Artificial Intelligence & Machine Learning LAB Manual

Sl. No	PROGRAMS
1.	Demonstration of GIT commands and GIT HUB.
2.	Write a program to demonstrate python NumPy and pandas' functions.
3.	Write a program to demonstrate Data Visualization in python using matplotlib for MTCARS dataset.
4.	Write a program to perform Exploratory Data Analysis (EDA) Uni-variate, Bi-variate, and Multi-variate Analysis on Titanic Dataset.
5.	Write a program to identify the attributes containing missing values, number of missing values. perform data cleaning by removing missing values using various techniques.
6.	Write a program to remove outliers in a dataset.
7.	Build simple linear regression machine learning model to analysis relationship between CIE and SEE.
8.	Build multi-linear Regression Model for House Price Prediction.
9.	Program to demonstrate Breast Cancer Detection using Decision Tree Classifier for Wisconsin (diagnostic) Dataset
10.	Build a Predictive model to analysis Heart Disease Prediction using Logistic Regression.
11.	Build a machine learning model to detect Lung Cancer using Support Vector Machine.
12.	Build a supervised machine learning program for Credit Card Fraud Detection using Random Forest Classifier.
13.	Program to demonstrate K-means unsupervised clustering algorithm (mall customer dataset is used to group income v/s spending)
14.	Program to demonstrate Dimensionality Reduction using Principal Component Analysis (PCA) for iris dataset.
15.	Build a Convolutional Neural Networks (CNN) model for MNIST dataset with following conditions.
16.	program to Build NLP pipeline for text processing using NLTK
17.	Write a program to perform Sentimental Analysis using NaiveBayesClassifier
18.	Build a Convolutional neural network to predict Diabetes disease using TensorFlow and keras
19.	Build a convolutional neural network to predict Lung cancer using TensorFlow and keras

WEEK-1

1.Demonstartion of GIT and GIT HUB

Git: configurations

```
$ git config --global user.name "FirstName LastName"
$ git config --global user.email "your-email@email-provider.com"
$ git config --global color.ui true
$ git config --list
```

Git: starting a repository

```
$ git init
$ git status
```

Git: staging files

```
$ git add <file-name>
$ git add <file-name> <another-file-name> <yet-another-file-name>
$ git add .
$ git add --all
$ git add -A
$ git rm --cached <file-name>
$ git reset <file-name>
```

Git: committing to a repository

```
$ git commit -m "Add three files"
$ git reset --soft HEAD^
$ git commit --amend -m <enter your message>
```

Git: pulling and pushing from and to repositories

```
$ git remote add origin <link>
$ git push -u origin master
$ git clone <clone>
$ git pull
```

Git: branching

```
$ git branch
$ git branch <branch-name>
$ git checkout <branch-name>
$ git merge <branch-name>
$ git checkout -b <branch-name>
```

1. **Creating a Git Repository (in git bash)**

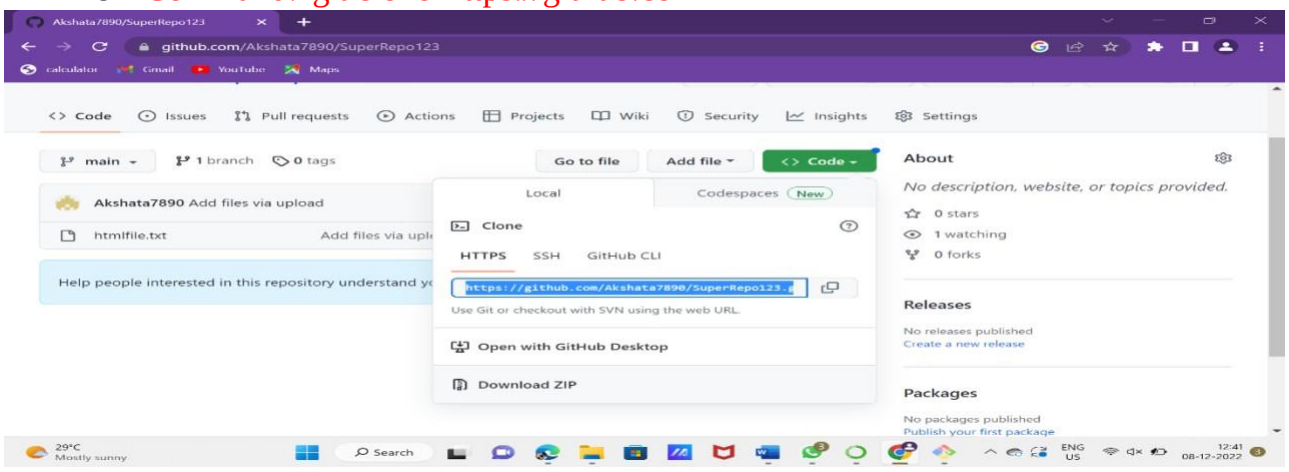
1. Create a directory to add project
2. Change the directory to newly created directory.
3. Initialize the new repository using GIT INIT
4. Create new file in the repository and write some code.
5. Type GIT ADD to add files into repository
6. Type GIT COMMIT to commit the files.

2. Creating a Git Repository in GitHub

1. Type www.github.com
2. Login to GIT HUB account with proper user name and password
3. Click the + symbol to add new repository
4. Type a short, memorable name for your repository.
5. Optionally, add a description of your repository.
6. Choose a repository visibility to public.
7. Select Initialize this repository with a README.
8. Click Create repository.

1. Cloning a Repository

1. Just copy the link of the git repository and run the command
2. After cloning the git repository, the repository will save in your system directly.
3. Command: git clone 'https://github.com'



3. Making and recording changes

Open file and make some changes using following command.

