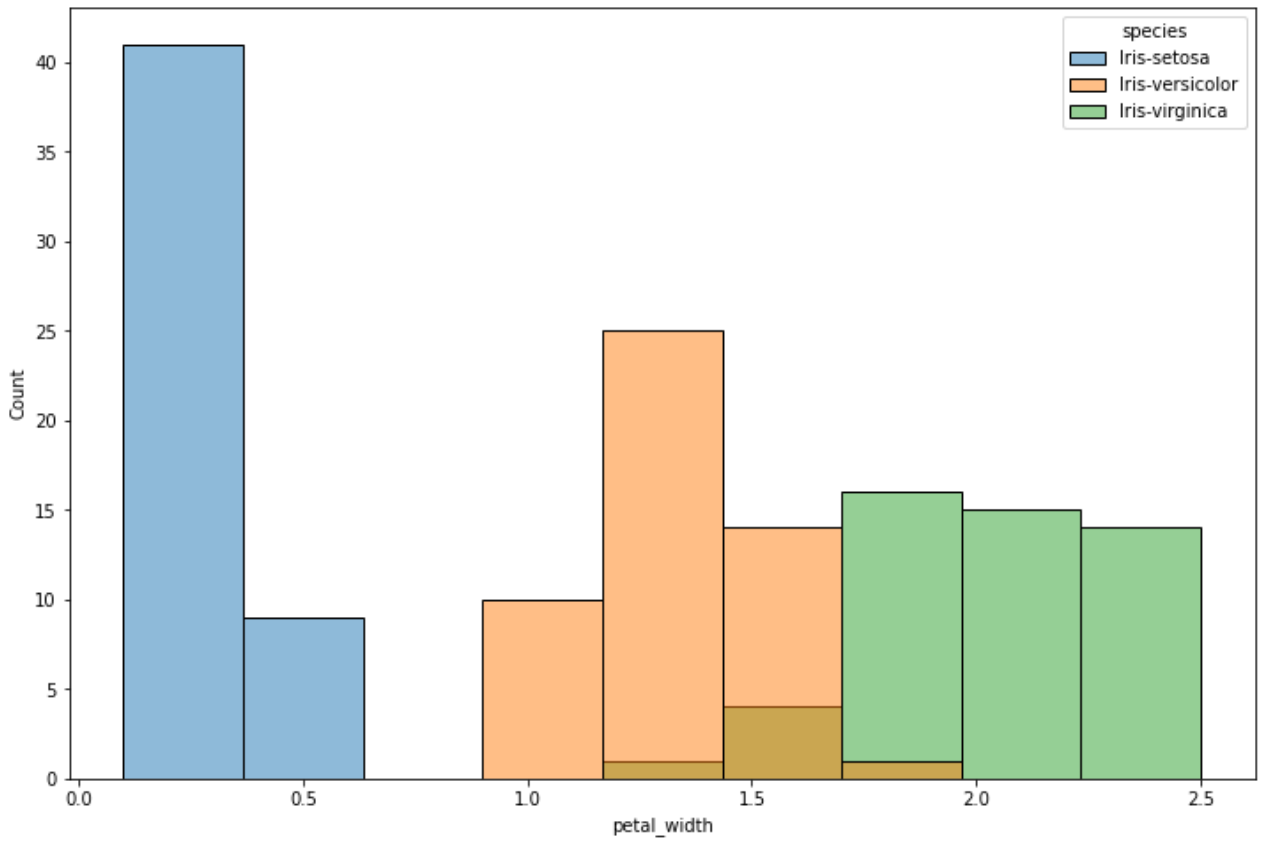
```
plt.figure(figsize=(12,8))
sns.histplot(x=data['sepal_width'], hue=data['species'])
plt.plot()
```

Out[26]: []



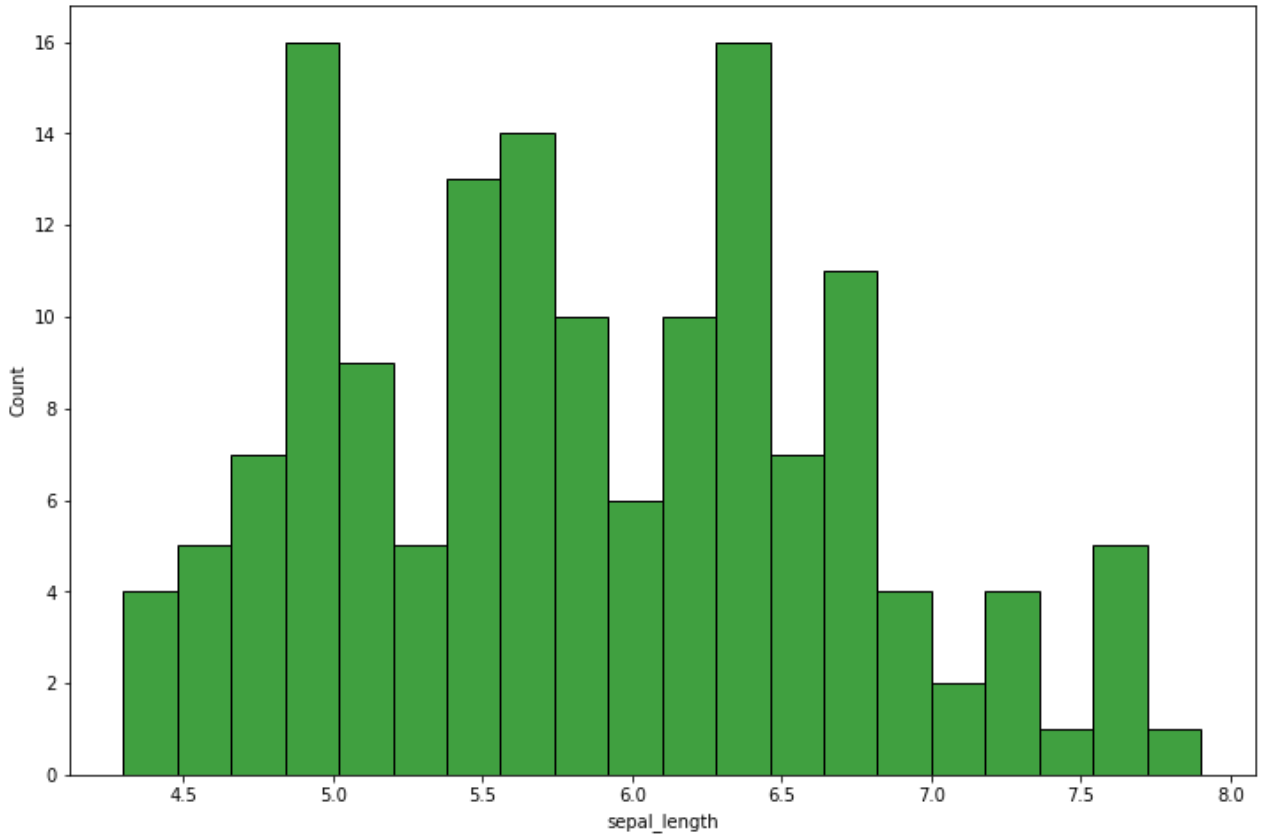
In [27]:
```
plt.figure(figsize=(12,8))
sns.histplot(x=data['petal_width'], hue=data['species'])
plt.plot()
```

Out[27]: []



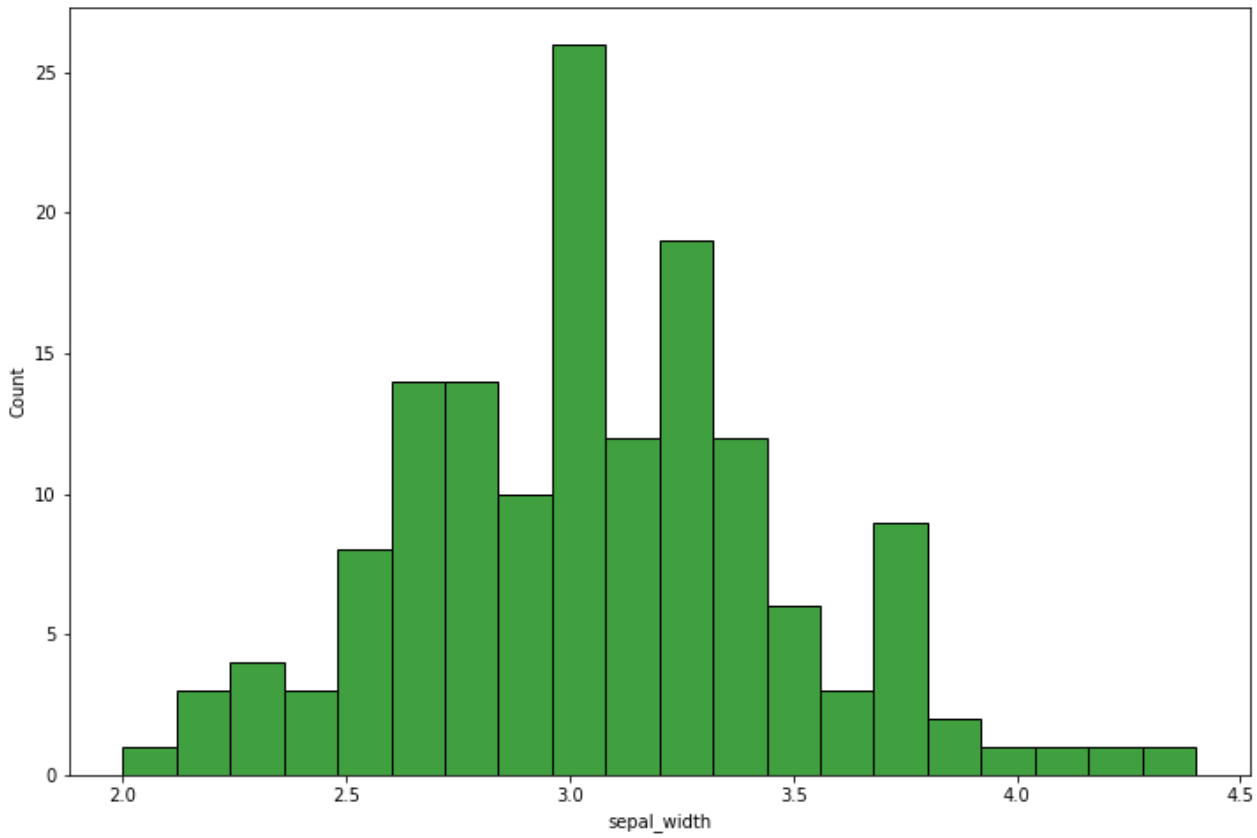
In [28]:
```
plt.figure(figsize=(12,8))
sns.histplot(x=data['sepal_length'],bins=20, color='green')
plt.plot()
```

Out[28]: []



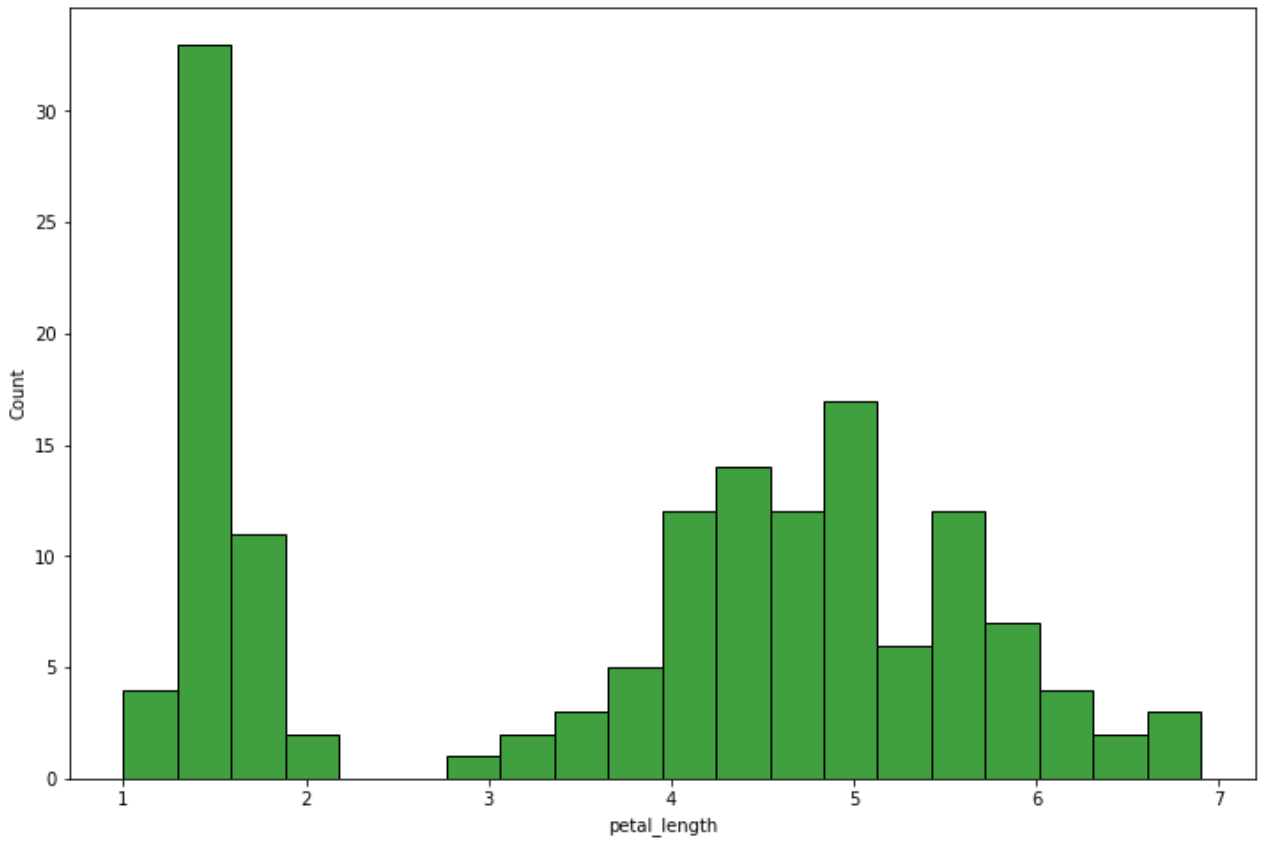
In [29]:
```
plt.figure(figsize=(12,8))
sns.histplot(x=data['sepal_width'],bins=20, color='green')
plt.plot()
```