Vim filename

```
Shilpa@DESKTOP-L0FCQRE MINGW64 ~/AIML (master)
$ mkdir AIML
```

```
Shilpa@DESKTOP-L0FCQRE MINGW64 ~ (master)
$ cd AIML
```

```
Shilpa@DESKTOP-L0FCQRE MINGW64 ~/AIML (master)
$ git init
Reinitialized existing Git repository in C:/Users/Shilpa/AIML/.git/

Shilpa@DESKTOP-L0FCQRE MINGW64 ~/AIML (master)
$ git config --global user.name "ravi"

Shilpa@DESKTOP-L0FCQRE MINGW64 ~/AIML (master)
$ git config --global user.email "ravi123@gmail.com"

Shilpa@DESKTOP-L0FCQRE MINGW64 ~/AIML (master)
$ git config --global --list
user.name=ravi
user.email=ravi123@gmail.com

Shilpa@DESKTOP-L0FCQRE MINGW64 ~/AIML (master)
$ vi liner.py
```

Type any Python Code :

```
import pandas as pd
import numpy as np
df=pd.read_csv("C:/Users/Shilpa/Desktop/dataset/marks1.csv")
df.info ()
x = df['CIE'].values.reshape(-1,1)
y = df['SEE'].values.reshape(-1,1)
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split (x, y,random_state =0)
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(x_train, y_train)
y_pred = lm.predict(x_test)
```

Save the file using = esc + shift+ zz :

4.Staging and committing changes

Use git add command for staging and git commit for committing changes.

```
Shilpa@DESKTOP-L0FCQRE MINGW64 ~/AIML (master)
$ git add .
warning: in the working copy of 'liner.py', LF will be replaced by CRLF the next time Git touches it

Shilpa@DESKTOP-L0FCQRE MINGW64 ~/AIML (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   liner.py

Shilpa@DESKTOP-L0FCQRE MINGW64 ~/AIML (master)
$ git commit -m "change 1"
[master (root-commit) 1de4c40] change 1
 1 file changed, 25 insertions(+)
 create mode 100644 liner.py
```

4. **To view the content of the file, use cat command.**
   Command : cat filename

```
Shilpa@DESKTOP-L0FCQRE MINGW64 ~/AIML (master)
$ cat liner.py
```

5. **Undoing changes and committing**

Make some changes in your file and save it. (Use git command : $git restore filename)

```
Shilpa@DESKTOP-L0FCQRE MINGW64 ~/AIML (master)
$ vi liner.py

Shilpa@DESKTOP-L0FCQRE MINGW64 ~/AIML (master)
$ git add .
warning: in the working copy of 'liner.py', LF will be replaced by CRLF the next time Git touches it
```

For undoing changes use the following command.

```
Shilpa@DESKTOP-L0FCQRE MINGW64 ~/AIML (master)
$ git commit -m "change 2"
[master 16b3807] change 2
 1 file changed, 1 insertion(+), 6 deletions(-)
```

6. To view the history to changes made to the file

```
$ git log
commit 16b3807fc20e4b23ccb7c9a792991c562ba702de (HEAD -> master)
Author: ravi <ravi123@gmail.com>
Date:   Mon Oct 16 14:35:27 2023 +0530

    change 2

commit 1de4c4058d998d115ff38b8bca2c323d3b0fa7ac
Author: ravi <ravi123@gmail.com>
Date:   Mon Oct 16 14:32:57 2023 +0530

    change 1
```

undoing changes :

```
Shilpa@DESKTOP-LOFCQRE MINGW64 ~/AIML (master)
$ git diff 1de4c4058d998d115ff38b8bca2c323d3b0fa7ac 16b3807fc20e4b23ccb7c9a792991c562ba702de

diff --git a/liner.py b/liner.py
index 56e2d04..4093420 100644
--- a/liner.py
+++ b/liner.py
@@ -16,10 +16,5 @@ h=y_pred.reshape(21,)
 mydict={"Actual": g,"Pred":h}
 com=pd.DataFrame(mydict)
 com.sample(10)
-def evaluationmatrices(Actual,Pred):
- MAE=mean_absolute_error(Actual,Pred)
- MSE=mean_squared_error(Actual,Pred)
- RMSE=np.sqrt(mean_squared_error(Actual,Pred))
- SCORE=r2_score (Actual,Pred)
- return print ("r2 score:",SCORE,"\n","MAE", MAE,"\n","mse",MSE,"\n","RMSE",RMSE)
+
```

Git Branching and merging

1. Creating and switching to new branches

To create a new branch, use the following code:

Command : **git branch branchname**

To switch to other branch use the following command.
Command : **git checkout branchname**

```
Shilpa@DESKTOP-LOFCQRE MINGW64 ~/AIML (master)
$ git branch
* master

Shilpa@DESKTOP-LOFCQRE MINGW64 ~/AIML (master)
$ git branch a1

Shilpa@DESKTOP-LOFCQRE MINGW64 ~/AIML (master)
$ git branch
  a1
* master

Shilpa@DESKTOP-LOFCQRE MINGW64 ~/AIML (master)
$ git checkout a1
Switched to branch 'a1'
```

2. Merging local branches together

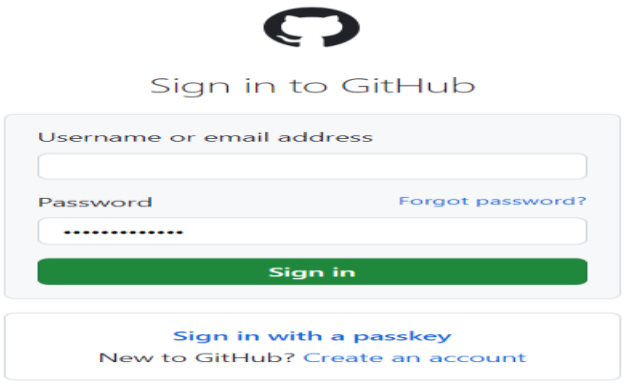
To merge the local branches , use the following