Out[29]:
```
[]
```



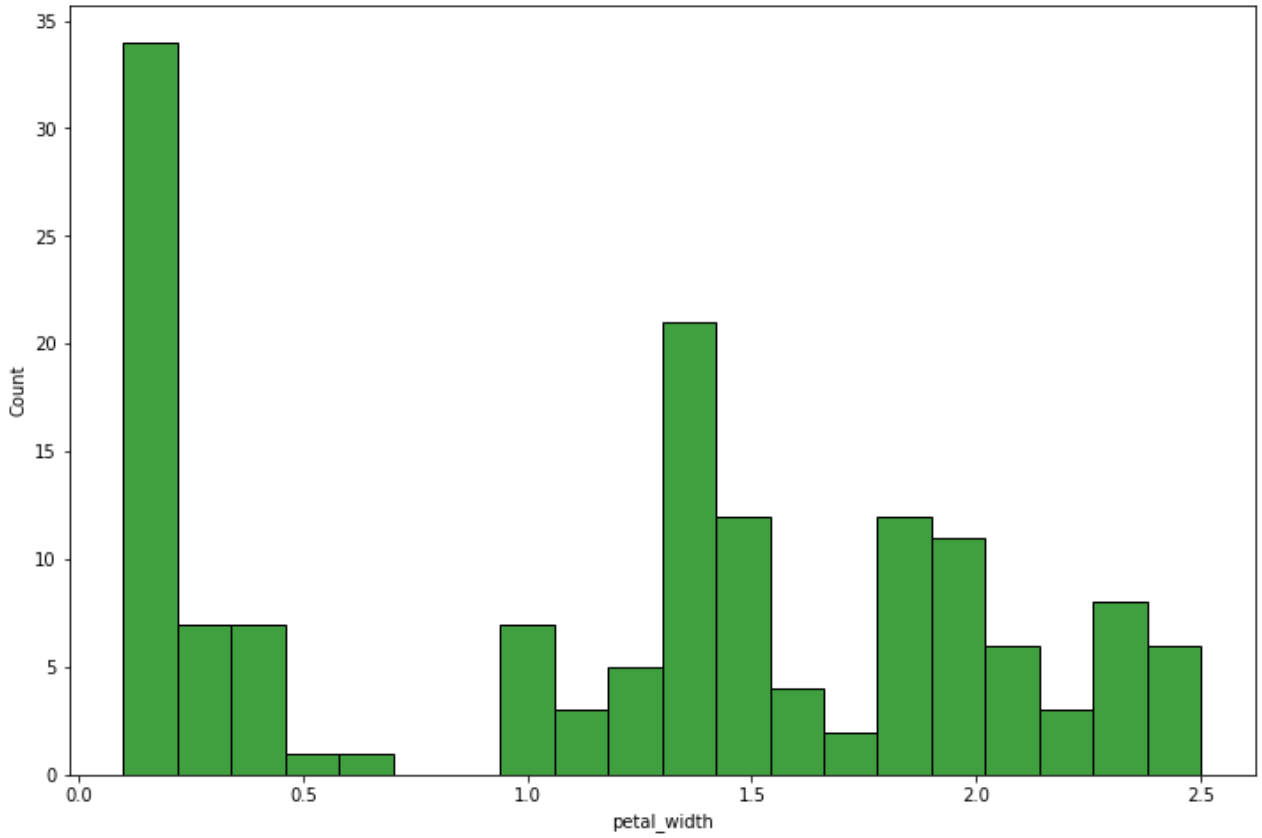
In [30]:
```
plt.figure(figsize=(12,8))
sns.histplot(x=data['petal_length'],bins=20, color='green')
plt.plot()
```

Out[30]: []



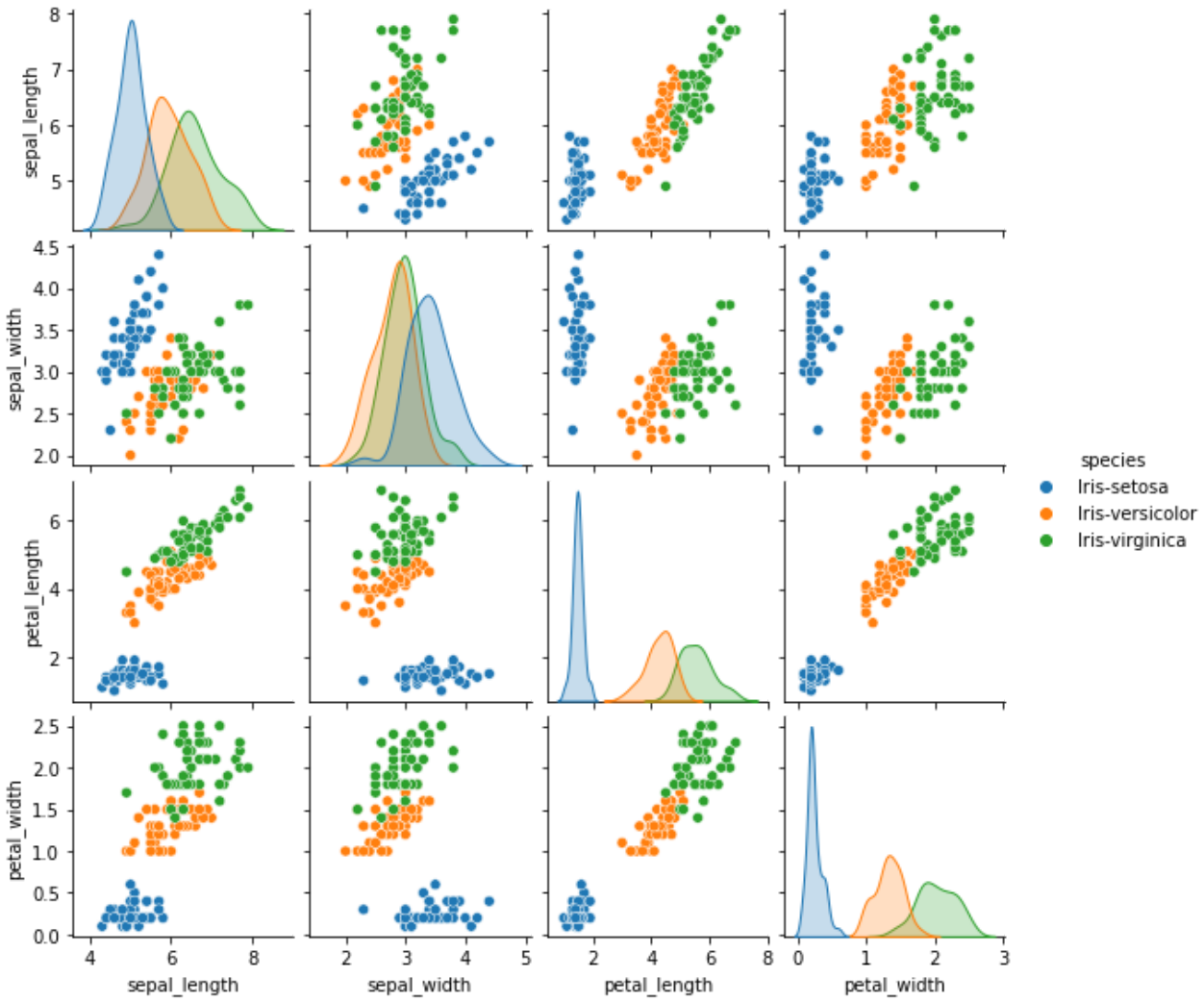
In [31]:
```
plt.figure(figsize=(12,8))
sns.histplot(x=data['petal_width'],bins=20, color='green')
plt.plot()
```

Out[31]: []



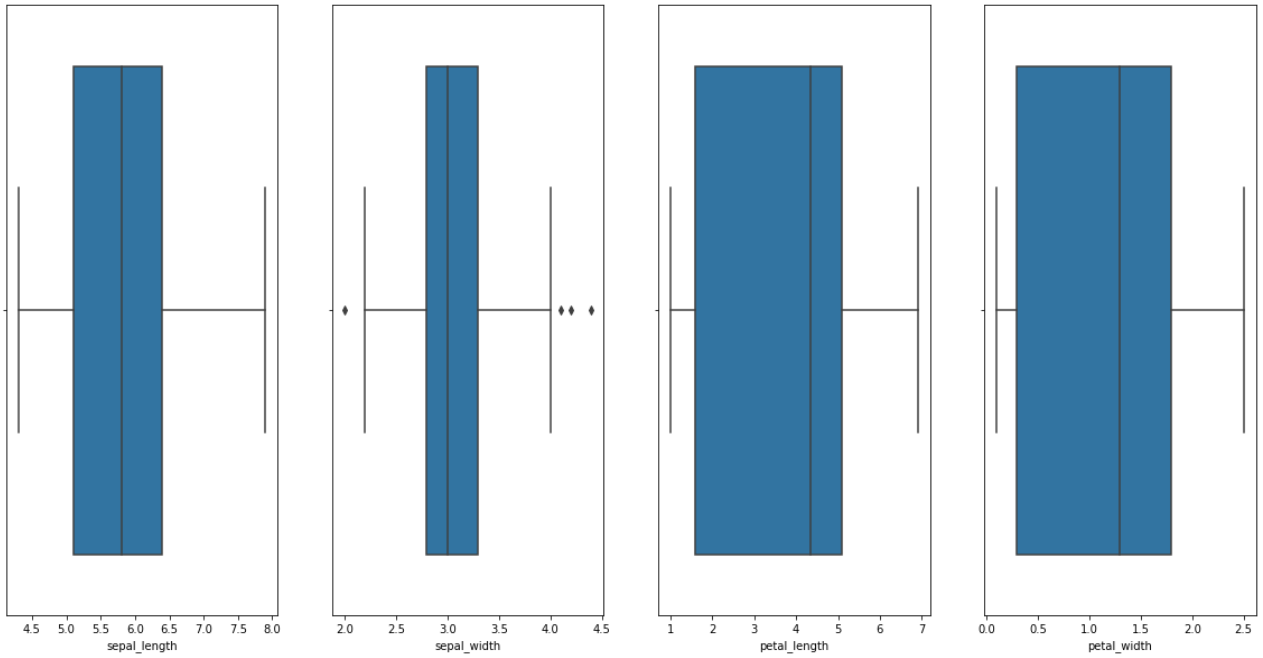
In [32]:
```
sns.pairplot(data, hue='species', height=2)
```

Out[32]:
```
<seaborn.axisgrid.PairGrid at 0x221433cf1c0>
```



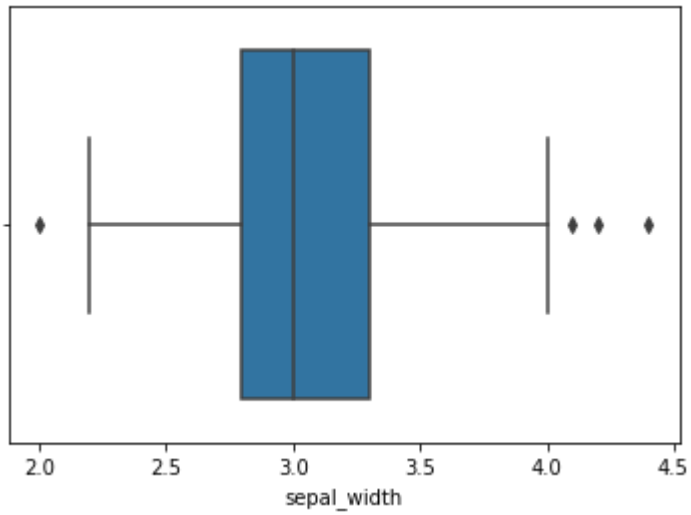
In [33]:
```
import warnings
```

In [34]:
```
warnings.filterwarnings('ignore')
features_=data.columns.values[:-1]
fig=plt.figure(figsize=(20,10))
for columns, feature in enumerate(features_):
    fig.add_subplot(1,4,columns+1)
    sns.boxplot(data[feature],data=data)
plt.show()
```



In [38]:
```
sns.boxplot(x='sepal_width', data=data)
```

Out[38]:
```
<AxesSubplot:xlabel='sepal_width'>
```



In the above graph, the values above 4 and below 2 are acting as outliers.

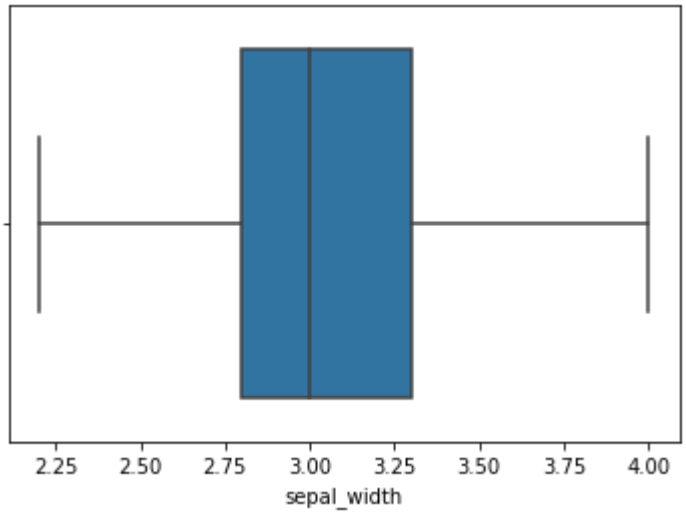
In [44]:
```
# Removing Outliers
# IQR
Q1 = np.percentile(data['sepal_width'], 25,interpolation = 'midpoint')

Q3 = np.percentile(data['sepal_width'], 75,interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", data.shape)

# Upper bound
upper = np.where(data['sepal_width'] >= (Q3+1.5*IQR))

# Lower bound
lower = np.where(data['sepal_width'] <= (Q1-1.5*IQR))
```

```
# Removing the Outliers
data.drop(upper[0], inplace = True)
data.drop(lower[0], inplace = True)

print("New Shape: ", data.shape)

sns.boxplot(x='sepal_width', data=data)
```

```
Old Shape:  (150, 5)
New Shape:  (146, 5)
```