code :Command : git merge branchname

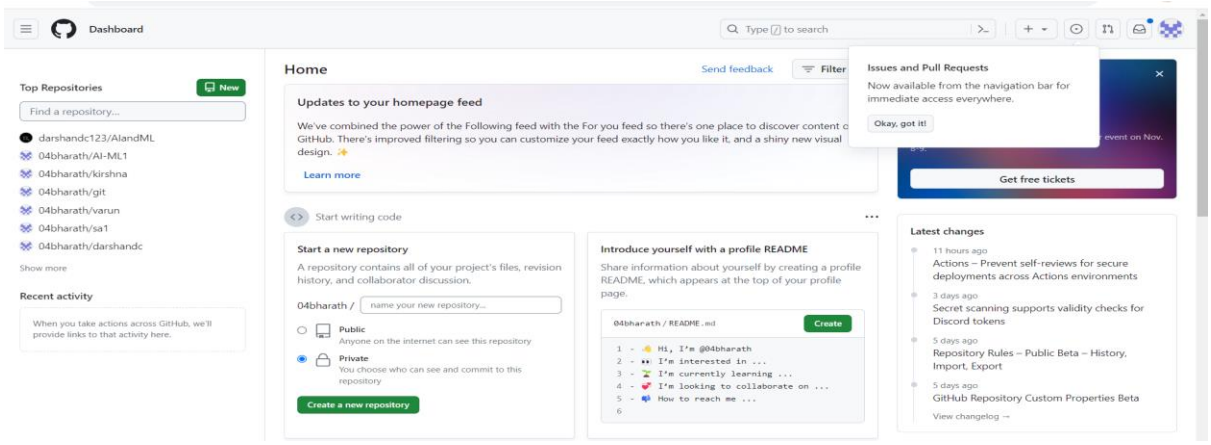
```
Shilpa@DESKTOP-LOFCQRE MINGW64 ~/AIML (a1)
$ git merge a1
Already up to date.

Shilpa@DESKTOP-LOFCQRE MINGW64 ~/AIML (a1)
$ git checkout master
Switched to branch 'master'
```

3. Pushing and pulling from and to repositories

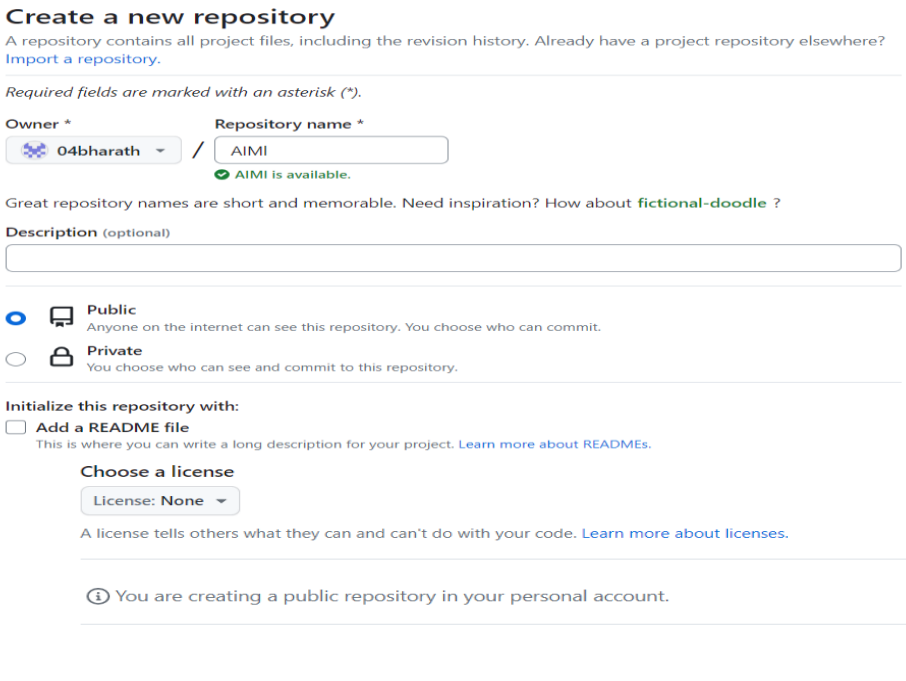
1. Login to github



2. After login click on new



3. Create new repository by giving suitable repository name and click on public and click create repository.



4. In your repository copy this 2 lines and paste into Gitbash

...or create a new repository on the command line

```
echo "# AIMl" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/04bharath/AIMl.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/04bharath/AIMl.git
git branch -M main
git push -u origin main
```

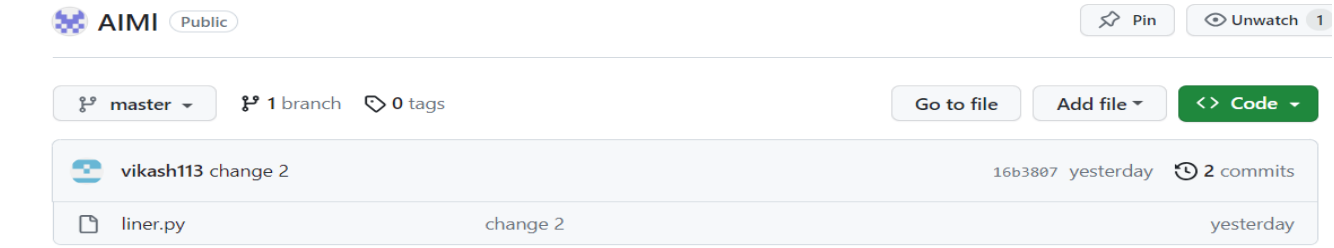
5) To connect to the remote directory from GIT BASH use the following code:

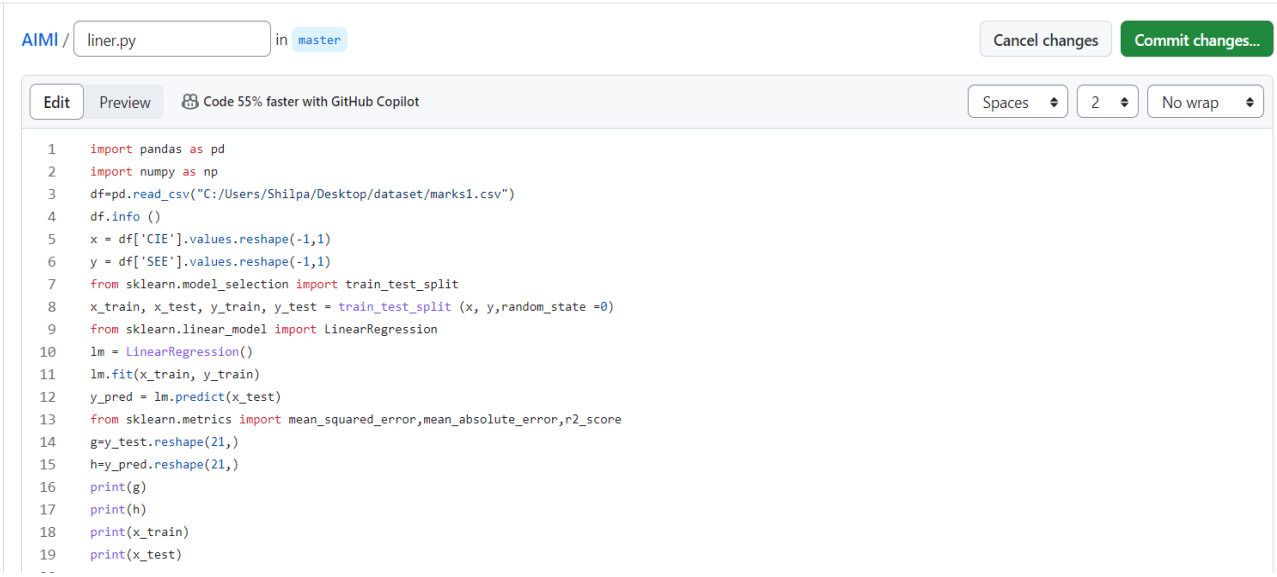
1. git remote add origin "enter the url of the github repo"

2. git push -u origin master

```
Shilpa@DESKTOP-LOFCQRE MINGW64 ~/AIML (master)
$ git remote add origin https://github.com/04bharath/AIML.git

Shilpa@DESKTOP-LOFCQRE MINGW64 ~/AIML (master)
$ git push -u origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 877 bytes | 877.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/04bharath/AIML.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

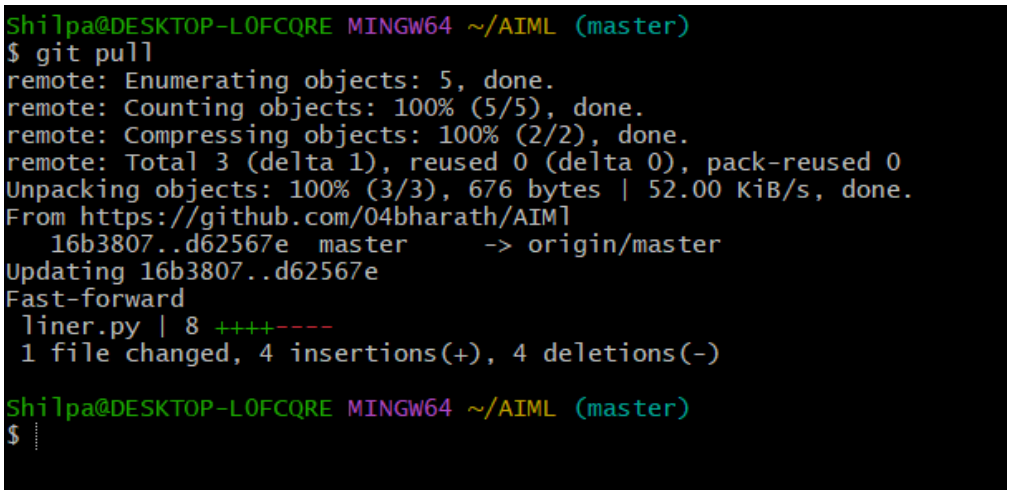
We can check the same file in the GIT-HUB





6) To pull the files from the GIT-HUB to git bash use the command

$ git pull



Week-3
ExploreNumpy operation:

1. Write a program to demonstrate python NumPy Arrays

```
import numpy as np
arr = np.array( [[ 1, 2, 3],
                 [ 4, 2, 5]] )
print("Array is of type: ", type(arr))
print("No. of dimensions: ", arr.ndim)
print("Shape of array: ", arr.shape)
print("Size of array: ", arr.size)
print("Array stores elements of type: ", arr.dtype)
```

2.Program to demonstrate array aggregate function:

```
import numpy as np
 a=np.array([1,2,3,4,5])
 print("a :",a)

sum=np.sum(a)
print("sum :",sum)

product=np.prod(a)
print("product :",product)

mean=np.mean(a)
print("mean :",mean)

 standard_deviation=np.std(a)
 print("standard_deviation :",standard_deviation)

 variance=np.var(a)
 print("variance :",variance)

minimum=np.min(a)
print("minimum value :",minimum)

maximum=np.max(a)
print("maximum value :",maximum)
```

```
minimum_index=np.argmin(a)
print("minimum index :",minimum_index)

maximum_index=np.argmax(a)
print("maximum-index :",maximum_index)

median=np.median(a)
print("median :",median)
```

3.Vectorized Operations Program

```
 # importing the modules
 import numpy as np
 import timeit

# vectorized sum
print(np.sum(np.arange(15)))
print("Time taken by vectorized sum : ", end = "")
%timeit np.sum(np.arange(15))

 # iterative sum
 total = 0
 for item in range(0, 15):
 total += item a = total
 print("\n" + str(a))
 print("Time taken by iterative sum : ", end = "")
 %timeit a
```

Exploring pandas operation

5. Program to demonstrate Lambda Reduce Filter and Map functionc

```
import pandas as pd
import numpy as np

data = pd.DataFrame([ [9, 4, 8, 9],
                [8, 10, 7, 6],
                [7, 6, 8, 5]],
               columns=['Maths', 'English','Science', 'History'])

print(data.agg(['sum', 'min', 'max']))
```

```
m=lambda x:x+10
print("The value of lambda is :",m(5))

print("the square of the column is:")
print(list(map(lambda x:x*x ,data['Maths'])))

a=list(filter(lambda x:x%2,data['Maths']))
print("the result of the filter function is :", a)


from functools import reduce
b=reduce(lambda x,y:x+y, data['Science'])
print(" The output of the reduce is:",b)
```

Output:

	Maths	English	Science	History
sum	24	20	23	20
min	7	4	7	5
max	9	10	8	9

The value of lambda is : 15
The square of column is : [81, 64, 49]
The result of the filter function is : [9, 7]
The output of the reduce is : 23

Aggregation and Grouping Program :

6.program to demonstrate aggregate functions

```
 # import module
 import pandas as pd

# Creating our
dataset df = pd.DataFrame([[9, 4, 8, 9], [8, 10, 7, 6], [7, 6, 8, 5]], columns=['Maths',
'English', 'Science', 'History'])

 # display dataset
 print(df)
#aggregate function
df.agg(['sum', 'min', 'max','mean','median','std','count','size',])
```

7.program to demonstrate PIVOT and MELT Function

2.Data visualization in python using matplotlib for MTCARS dataset: Create the following plots to visualize/summarize the data and customize appropriately.

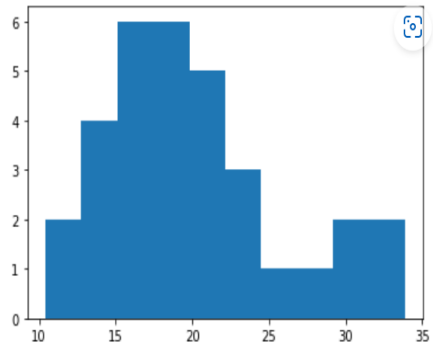
1. **Histogram to check the frequency distribution of the variable 'mpg' (Miles per gallon) and note down the interval having the highest frequency.**
2. **Scatter plot to determine the relation between weight of the car and mpg.**
3. **Bar plot to check the frequency distribution of transmission type of cars.**
4. **Box and Whisker plot of mpg and interpret the five number summary.**

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_csv('mtcars.csv')
print(df.head(10))
```

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

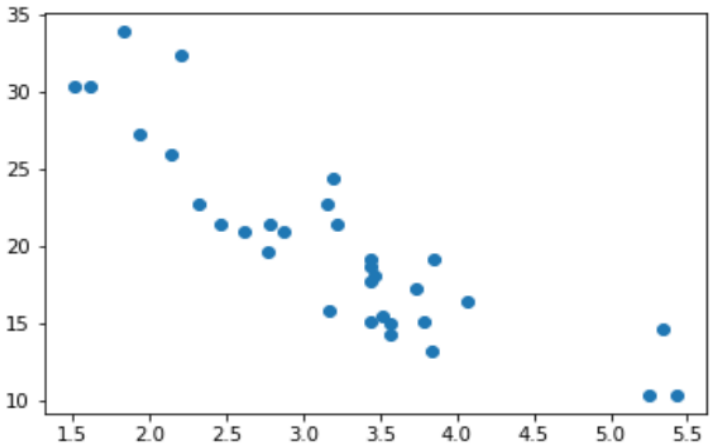
```
print(plt.hist(x=df['mpg']))
```

```
(array([2., 4., 6., 6., 5., 3., 1., 1., 2., 2.]),
 array([10.4 , 12.75, 15.1 , 17.45, 19.8 , 22.15, 24.5 , 26.85, 29.2 ,
        31.55, 33.9 ]),
 <BarContainer object of 10 artists>)
```



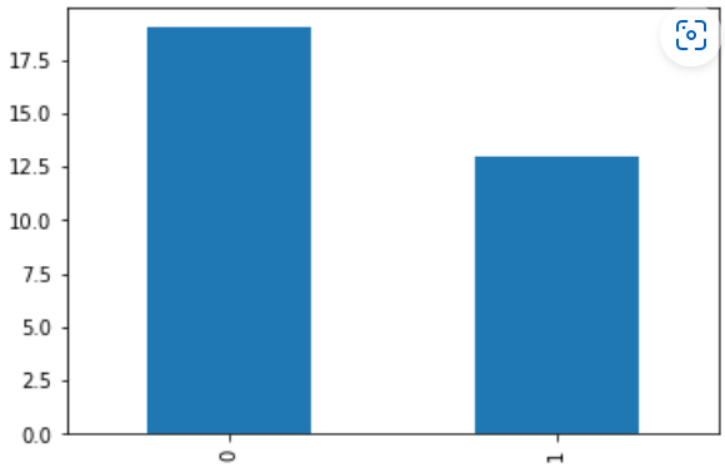
print(plt.scatter(x='wt',y='mpg',data=df))

```
<matplotlib.collections.PathCollection at 0x1fb004f40d0>
```



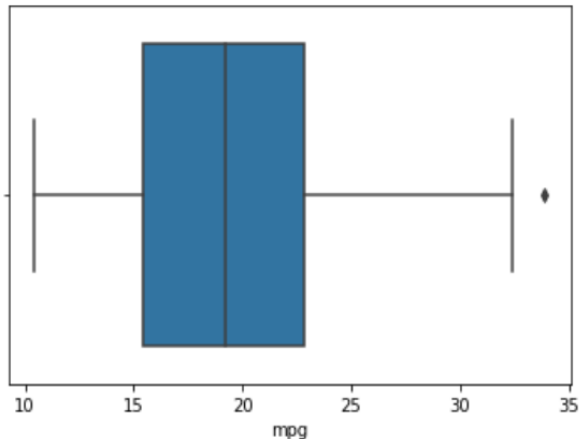
print(df['am'].value_counts().plot(kind='bar'))

```
<AxesSubplot:>
```



print(sns.boxplot(df['mpg']))

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



print(df['mpg'].min())
10.4

print(df['mpg'].max())
33.9

print(df['mpg'].quantile([.1, .25, .5, .75]))

```
0.10    14.340
0.25    15.425
0.50    19.200
0.75    22.800
Name: mpg, dtype: float64
```

3.Perform Exploratory Data Analysis (EDA) and Uni-variate, Bi-variate, and Multi-variate Analysis on titanic Dataset.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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