Out[44]:

```
<AxesSubplot:xlabel='sepal_width'>
```

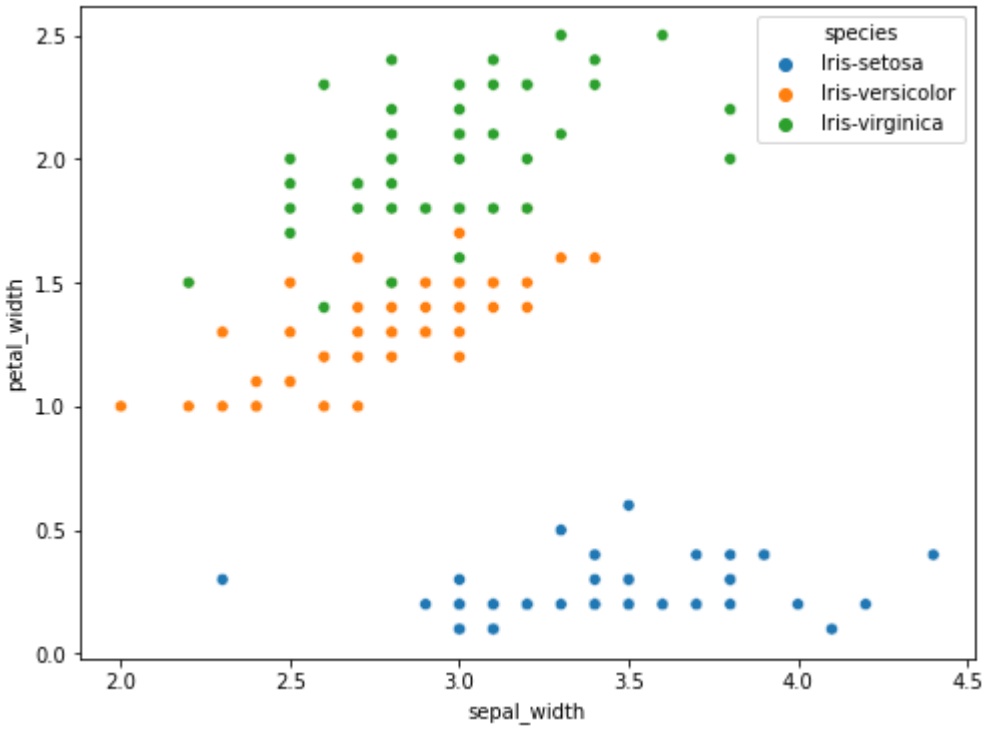


In [35]:

```
plt.figure(figsize=(8,6))
sns.scatterplot(data=data,x='sepal_width', y='petal_width', hue='species')
plt.plot()
```

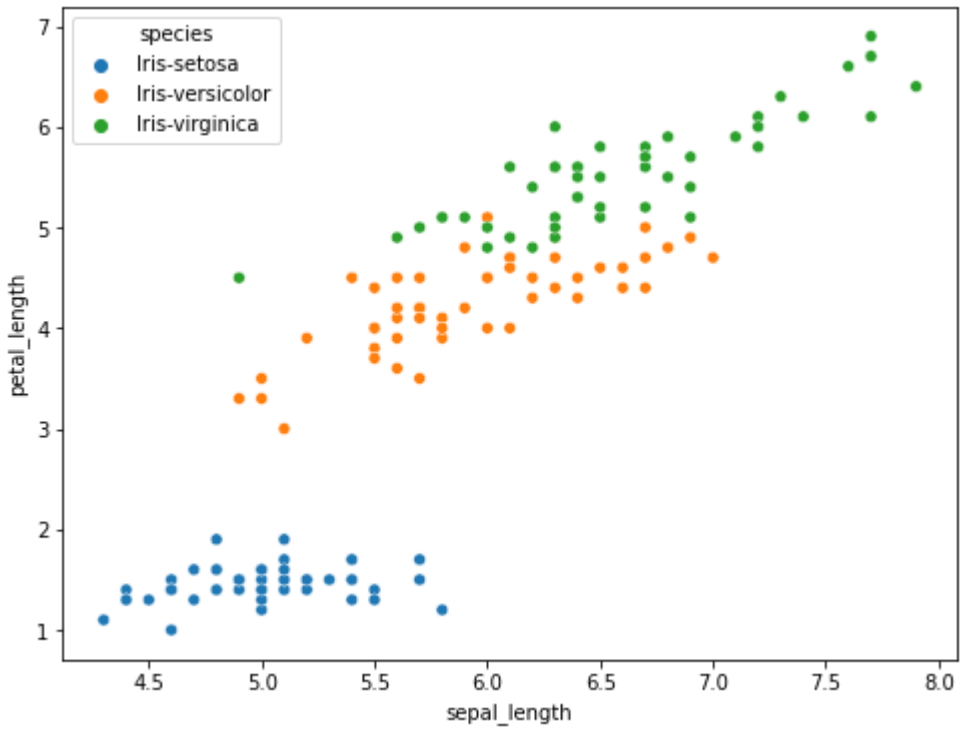
Out[35]:

```
[]
```



In [36]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data=data,x='sepal_length', y='petal_length', hue='species')
plt.plot()
```

Out[36]: []



Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

Java WordCount.java

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }
```

```
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0: 1);
  }
}
```

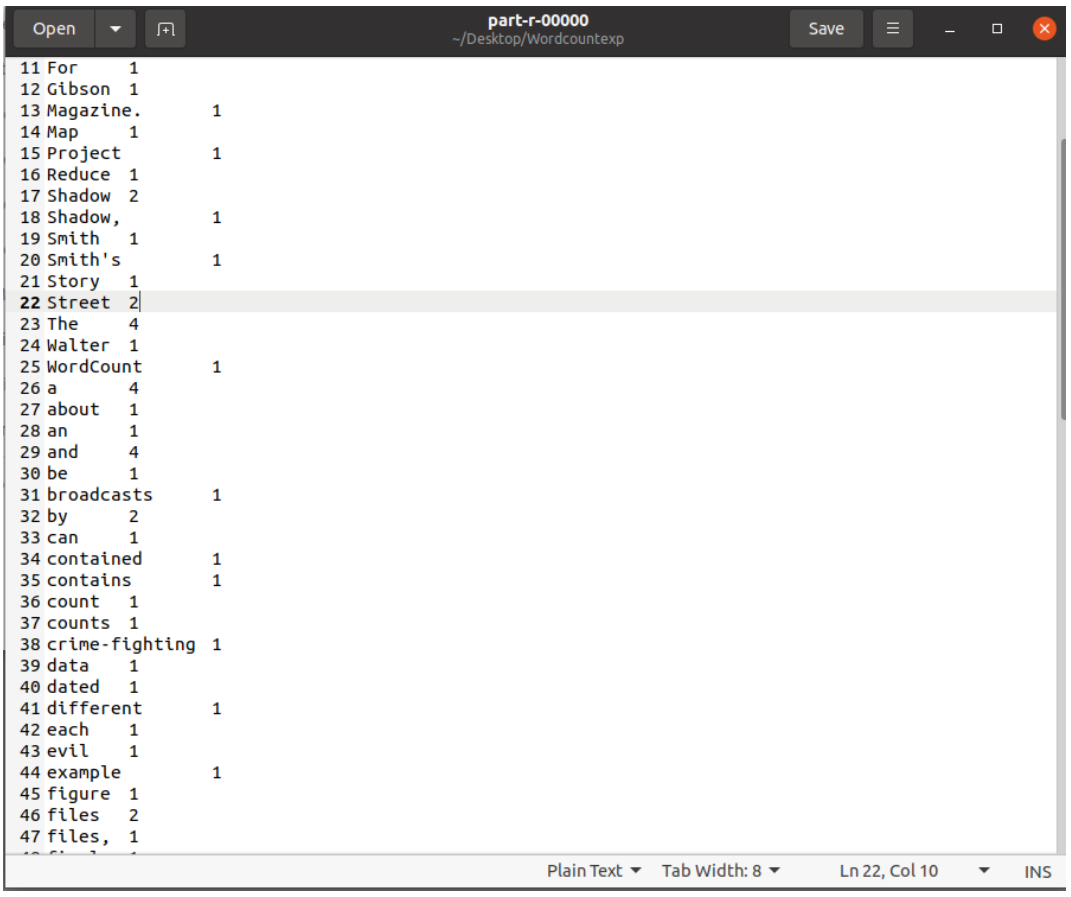
Input: input.txt

WordCount example reads text files and counts how often words occur. The input is text files, and the output is text files, each line of which contains a word and the count of how often it occurred, separated by a tab.

Map Reduce Project that works on weather data and process it, the final outcome of the project can be processed further to find similarities on different weather stations

The Shadow was an American pulp magazine published by Street & Smith from 1931 to 1949. Each issue contained a novel about The Shadow, a mysterious crime-fighting figure who spoke the line "Who knows what evil lurks in the hearts of men? The Shadow knows" in radio broadcasts of stories from Street & Smith's Detective Story Magazine. For the first issue, dated April 1931, Walter Gibson wrote the lead novel,

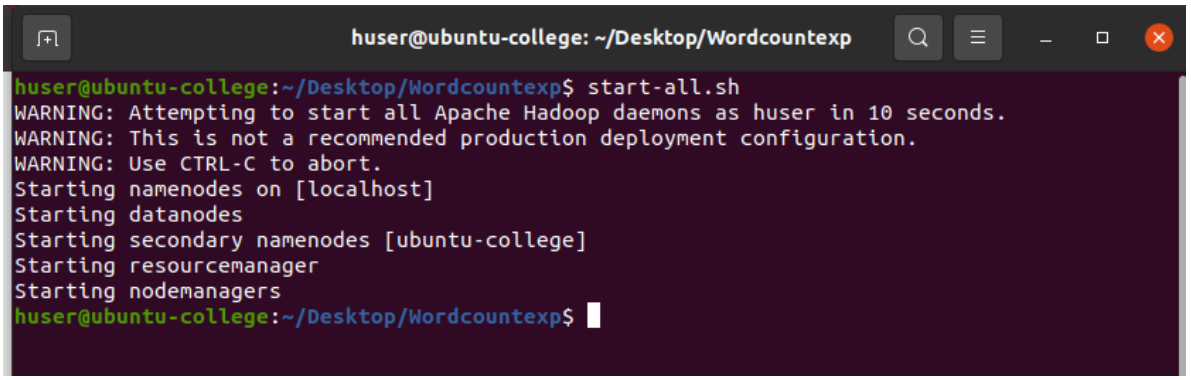
Output: output file (part-r-00000)



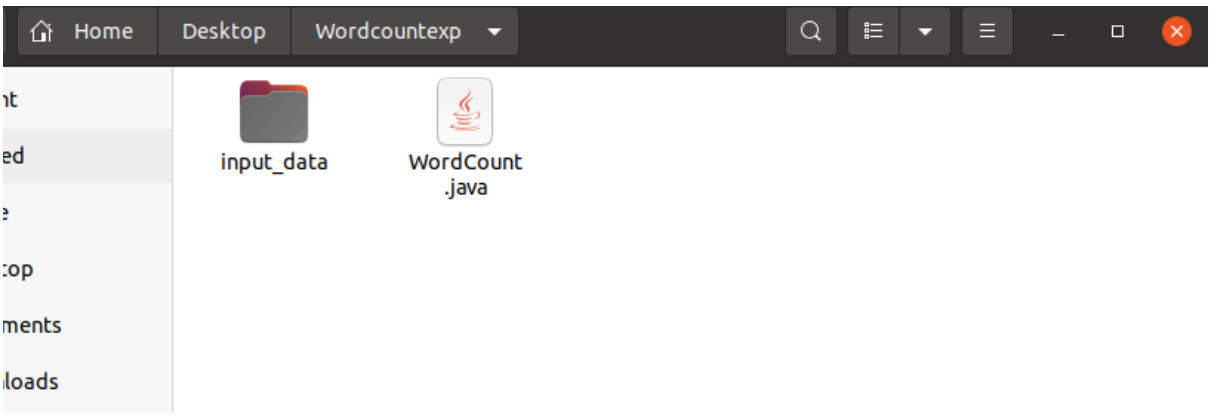
Wordcount Steps to run:

1. Starting Hadoop

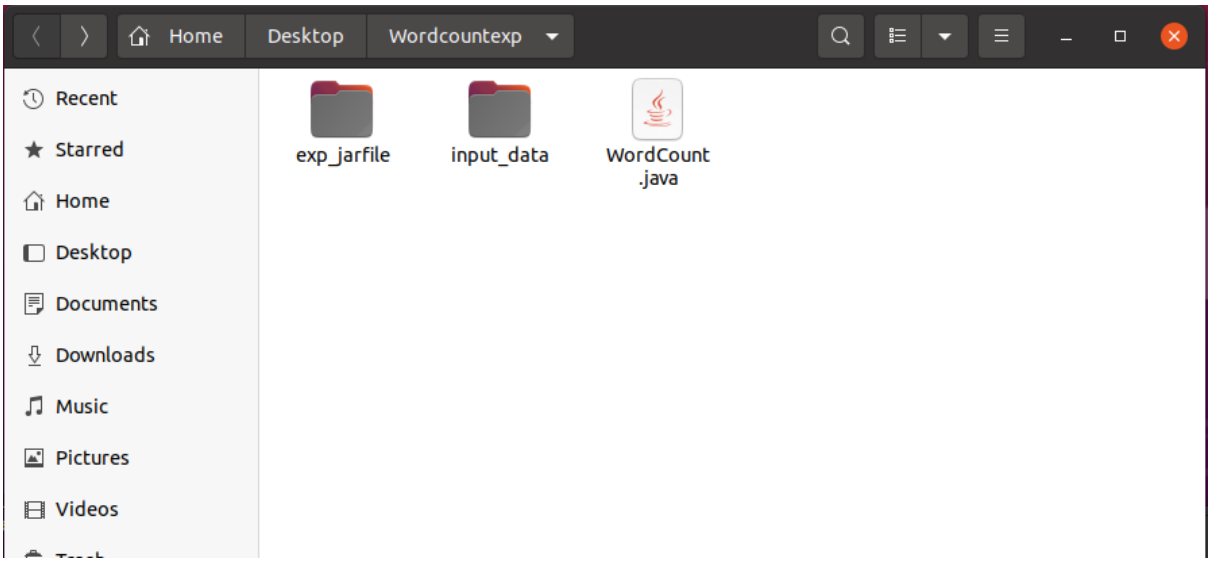
   start-all.sh



2. Made A folder “wordcountexp” and write WordCount.java code.
3. Create new folder for input data.



4. Add input text file in the input data folder.
5. Create new folder to hold java class files.



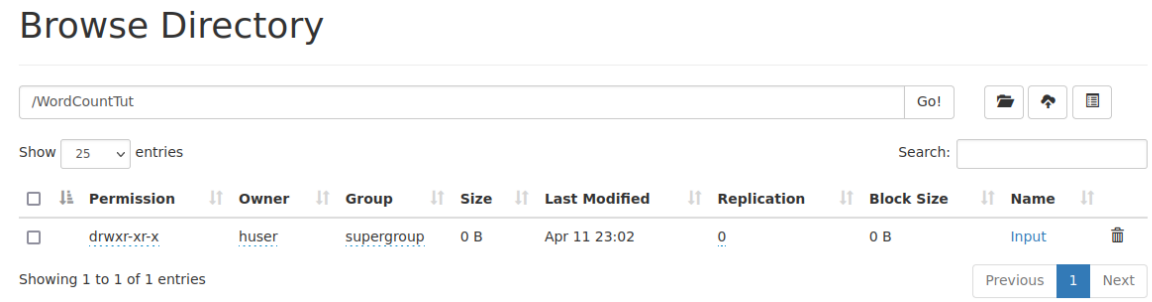
6. Set HADOOP_CLASSPATH environment variable.

```
export HADOOP_CLASSPATH=$(hadoop classpath)
```

7. Create a directory on HDFS

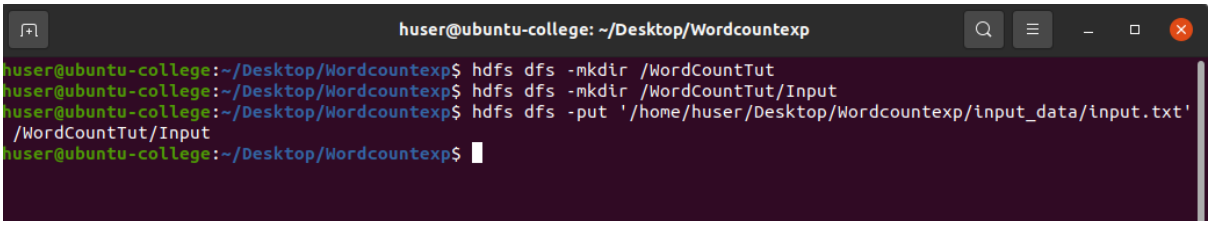
```
hdfs dfs -mkdir /WordCountTut
hdfs dfs -mkdir /WordCountTut/Input
```

8. Checking on localhost:9870



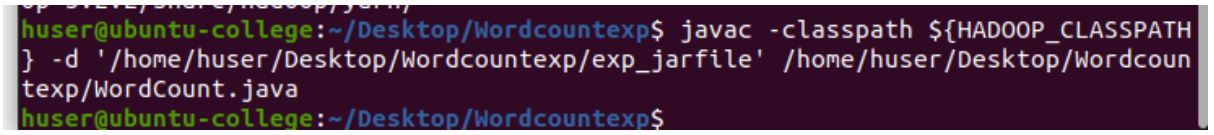
9. Upload the input file (device) to that directory.

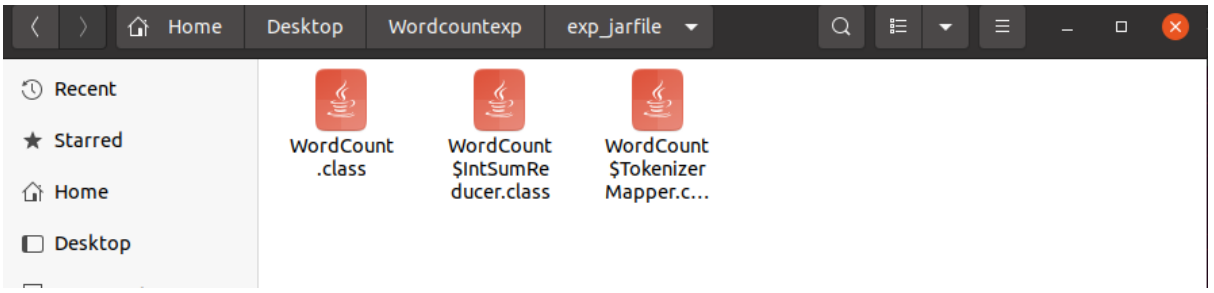
hdfs dfs -put <Input file > <hdfs input dir>



10. Compile the java code:

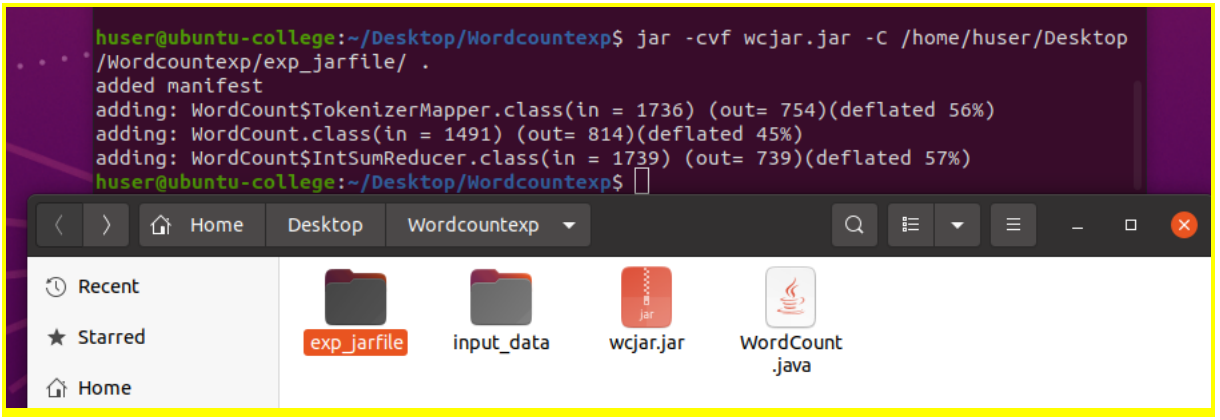
javac -classpath $(HADOOP_CLASSPATH) -d <Classes_folder> <java file>





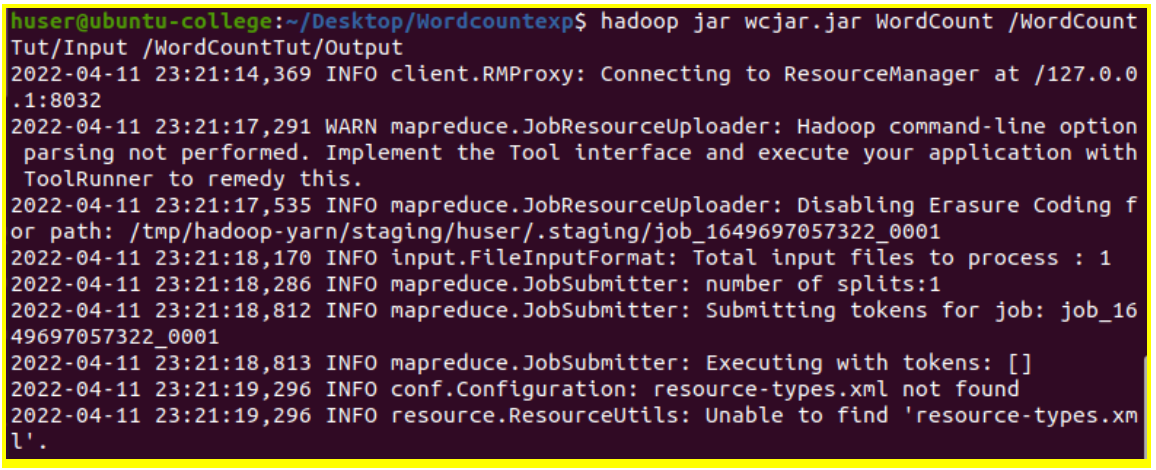
11. Creation .jar file of classes:

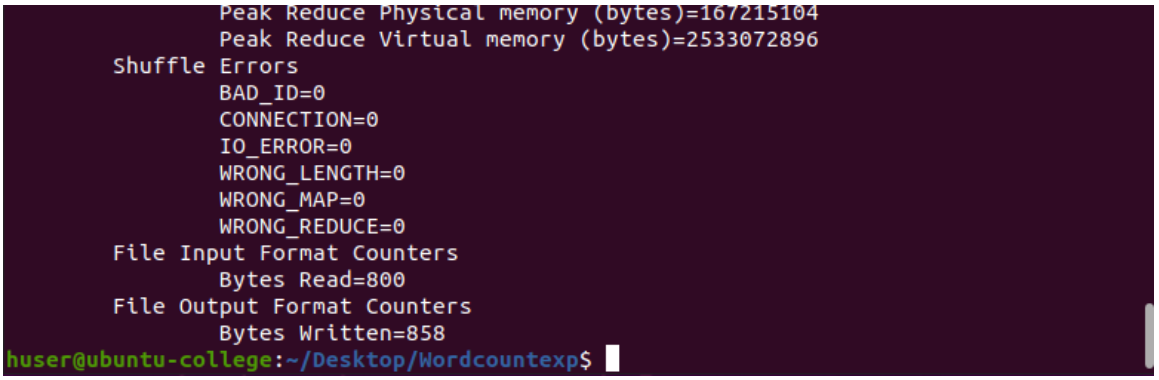
jar -cvf <jar file name > -C <classes folder> .



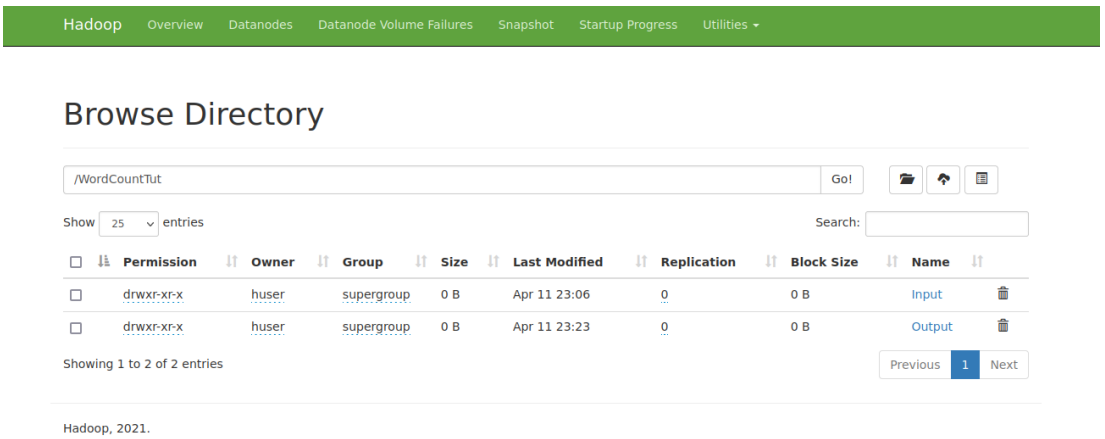
12.Running the jar file on Hadoop

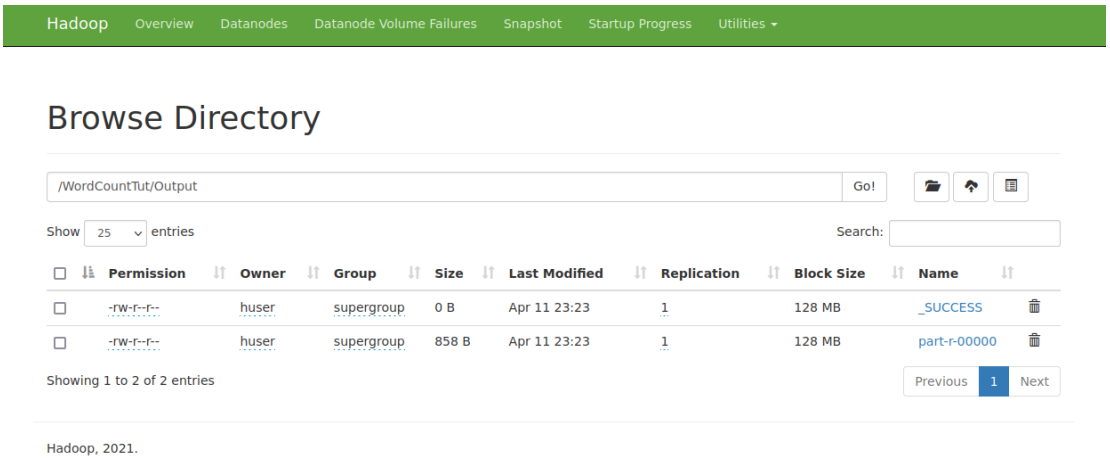
hadoop jar <jar file> <class name> <hdfs input dir> <hdfs output dir>

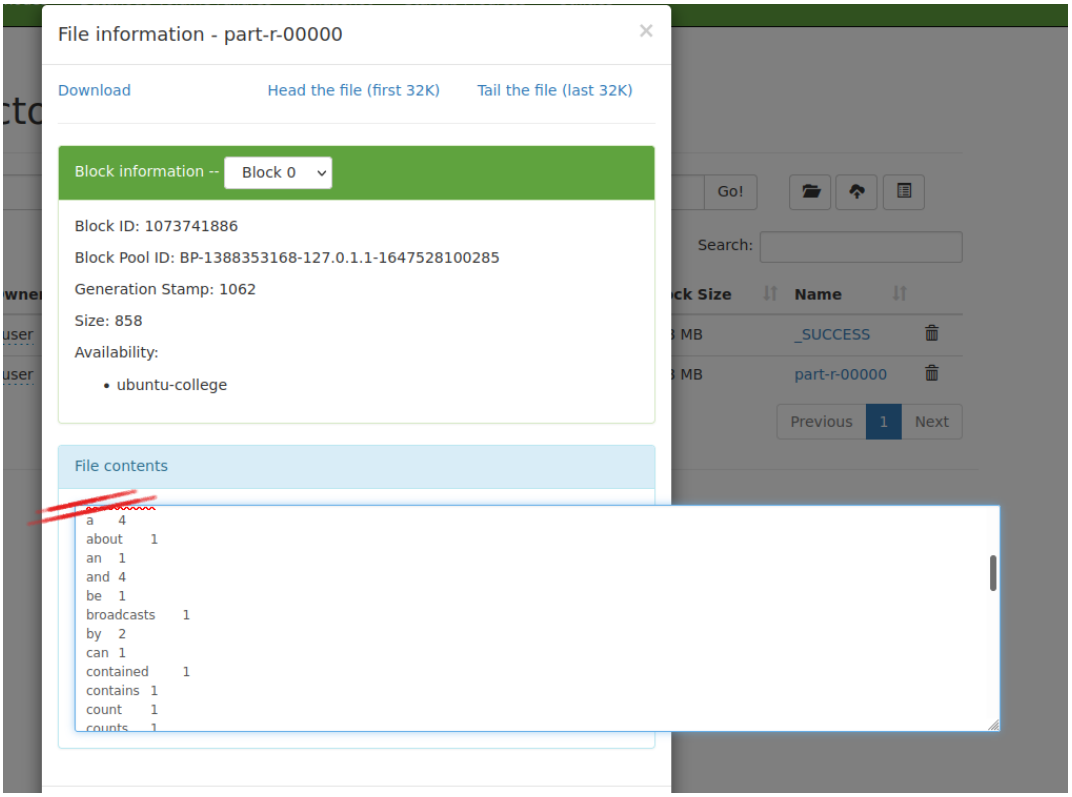




13.Check output on localhost:9870 /localhost:50070


Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Browse Directory
/WordCountTut
Go!
Show 25 entries
Search:
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
drwxr-xr-x huser supergroup 0 B Apr 11 23:06 0 0 B Input
drwxr-xr-x huser supergroup 0 B Apr 11 23:23 0 0 B Output
Showing 1 to 2 of 2 entries
Previous
1
Next
Hadoop, 2021.


Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Browse Directory
/WordCountTut/Output
Go!
Show 25 entries
Search:
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
-rw-r--r-- huser supergroup 0 B Apr 11 23:23 1 128 MB _SUCCESS
-rw-r--r-- huser supergroup 858 B Apr 11 23:23 1 128 MB part-r-00000
Showing 1 to 2 of 2 entries
Previous
1
Next
Hadoop, 2021.


File information - part-r-00000
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information --
Block 0
Block ID: 1073741886
Block Pool ID: BP-1388353168-127.0.1.1-1647528100285
Generation Stamp: 1062
Size: 858
Availability:
ubuntu-college
File contents
a 4
about 1
an 1
and 4
be 1
broadcasts 1
by 2
can 1
contained 1
contains 1
count 1
counts 1
Go!
Search:
Name
_SUCCESS
part-r-00000
Previous
1
Next

Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

Logs File Analysis Hadoop

Code:

1> LogFileMapper.java (Use for mapping the IP addresses from input csv file)

```
package LogFileCountry;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.*;


public class LogFileMapper extends MapReduceBase implements Mapper<LongWritable,
Text, Text, IntWritable> {

	private final static IntWritable one = new IntWritable(1);


	public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable>
output, Reporter reporter) throws IOException {


			String valueString = value.toString();

			String[] SingleIpData = valueString.split("-");

			output.collect(new Text(SingleIpData[0]), one);

	}

}
```

2>LogFileReduce.java (Use for reducing data received from mapper process to final output)

```
package LogFileCountry;

import java.io.IOException;

import java.util.*;


import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.*;


public class LogFileReducer extends MapReduceBase implements Reducer<Text, IntWritable,
Text, IntWritable> {

	public void reduce(Text t_key, Iterator<IntWritable> values,
OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException {

		Text key = t_key;

		int frequencyForIp = 0;

		while (values.hasNext()) {

			// replace type of value with the actual type of our value

			IntWritable value = (IntWritable) values.next();

			frequencyForIp += value.get();

		}

		output.collect(key, new IntWritable(frequencyForIp));

	}

}
```

3>LogFileCountryDriver.java (The driver code to run map-reduce on hdfs)

```
package LogFileCountry;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapred.*;


public class LogFileCountryDriver {
```

```
public static void main(String[] args) {
        JobClient my_client = new JobClient();
        // Create a configuration object for the job
        JobConf job_conf = new JobConf(LogFileCountryDriver.class);

        // Set a name of the Job
        job_conf.setJobName("LogFileIP");
        // Specify data type of output key and value
        job_conf.setOutputKeyClass(Text.class);
        job_conf.setOutputValueClass(IntWritable.class);
        // Specify names of Mapper and Reducer Class
        job_conf.setMapperClass(LogFileCountry.LogFileMapper.class);
        job_conf.setReducerClass(LogFileCountry.LogFileReducer.class);

        // Specify formats of the data type of Input and output
        job_conf.setInputFormat(TextInputFormat.class);
        job_conf.setOutputFormat(TextOutputFormat.class);

        // Set input and output directories using command line arguments,
        //arg[0] = name of input directory on HDFS, and arg[1] =  name of output
directory to be created to store the output file.

        FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));

        my_client.setConf(job_conf);
        try {   // Run the job
                JobClient.runJob(job_conf);
        } catch (Exception e) {
                e.printStackTrace();
```

```
            }
        }
}
```

4>log_file.txt (Input file sample)

0.223.157.186 - - [15/Jul/2009:20:50:32 -0700] "GET /assets/js/the-associates.js HTTP/1.1" 304 -

10.223.157.186 - - [15/Jul/2009:20:50:33 -0700] "GET /assets/img/home-logo.png HTTP/1.1" 304 -

10.223.157.186 - - [15/Jul/2009:20:50:33 -0700] "GET /assets/img/dummy/primary-news-2.jpg HTTP/1.1" 304 -

10.223.157.186 - - [15/Jul/2009:20:50:33 -0700] "GET /assets/img/dummy/primary-news-1.jpg HTTP1.1" 304 -

10.223.157.186 - - [15/Jul/2009:20:50:33 -0700] "GET /assets/img/home-media-block-placeholder.jpg HTTP/1.1" 304 -

10.223.157.186 - - [15/Jul/2009:20:50:33 -0700] "GET /assets/img/dummy/secondary-news-4.jpg HTTP/1.1" 304 -

10.223.157.186 - - [15/Jul/2009:20:50:33 -0700] "GET /assets/img/loading.gif HTTP/1.1" 304 -

10.223.157.186 - - [15/Jul/2009:20:50:33 -0700] "GET /assets/img/search-button.gif HTTP/1.1" 304 –

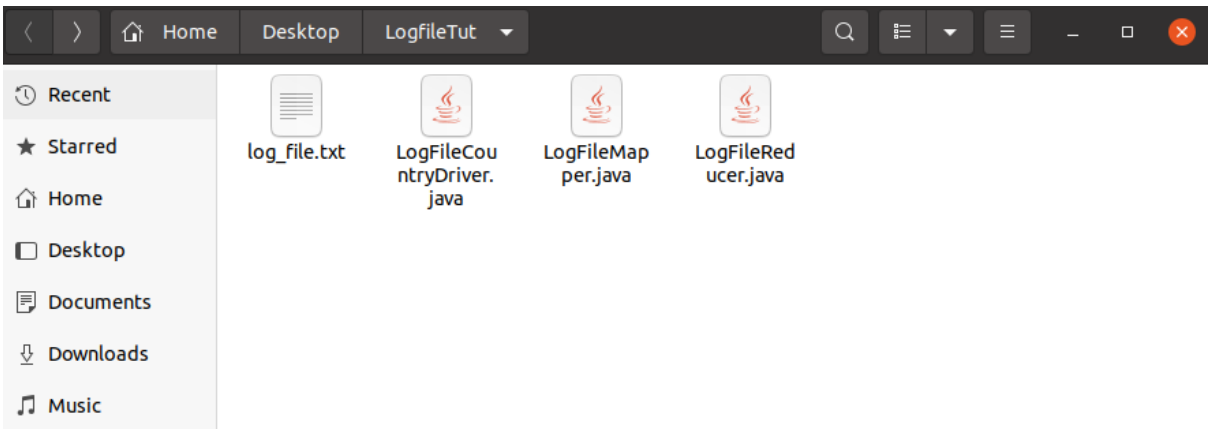
Step For Logs File Code:

1) Starting Hadoop and check if it is started.

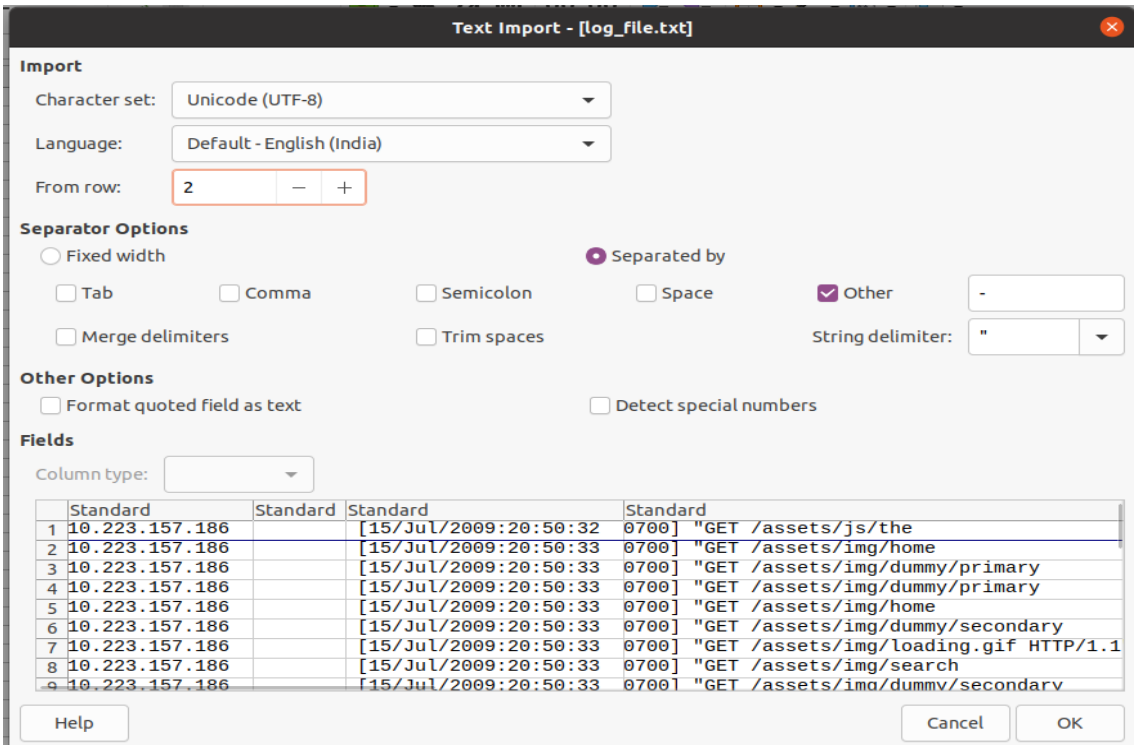
start-all.sh

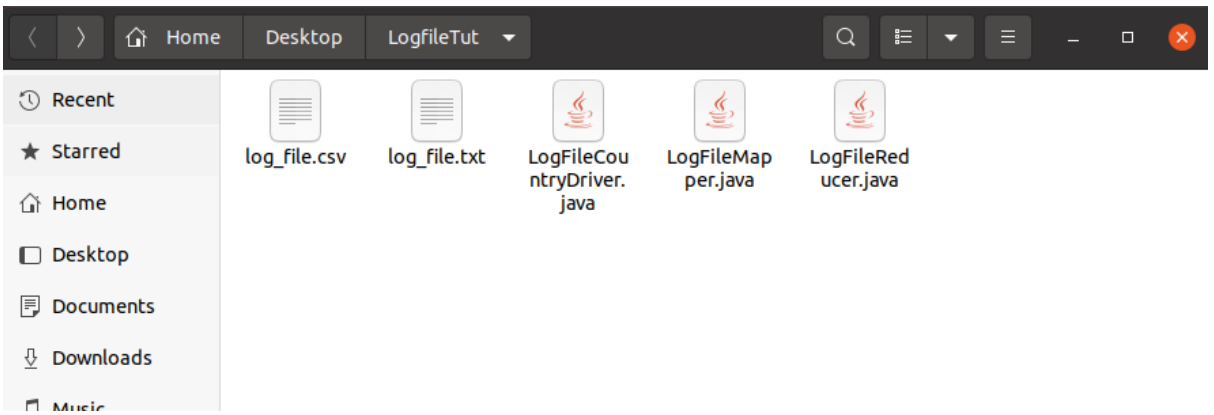
```
huser@ubuntu-college:~/Desktop/LogfileTut$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as huser in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [ubuntu-college]
Starting resourcemanager
Starting nodemanagers
huser@ubuntu-college:~/Desktop/LogfileTut$ jps
63264 ResourceManager
63030 SecondaryNameNode
63752 Jps
63401 NodeManager
62718 NameNode
62846 DataNode
```

2) Create folder “LogFileTut”. Copy the log_file.txt given and create the java files.
   i) LogFileMapper.java
   ii) LogFileReducer.java
   iii) LogFileCountryDriver.java



3) Convert the log_file.txt to .csv file. Open LibreOffice Calc-> Open -> log_file.txt. Save As .csv in the LogFileTut folder.





4) Give Read permission to all the files in directories.

sudo chmod +r *.*

5) Set HADOOP_CLASSPATH environment variable.