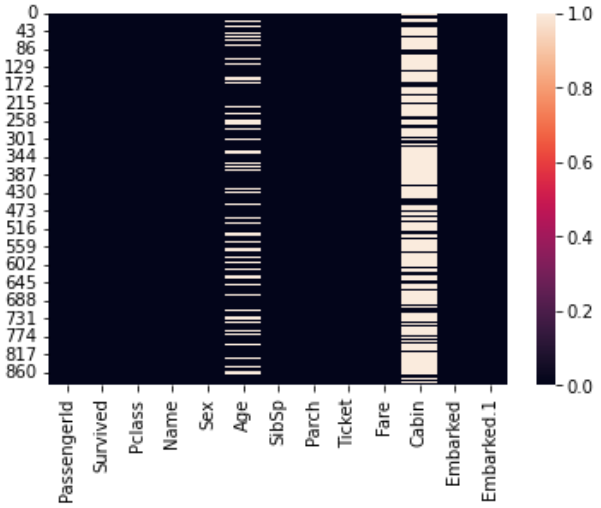
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 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
 12  Embarked.1   889 non-null    object
dtypes: float64(2), int64(5), object(6)
memory usage: 90.6+ KB
```

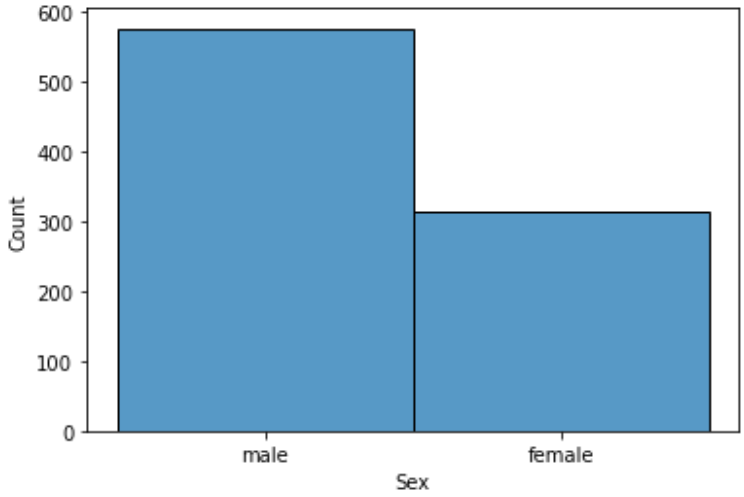
```
data.describe()
```

	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

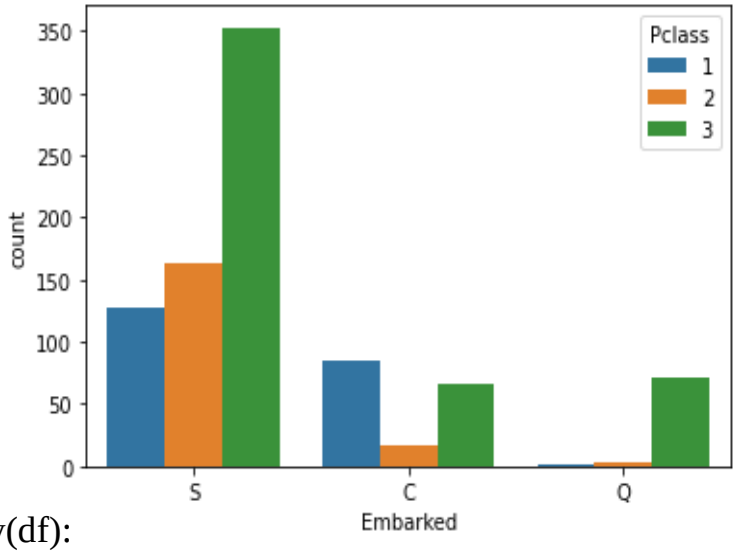
```
sns.heatmap(data.isna())
```




```
g=sns.histplot(x='Sex', data=data)
```




```
g=sns.countplot(x='Embarked', hue='Pclass', data=data)
```

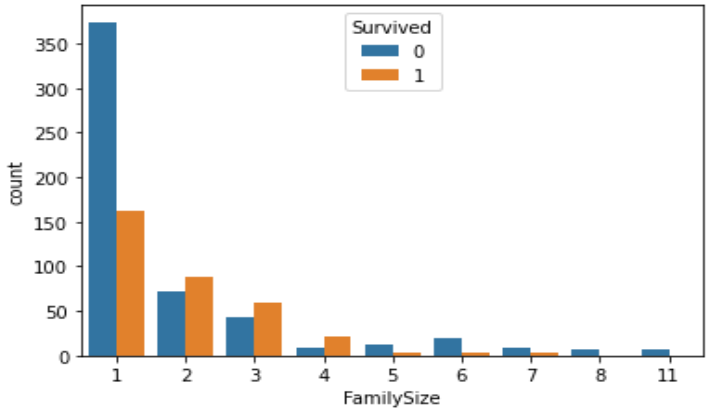



```
def add_family(df):
    df['FamilySize']=df['SibSp'] + df['Parch'] +1
    return df
```

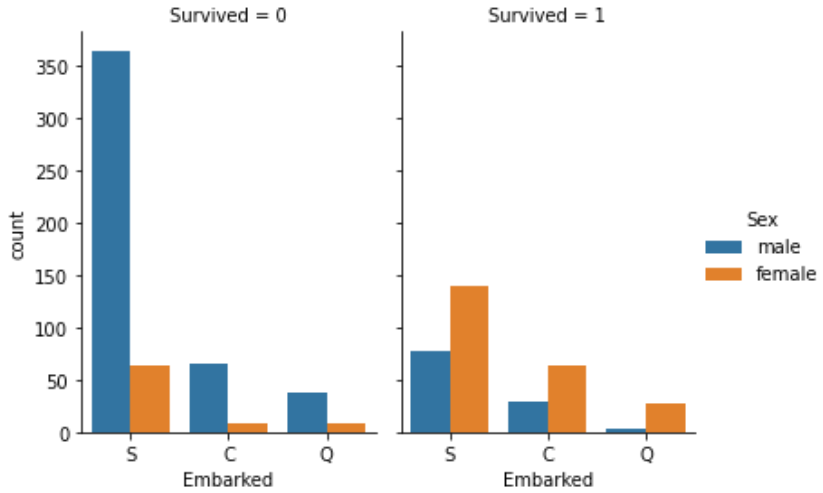
```
data=add_family(data)
data.head(10)
```

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