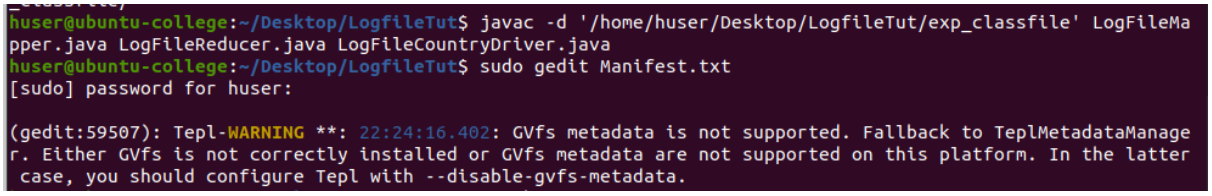
export HADOOP_CLASSPATH=$(hadoop classpath) or

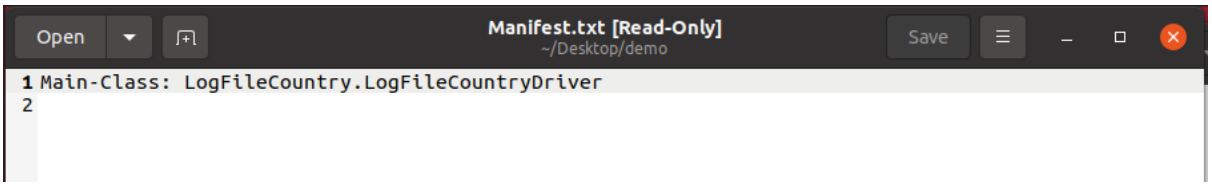
export CLASSPATH= "$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.2.2.jar: $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-common-3.2.2.jar: $HADOOP_HOME/share/hadoop/common/hadoop-common-3.2.2.jar: $HADOOP_HOME/lib/*: ~/home/huser/Desktop/LogFileTut/*"

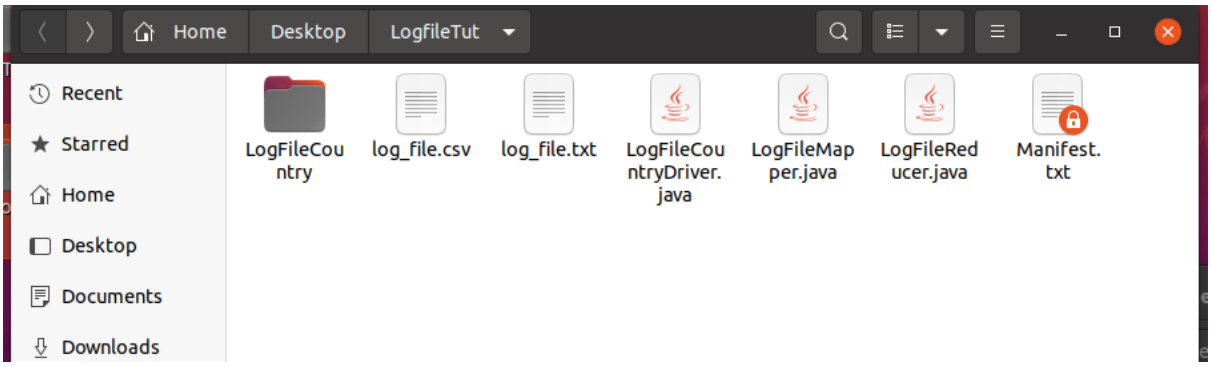
6) Compile the java code:

javac -classpath $(HADOOP_CLASSPATH) -d . <java file (3 files)>

7) Create Manifest.txt file.

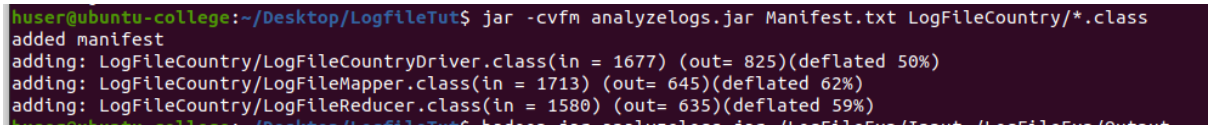


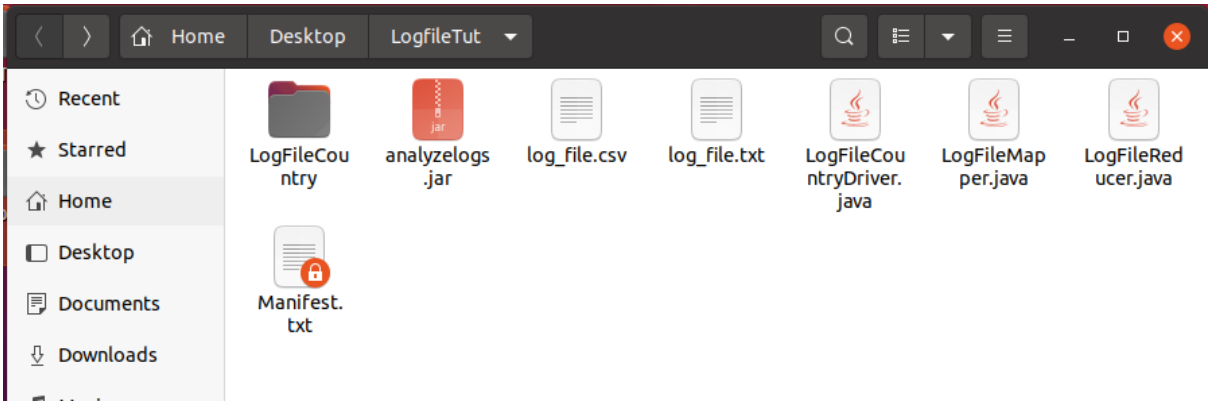




8) Creation .jar file of classes:

jar -cvfm <jar file name > Manifest.txt <classes folder/*.class>



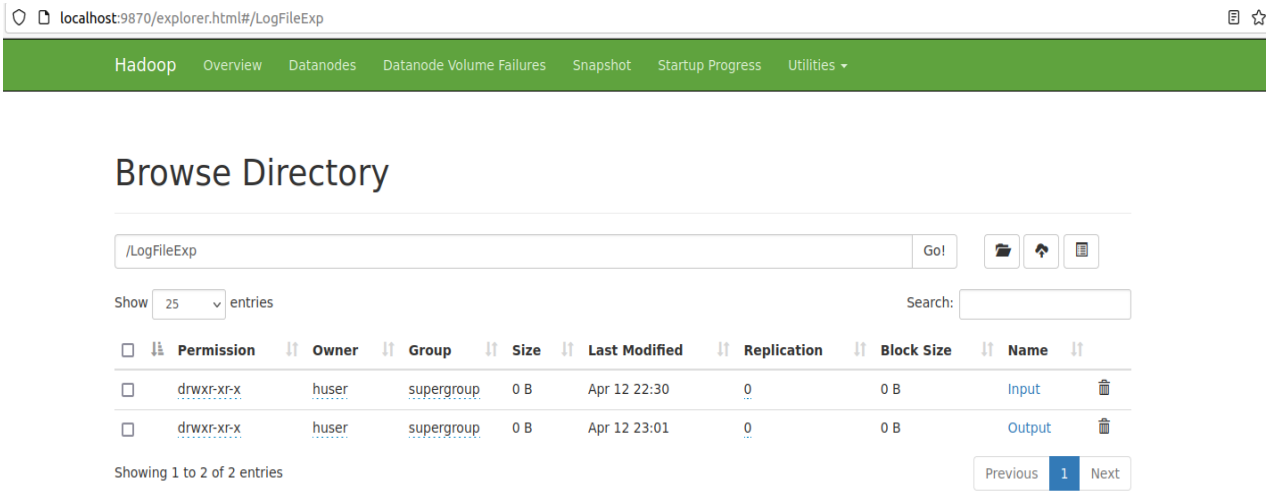


9) Create a directory on HDFS .And check on localhost:9870

hdfs dfs -mkdir / LogFileExp

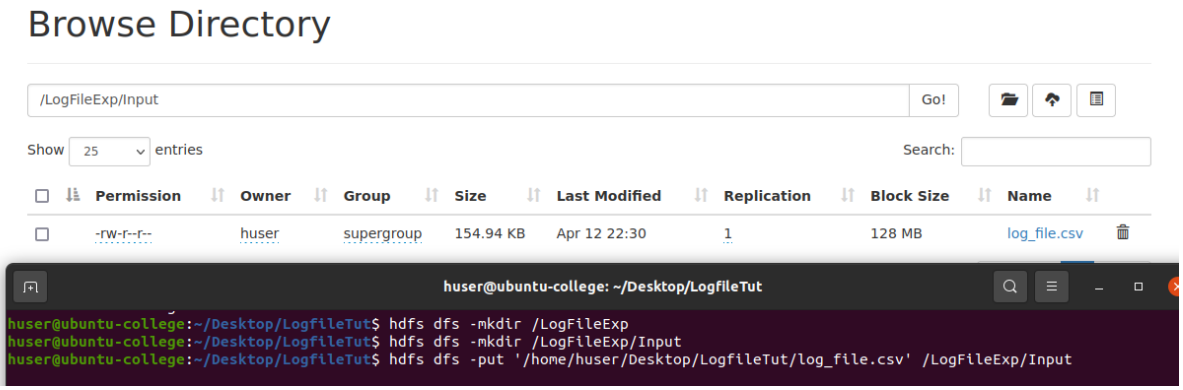
hdfs dfs -mkdir / LogFileExp/Input

hdfs dfs -mkdir / LogFileExp/Output



10) Upload the log_file.csv in hadoop dir /LogFileExp/Input

hdfs dfs -put <Input file > <hdfs input dir>



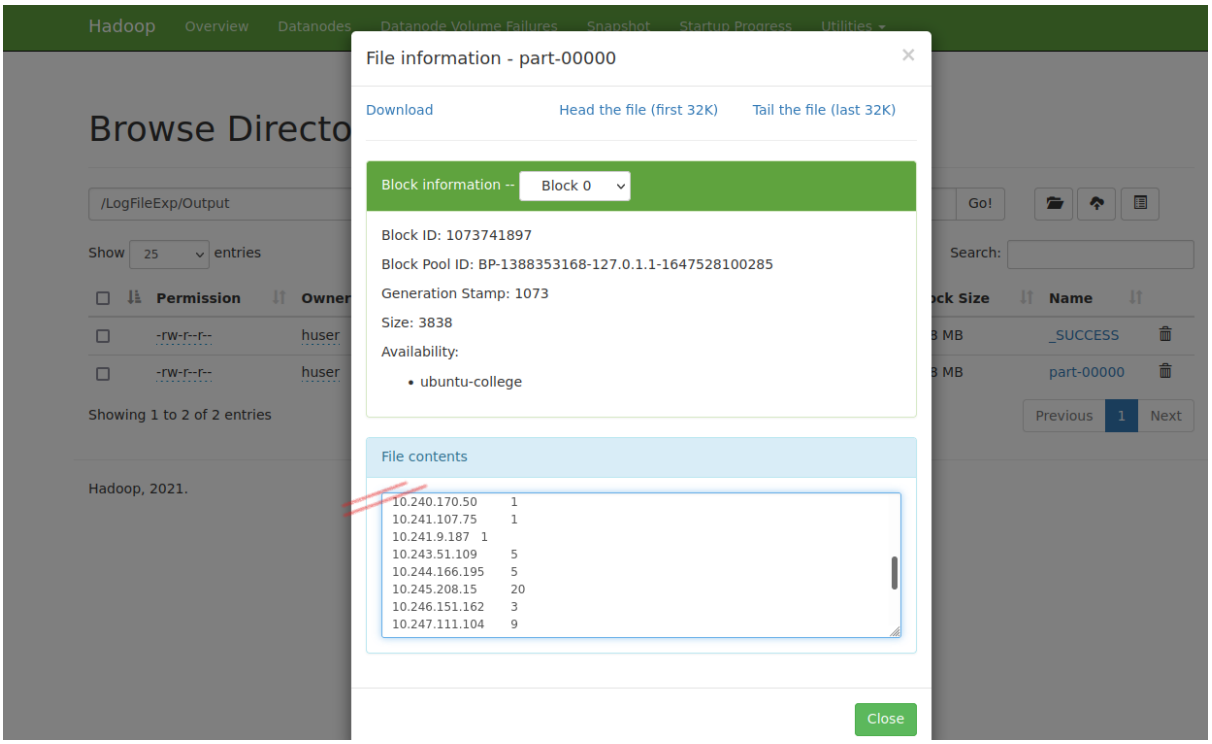
11) Running the jar file on Hadoop.

hadoop jar <jar file> <class name> <hdfs input dir> <hdfs output dir>

```
huser@ubuntu-college:~/Desktop/LogfileTut$ hadoop jar analyzelogs.jar /LogFileExp/Input /LogFileExp/Output
2022-04-12 22:51:25,988 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0.1:8032
2022-04-12 22:51:27,403 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0.1:8032
2022-04-12 22:51:35,208 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement th
e Tool interface and execute your application with ToolRunner to remedy this.
2022-04-12 22:51:36,276 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hu
ser/.staging/job_1649777619248_0001
2022-04-12 22:51:39,085 INFO mapred.FileInputFormat: Total input files to process : 1
2022-04-12 22:51:40,281 INFO mapreduce.JobSubmitter: number of splits:2
2022-04-12 22:51:40,821 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1649777619248_0001
2022-04-12 22:51:40,823 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-04-12 22:51:42,337 INFO conf.Configuration: resource-types.xml not found
2022-04-12 22:51:42,337 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-04-12 22:52:55,956 INFO impl.YarnClientImpl: Submitted application application_1649777619248_0001
2022-04-12 22:52:58,540 INFO mapreduce.Job: The url to track the job: http://ubuntu-college:8088/proxy/application_1649777
619248_0001/
2022-04-12 22:52:58,584 INFO mapreduce.Job: Running job: job_1649777619248_0001
2022-04-12 22:57:02,946 INFO mapreduce.Job: Job job_1649777619248_0001 running in uber mode : false
2022-04-12 22:57:03,272 INFO mapreduce.Job:  map 0% reduce 0%
2022-04-12 23:00:09,974 INFO mapreduce.Job:  map 83% reduce 0%
2022-04-12 23:00:27,106 INFO mapreduce.Job:  map 100% reduce 0%
2022-04-12 23:01:05,301 INFO mapreduce.Job:  map 100% reduce 100%
2022-04-12 23:01:08,520 INFO mapreduce.Job: Job job_1649777619248_0001 completed successfully
2022-04-12 23:01:24,647 INFO mapreduce.Job: Counters: 54
```

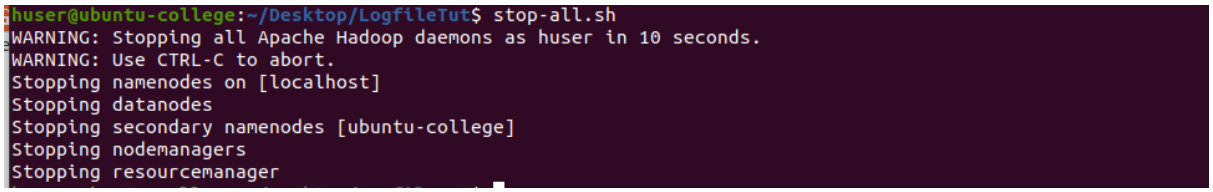
12) Check the Output file.

```
huser@ubuntu-college:~/Desktop/LogfileTut$ hdfs dfs -cat /LogFileExp/Output/part-00000
10.1.1.236      7
10.1.181.142    14
10.1.232.31     5
10.10.55.142    14
10.102.101.66   1
10.103.184.104  1
10.103.190.81   53
10.103.63.29    1
10.104.73.51    1
10.105.160.183  1
10.108.91.151   1
10.109.21.76    1
10.11.131.40    1
10.111.71.20    8
```



13) Stop all processes :

stop-all.sh


huser@ubuntu-college:~/Desktop/LogfileTut$ stop-all.sh
WARNING: Stopping all Apache Hadoop daemons as huser in 10 seconds.
WARNING: Use CTRL-C to abort.
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [ubuntu-college]
Stopping nodemanagers
Stopping resourcemanager

Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

Weather Analyse (average Temperature, dew point, wind speed)

Weather.java

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.KeyValueTextInputFormat;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

/**
```

```
* This is an Hadoop Map/Reduce application for Working on weather data It reads
* the text input files, breaks each line into stations weather data and finds
* average for temperature , dew point , wind speed. The output is a locally
* sorted list of stations and its 12 attribute vector of average temp , dew ,
* wind speed of 4 sections for each month.
*
* To run: bin/hadoop jar target/weather-1.0.jar [-m <i>maps</i>] [-r
* <i>reduces</i>] <i>in-dir for job 1</i> <i>out-dir for job 1</i> <i>out-dir
* for job 2</i>
*/
public class Weather extends Configured implements Tool {
        final long DEFAULT_SPLIT_SIZE = 128 * 1024 * 1024;

        /**
         * Map Class for Job 1
         *
         * For each line of input, emits key value pair with
         * station_yearmonth_sectionno as key and 3 attribute vector with
         * temperature , dew point , wind speed as value.Map method will strip the
         * day and hour from field and replace it with section no (
         * <b>station_yearmonth_sectionno</b>, <b><temperature,dew point , wind
         * speed></b>).
         */
        public static class MapClass extends MapReduceBase
                        implements Mapper<LongWritable, Text, Text, Text> {

                private Text word = new Text();
                private Text values = new Text();

                public void map(LongWritable key, Text value,
                                                OutputCollector<Text, Text> output,
```

```
                                        Reporter reporter) throws IOException {
                String line = value.toString();
                StringTokenizer itr = new StringTokenizer(line);
                int counter = 0;
                String key_out = null;
                String value_str = null;
                boolean skip = false;
                loop:while (itr.hasMoreTokens() && counter<13) {
                        String str = itr.nextToken();
                        switch (counter) {
                        case 0:
                                key_out = str;
                                if(str.contains("STN")){//Ignoring rows where station id is all
9
                                        skip = true;
                                        break loop;
                                }else{
                                        break;
                                }
                        case 2:
                                int                     hour                    =
Integer.valueOf(str.substring(str.lastIndexOf("_")+1, str.length()));
                                str = str.substring(4,str.lastIndexOf("_")-2);
                                /*if(hour<=5){
                                        str = str.concat("_section4");
                                }else if(hour>5 && hour<=11){
                                        str = str.concat("_section1");
                                }else if(hour>11 && hour<=17){
                                        str = str.concat("_section2");
                                }else if(hour>17 && hour<=23){
                                        str = str.concat("_section3");
                                }*/
```

```
                if(hour>4 && hour<=10){
                        str = str.concat("_section1");
                }else if(hour>10 && hour<=16){
                        str = str.concat("_section2");
                }else if(hour>16 && hour<=22){
                        str = str.concat("_section3");
                }else{
                        str = str.concat("_section4");
                }


                key_out = key_out.concat("_").concat(str);
                break;
        case 3://Temperature
                if(str.equals("9999.9")){//Ignoring rows where temperature
is all 9
                        skip = true;
                        break loop;
                }else{
                        value_str = str.concat(" ");
                        break;
                }
        case 4://Dew point
                if(str.equals("9999.9")){//Ignoring rows where dew point is
all 9
                        skip = true;
                        break loop;
                }else{
                        value_str = value_str.concat(str).concat(" ");
                        break;
                }
```

```
                                case 12://Wind speed
                                        if(str.equals("999.9")){//Ignoring rows where wind speed is
all 9
                                                skip = true;
                                                break loop;
                                        }else{
                                                value_str = value_str.concat(str).concat(" ");
                                                break;
                                        }
                                default:
                                        break;
                                }
                                counter++;
                        }
                        if(!skip){
                                word.set(key_out);
                                values.set(value_str);
                                output.collect(word, values);
                        }
                }
        }
        /**
         * Reducer Class for Job 1
         *
         * A reducer class that just emits 3 attribute vector with average
         * temperature , dew point , wind speed for each of the section of the month
         * for each input
         */
        public static class Reduce extends MapReduceBase
                        implements Reducer<Text, Text, Text, Text> {
                private Text value_out_text = new Text();
```

```
        public void reduce(Text key, Iterator<Text> values,
                        OutputCollector<Text, Text> output, Reporter reporter) throws
IOException {
                double sum_temp = 0;
                double sum_dew = 0;
                double sum_wind = 0;
                int count = 0;

                while (values.hasNext()) {

                        String str = values.next().toString();

                        StringTokenizer itr = new StringTokenizer(str);
                        int count_vector = 0;
                        while (itr.hasMoreTokens()) {
                                String nextToken = itr.nextToken(" ");
                                if(count_vector==0){
                                        sum_temp += Double.valueOf(nextToken);
                                }
                                if(count_vector==1){
                                        sum_dew += Double.valueOf(nextToken);
                                }
                                if(count_vector==2){
                                        sum_wind += Double.valueOf(nextToken);
                                }
                                count_vector++;
                        }
                        count++;
                }
                double avg_tmp = sum_temp / count;
                double avg_dew = sum_dew / count;
```

```
                double avg_wind = sum_wind / count;
                System.out.println(key.toString()+" count is "+count+" sum of temp is "+sum_temp+" sum of dew is "+sum_dew+" sum of wind is "+sum_wind+"\n");
                String value_out = String.valueOf(avg_tmp).concat(" ").concat(String.valueOf(avg_dew)).concat(" ").concat(String.valueOf(avg_wind));
                value_out_text.set(value_out);
                output.collect(key, value_out_text);
            }
        }
        static int printUsage() {
            System.out.println("weather [-m <maps>] [-r <reduces>] <job_1 input> <job_1 output> <job_2 output>");
            ToolRunner.printGenericCommandUsage(System.out);
            return -1;
        }
        /**
         * The main driver for weather map/reduce program.
         * Invoke this method to submit the map/reduce job.
         * @throws IOException When there is communication problems with the
         *                     job tracker.
         */
        public int run(String[] args) throws Exception {
            Configuration config = getConf();

            // We need to lower input block size by factor of two.

            JobConf conf = new JobConf(config, Weather.class);
            conf.setJobName("Weather Job1");

            // the keys are words (strings)
            conf.setOutputKeyClass(Text.class);
            // the values are counts (ints)
```

```
conf.setOutputValueClass(Text.class);

conf.setMapOutputKeyClass(Text.class);
conf.setMapOutputValueClass(Text.class);

conf.setMapperClass(MapClass.class);
//conf.setCombinerClass(Combiner.class);
conf.setReducerClass(Reduce.class);
List<String> other_args = new ArrayList<String>();
for(int i=0; i < args.length; ++i) {
        try {
                if ("-m".equals(args[i])) {
                        conf.setNumMapTasks(Integer.parseInt(args[++i]));
                } else if ("-r".equals(args[i])) {
                        conf.setNumReduceTasks(Integer.parseInt(args[++i]));
                } else {
                        other_args.add(args[i]);
                }
        } catch (NumberFormatException except) {
                System.out.println("ERROR: Integer expected instead of " + args[i]);
                return printUsage();
        } catch (ArrayIndexOutOfBoundsException except) {
                System.out.println("ERROR: Required parameter missing from " +
                        args[i-1]);
                return printUsage();
        }
}
// Make sure there are exactly 2 parameters left.
FileInputFormat.setInputPaths(conf, other_args.get(0));
FileOutputFormat.setOutputPath(conf, new Path(other_args.get(1)));
JobClient.runJob(conf);
```

```
        return 0;
    }
    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new Weather(), args);
        System.exit(res);
    }
}
```

Input: sample_weather.txt (sample)

```
690190 13910  20060201_0 51.75   33.0 24  1006.3 24  943.9 24  15.0 24  10.7 24  22.0  28.9   0.00I 999.9  000000
690190 13910  20060201_1 54.74   33.0 24  1006.3 24  943.9 24  15.0 24  10.7 24  22.0  28.9   0.00I 999.9  00000
690190 13910  20060201_2 50.59   33.0 24  1006.3 24  943.9 24  15.0 24  10.7 24  22.0  28.9   0.00I 999.9  00000
690190 13910  20060201_3 51.67   33.0 24  1006.3 24  943.9 24  15.0 24  10.7 24  22.0  28.9   0.00I 999.9  000000
690190 13910  20060201_4 65.67   33.0 24  1006.3 24  943.9 24  15.0 24  10.7 24  22.0  28.9   0.00I 999.9  000000
690190 13910  20060201_5 55.37   33.0 24  1006.3 24  943.9 24  15.0 24  10.7 24  22.0  28.9   0.00I 999.9  000000
690190 13910  20060201_6 49.26   33.0 24  1006.3 24  943.9 24  15.0 24  10.7 24  22.0  28.9   0.00I 999.9  000000
690190 13910  20060201_7 55.44   33.0 24  1006.3 24  943.9 24  15.0 24  10.7 24  22.0  28.9   0.00I 999.9  000000
690190 13910  20060201_8 64.05   33.0 24  1006.3 24  943.9 24  15.0 24  10.7 24  22.0  28.9   0.00I 999.9  000000
```

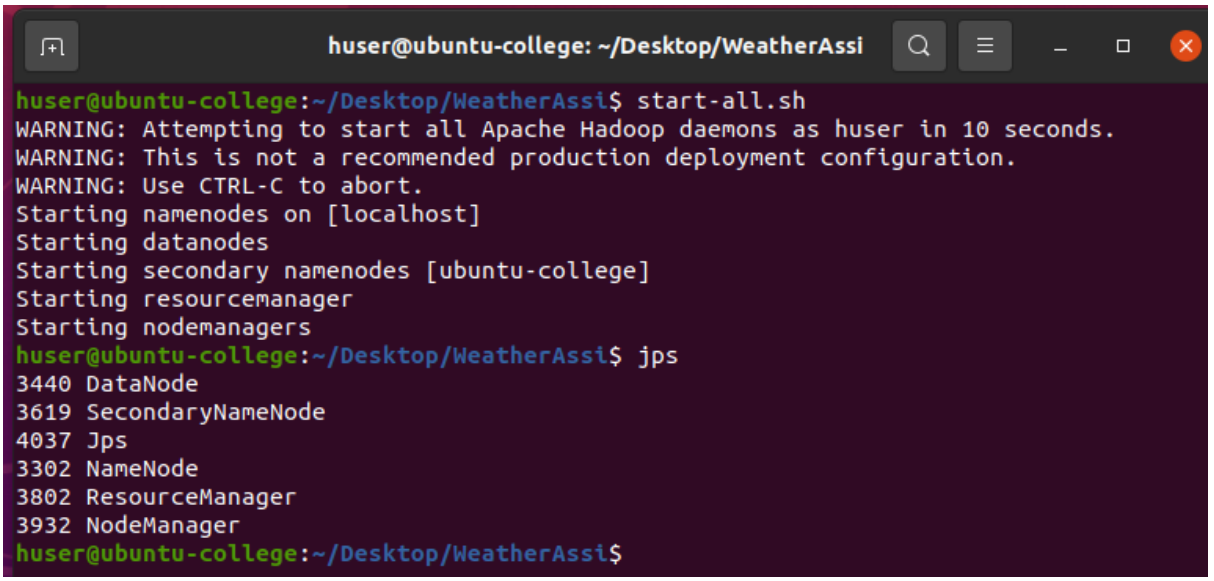
Output: part-00000.txt (on Hadoop)

```
690190_02_section1  53.87166666666666 25.899999999999995 7.774999999999998
690190_02_section2  54.76125000000001 25.900000000000006 7.774999999999999
690190_02_section3  53.25041666666667 25.899999999999995 7.774999999999996
690190_02_section4  52.44708333333333 25.900000000000006 7.774999999999999
```

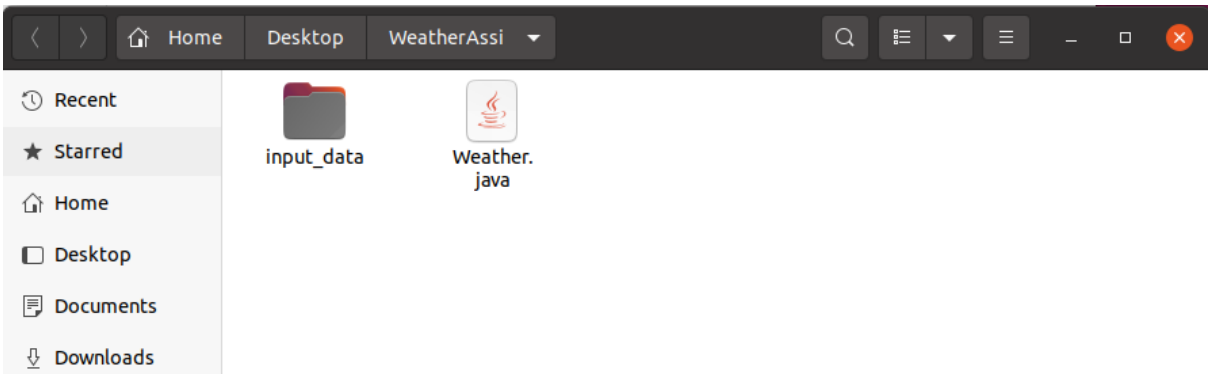
Weather Data Analysis Steps to run:

1. Starting Hadoop

   start-all.sh

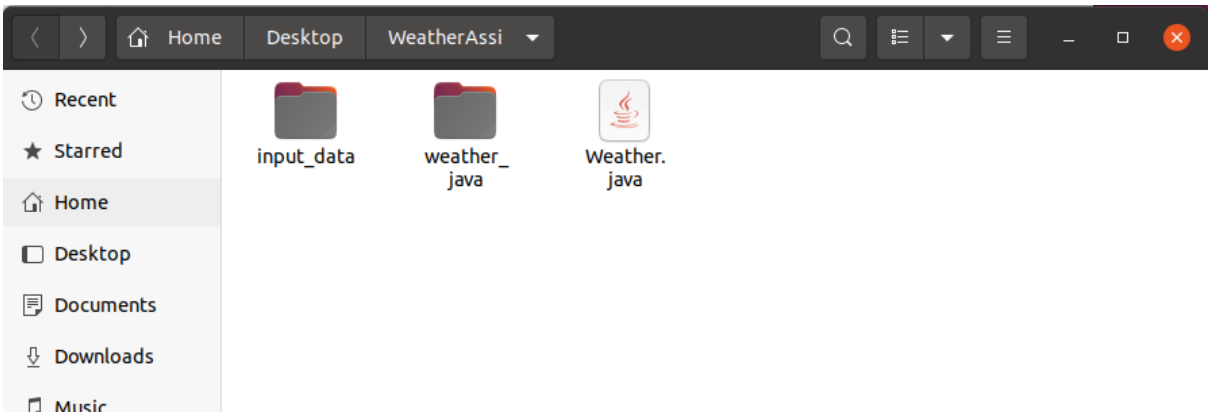


2. Made A folder “WeatherAssi” and write Weather.java code.
3. Create new folder for input data.



4. Add input text file in the input data folder.

5. Create new folder to hold java class files.



6. Set HADOOP_CLASSPATH environment variable.

   export HADOOP_CLASSPATH=$(hadoop classpath)

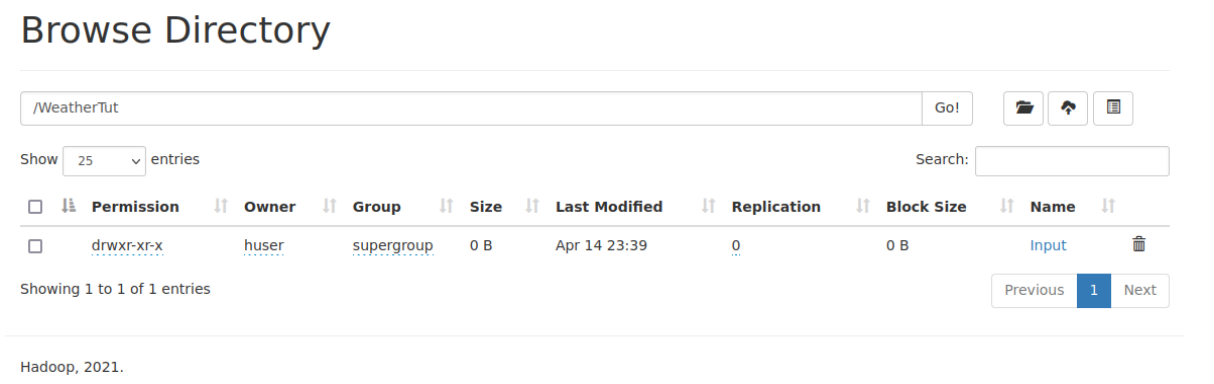
```
huser@ubuntu-college:~/Desktop/WeatherAssi$ export HADOOP_CLASSPATH=$(hadoop classpa
th)
huser@ubuntu-college:~/Desktop/WeatherAssi$ echo $HADOOP_CLASSPATH
/home/huser/hadoop/hadoop-3.2.2/etc/hadoop:/home/huser/hadoop/hadoop-3.2.2/share/had
oop/common/lib/*:/home/huser/hadoop/hadoop-3.2.2/share/hadoop/common/*:/home/huser/h
adoop/hadoop-3.2.2/share/hadoop/hdfs:/home/huser/hadoop/hadoop-3.2.2/share/hadoop/hd
fs/lib/*:/home/huser/hadoop/hadoop-3.2.2/share/hadoop/hdfs/*:/home/huser/hadoop/hado
op-3.2.2/share/hadoop/mapreduce/lib/*:/home/huser/hadoop/hadoop-3.2.2/share/hadoop/m
apreduce/*:/home/huser/hadoop/hadoop-3.2.2/share/hadoop/yarn:/home/huser/hadoop/hado
op-3.2.2/share/hadoop/yarn/lib/*:/home/huser/hadoop/hadoop-3.2.2/share/hadoop/yarn/*
huser@ubuntu-college:~/Desktop/WeatherAssi$
```

7. Create a directory on HDFS

   hdfs dfs -mkdir /WeatherTut

   hdfs dfs -mkdir /WeatherTut/Input

8. Checking on localhost:9870



9. Upload the input file (device) to that directory.

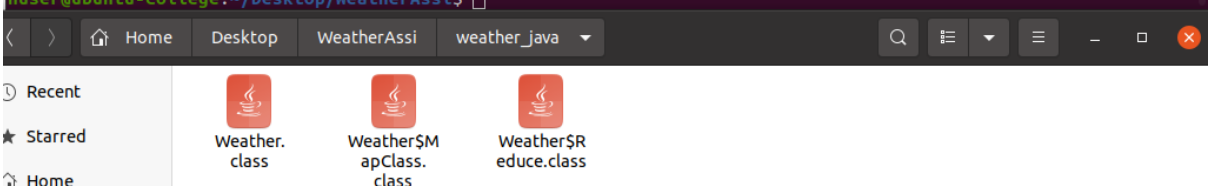
   hdfs dfs -put <Input file > <hdfs input dir>

```
huser@ubuntu-college:~/Desktop/WeatherAssi$ hdfs dfs -mkdir /WeatherTut
huser@ubuntu-college:~/Desktop/WeatherAssi$ hdfs dfs -mkdir /WeatherTut/Input
huser@ubuntu-college:~/Desktop/WeatherAssi$ hdfs dfs -put input_data/sample_weather.txt /WeatherTut/Input
huser@ubuntu-college:~/Desktop/WeatherAssi$
```

10. Compile the java code:

javac -classpath $(HADOOP_CLASSPATH) -d <Classes_folder> <java file>

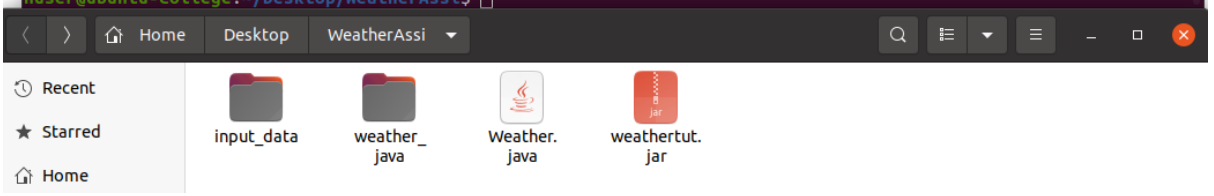
```
huser@ubuntu-college:~/Desktop/WeatherAssi$ hdfs dfs -put input_data/sample_weather.txt /WeatherTut/Input
huser@ubuntu-college:~/Desktop/WeatherAssi$ javac -classpath ${HADOOP_CLASSPATH} -d '/home/huser/Desktop/Weather
Assi/weather_java' /home/huser/Desktop/WeatherAssi/Weather.java
huser@ubuntu-college:~/Desktop/WeatherAssi$
```



11. Creation .jar file of classes:

jar -cvf <jar file name > -C <classes folder> .

```
huser@ubuntu-college:~/Desktop/WeatherAssi$ jar -cvf weathertut.jar -C /home/huser/Desktop/WeatherAssi/weather_j
ava/ .
added manifest
adding: Weather$Reduce.class(in = 2782) (out= 1285)(deflated 53%)
adding: Weather$MapClass.class(in = 2897) (out= 1378)(deflated 52%)
adding: Weather.class(in = 3267) (out= 1703)(deflated 47%)
huser@ubuntu-college:~/Desktop/WeatherAssi$
```



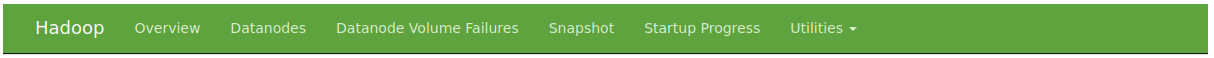
12. Running the jar file on Hadoop

hadoop jar <jar file> <class name> <hdfs input dir> <hdfs output dir>

```
huser@ubuntu-college:~/Desktop/WeatherAssi$ hadoop jar weathertut.jar Weather /WeatherTut/Input /WeatherTut/Outp
ut
2022-04-14 23:46:46,131 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0.1:8032
2022-04-14 23:46:46,859 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0.1:8032
2022-04-14 23:46:47,402 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/
staging/huser/.staging/job_1649959472953_0001
2022-04-14 23:46:47,836 INFO mapred.FileInputFormat: Total input files to process : 1
2022-04-14 23:46:47,988 INFO mapreduce.JobSubmitter: number of splits:2
2022-04-14 23:46:48,804 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1649959472953_0001
2022-04-14 23:46:48,806 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-04-14 23:46:49,499 INFO conf.Configuration: resource-types.xml not found
2022-04-14 23:46:49,500 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-04-14 23:46:50,212 INFO impl.YarnClientImpl: Submitted application application_1649959472953_0001
2022-04-14 23:46:50,314 INFO mapreduce.Job: The url to track the job: http://ubuntu-college:8088/proxy/applicati
on_1649959472953_0001/
2022-04-14 23:46:50,325 INFO mapreduce.Job: Running job: job_1649959472953_0001
```

```
                Peak Map Physical memory (bytes)=253128704
                Peak Map Virtual memory (bytes)=2524299264
                Peak Reduce Physical memory (bytes)=146530304
                Peak Reduce Virtual memory (bytes)=2532839424
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=16149
        File Output Format Counters
                Bytes Written=296
huser@ubuntu-college:~/Desktop/WeatherAssi$
```

13. Check output on localhost:9870 /localhost:50070



Browse Directory

/WeatherTut Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	huser	supergroup	0 B	Apr 14 23:40	0	0 B	Input
drwxr-xr-x	huser	supergroup	0 B	Apr 14 23:48	0	0 B	Output

Showing 1 to 2 of 2 entries Previous 1 Next

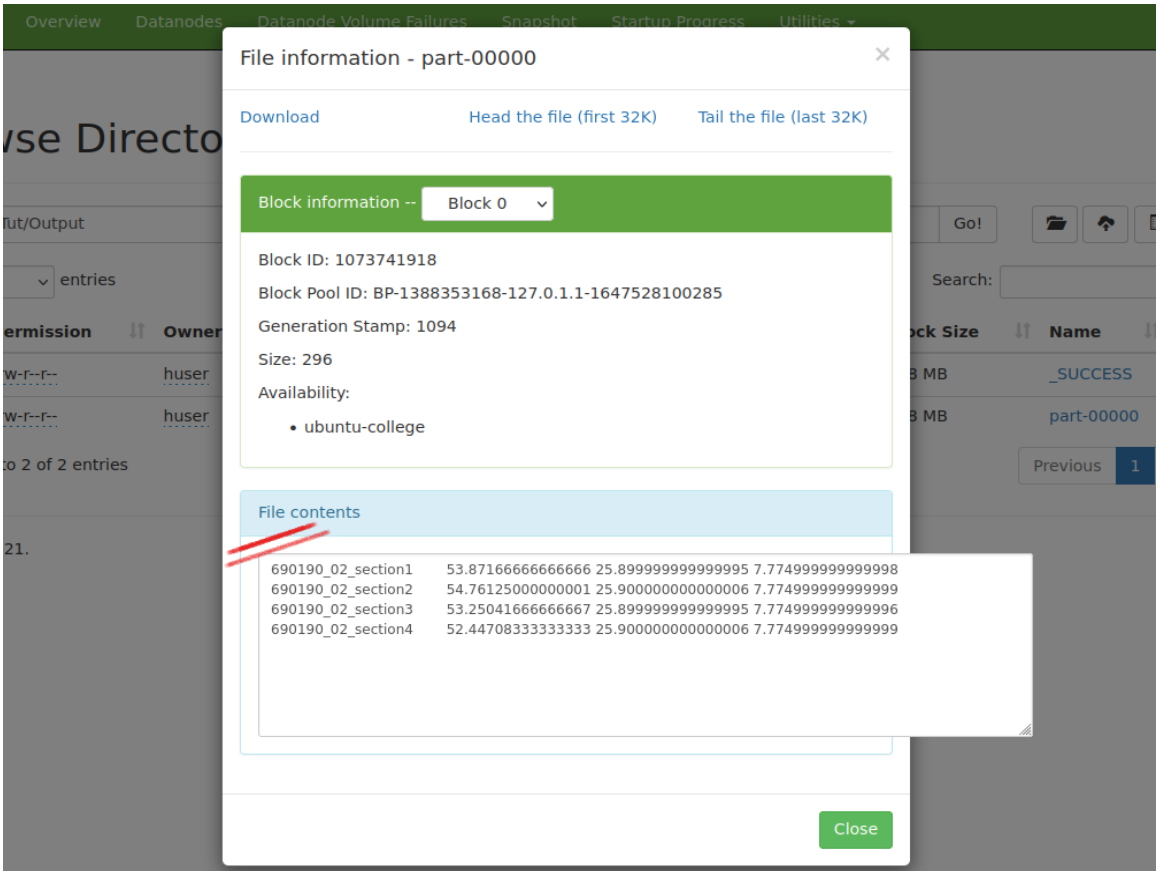
Browse Directory

/WeatherTut/Output Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	huser	supergroup	0 B	Apr 14 23:48	1	128 MB	_SUCCESS
-rw-r--r--	huser	supergroup	296 B	Apr 14 23:48	1	128 MB	part-00000

Showing 1 to 2 of 2 entries Previous 1 Next



14. Stop Hadoop services:

stop-all.sh

```
huser@ubuntu-college:~/Desktop/WeatherAssi$ stop-all.sh
WARNING: Stopping all Apache Hadoop daemons as huser in 10 seconds.
WARNING: Use CTRL-C to abort.
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [ubuntu-college]
Stopping nodemanagers
localhost: WARNING: nodemanager did not stop gracefully after 5 seconds: Trying to kill with kill -9
Stopping resourcemanager
WARNING: resourcemanager did not stop gracefully after 5 seconds: Trying to kill with kill -9
huser@ubuntu-college:~/Desktop/WeatherAssi$
```