```
g=sns.countplot(x='FamilySize', hue='Survived', data=data)
```




```
g=sns.catplot(x="Embarked", hue="Sex", col="Survived",data=data,
kind="count", height=4, aspect=.7)
```



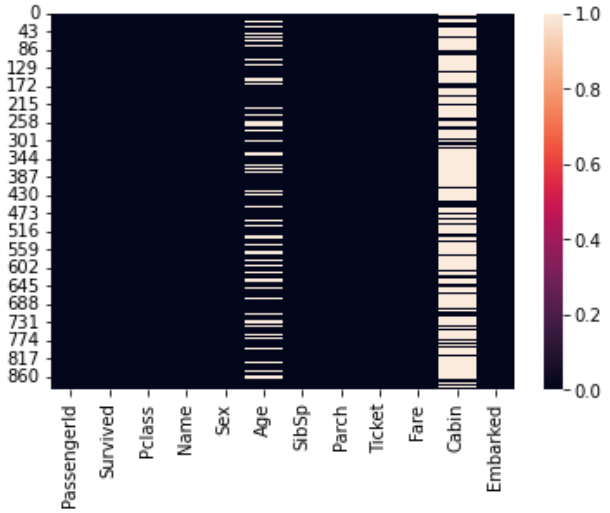
4.Write a program to identify the attributes containing missing values, number of missing values. perform data cleaning by removing missing values using various techniques.

```
import pandas as pd
import seaborn as sns

df=pd.read_csv("C:/Users/Shilpa/Desktop/LAB_programs_dataset/Titanic_dataset.csv")
print(df.head())

#Checking missing values
print(df.isna().sum())
sns.heatmap(df.isna())

#Filling missing values through mean
df['Age']. fillna(df['Age']. mean(),inplace=True)


#Filling missing values through mode
df['Embarked'].fillna(df['Embarked'].mode()[0],inplace=True)

#Dropping column
df.drop(['Cabin'],axis=1,inplace=True)

#Dropping specific rows
df.drop(df[(df['Name']=="Braund, Mr. Owen Harris")].index,inplace=True)
df.drop(df[(df['PassengerId']==5)].index,inplace=True)
```

```
print(df.isna().sum())
sns.heatmap(df.isna())
```

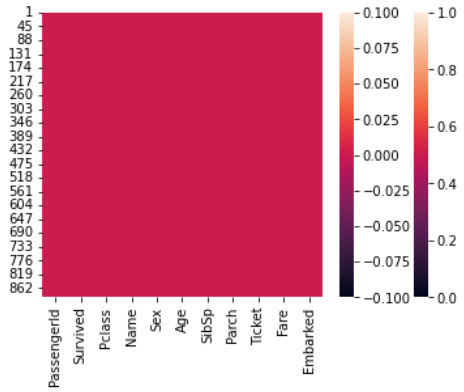
Output:

```
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
1            2         1       1  ...  71.2833   C85         C
2            3         1       3  ...   7.9250   NaN         S
3            4         1       1  ...  53.1000  C123         S
4            5         0       3  ...   8.0500   NaN         S

[5 rows x 12 columns]
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
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
```

```
Fare          0
Embarked      0
dtype: int64
```



5. Write a program to demonstrate to remove outliers in a dataset

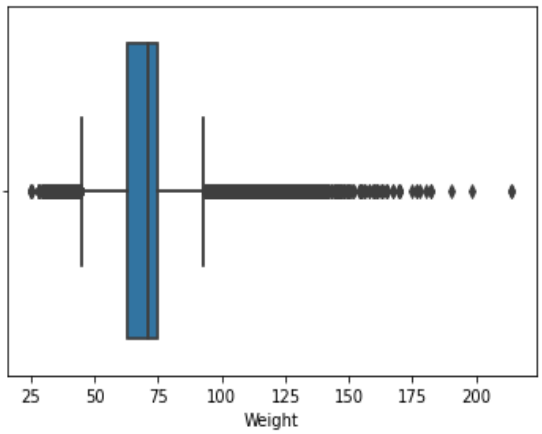
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data=pd.read_csv("C:/Users/Shilpa/Desktop/LAB_programs_dataset/Athlete_even
ts_outliers.csv")

print(data.head(10))

#Removing missing values in Height and weight columns
c=data['Age'].mean()
data['Age'].fillna(c,inplace=True)

a=data['Height'].mean()
data['Height'].fillna(a,inplace=True)
print(a)

b=data['Weight'].mean()
data['Weight'].fillna(b,inplace=True)
print(b)

data.info()

data['Weight'].skew()
sns.boxplot(data['Weight'])

q1=data['Weight'].quantile(0.25)
q3=data['Weight'].quantile(0.75)
```

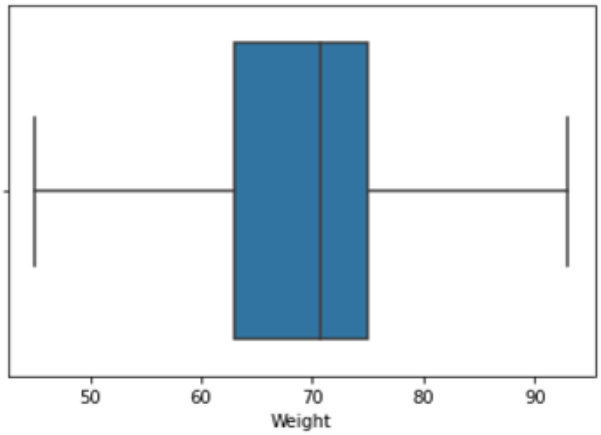
```
IQR=q3-q1

lower=q1-(1.5*IQR)
upper=q3+(1.5*IQR)
data['Weight']=np.where(data['Weight']>upper,upper,np.where(data['Weight']<low
er,lower,data['Weight']))

sns.boxplot(data['Weight'])
data['Weight'].skew()
```

Output:

175.33896987366376

70.70239290053351





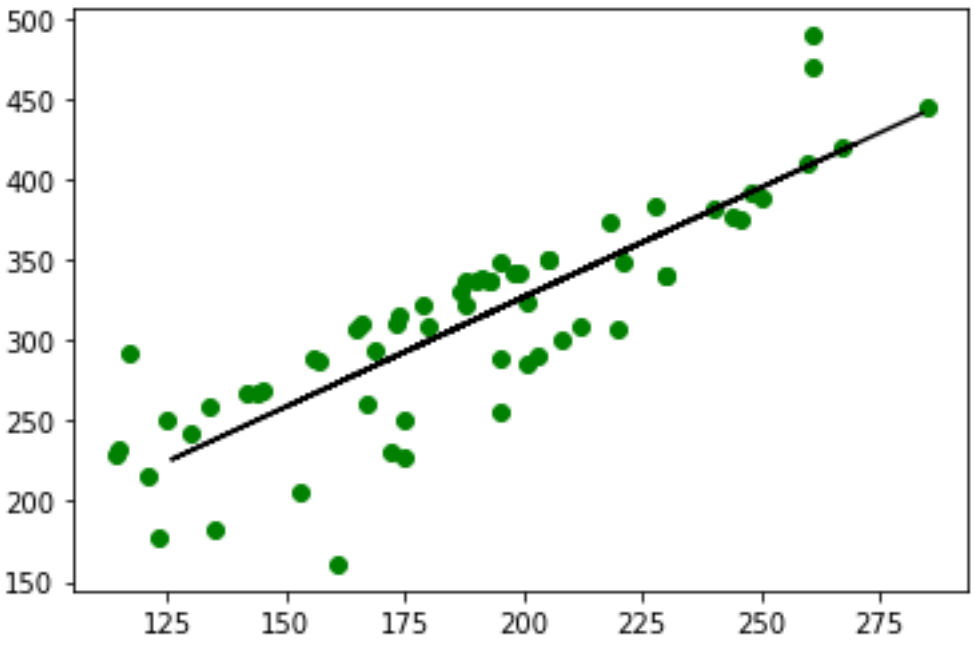
6.Build Simple Linear Regression Machine Learning Model to analysis relationship between CIE and SEE

```
import pandas as pd
import numpy as np

df=pd.read_csv("CIE_SEE.csv")
print(df.info ())

x=df['CIE'].values.reshape(-1,1)
y=df['SEE'].values.reshape(-1,1)

from sklearn.model_selection import train_test_split
x_train, x_test,y_train,y_test=train_test_split(x,y,random_state=0)

from sklearn.linear_model import LinearRegression
lm=LinearRegression()
lm.fit(x_train,y_train)
y_pred=lm.predict(x_test)

from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
g=y_test.reshape(21,)
h=y_pred.reshape(21,)

print("MAE--->",mean_absolute_error(g,h))
print("MSE--->",mean_squared_error(g,h))
print("score--->",r2_score(g,h))
print (" RMSE--->",np.sqrt(mean_squared_error(g,h)))

import matplotlib.pyplot as plt
plt.scatter(x_train, y_train,color='g')
plt.plot(x_test, y_pred,color='k')
```

```
plt.show()
```

output:

```
RangeIndex: 83 entries, 0 to 82
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   CIE     83 non-null     int64
 1   SEE     83 non-null     int64
dtypes: int64(2)
memory usage: 1.4 KB
None
MAE---> 29.367841250713415
MSE---> 1593.0385277231076
score---> 0.4491593937437446
 RMSE---> 39.912886737532624
```



7.Build a Multi Linear Regression Model for House Price Prediction

```
import pandas as pd
import numpy as np

df=pd.read_csv("C:/Users/Shilpa/Desktop/LAB_programs_dataset/Housing_multilinear_
reg.csv")
df.head()

df=pd.get_dummies(df)

df.drop(['mainroad_no','guestroom_no','basement_yes','hotwaterheating_yes','airconditio
ning_yes'],axis=1,inplace=True)

x=df.iloc[:,1:]
y=df.iloc[:,0]  #0 because target variable price is in zero column


from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

from sklearn.linear_model import LinearRegression
lm=LinearRegression()
lm.fit(x_train,y_train)
y_pred=lm.predict(x_test)

from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error

print("MSE         ---- >" , mean_squared_error(y_test,y_pred))
print("RMSE       ----- > " , np.sqrt(mean_squared_error(y_test,y_pred)))
print ("MAE        ----- >", mean_absolute_error(y_test,y_pred))
print("r2 score “ ----- >",r2_score(y_test,y_pred))
```

Output:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished

MSE ---- > 1080485739437.5288
RMSE ----- > 1039464.1597657559
MAE ----- > 797371.25393815
r2 score "----- > 0.6598261620391519

8.Build predictive machine learning model for Breast Cancer Detection using Decision Tree Classifier for Wisconsin (diagnostic) Dataset

```
import pandas as pd
data=pd.read_csv("C:/Users/Shilpa/Desktop/LAB_programs_dataset/breast_cancer_analysis_DecisionTrees.csv")
print (data.info ())

data=data.drop(['id'],axis=1)
x=data.drop(['diagnosis'],axis=1)
y=data['diagnosis']

from sklearn.model_selection  import train_test_split
x_train, x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier()
model.fit(x_train,y_train)

y_pred=model.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report
print ("The accuracy of the model built is ", accuracy_score(y_pred,y_test)*100)

#Finding Best Hyperparameters for Decision Trees Using GridSearch
from sklearn.model_selection import GridSearchCV
pram_dict={'criterion':['gini','entropy'],
        'max_depth':range(1,10),
        'min_samples_split':range(1,10),
```

```
        'min_samples_leaf':range(1,5)}
grid=GridSearchCV(model, param_grid=pram_dict,cv=10,verbose=1,n_jobs=-1)
grid.fit(x_train,y_train)
print(grid.best_score_)
```

output:

The accuracy of the model built is 91.22807017543859

Fitting 10 folds for each of 648 candidates, totalling 6480 fits
0.9405797101449276

9.Build a Predictive Model to Analysis Heart Disease Prediction using Logistic Regression.

```
import pandas as pd
import numpy as np
data=pd.read_csv("C:/Users/Shilpa/Desktop/LAB_programs_dataset/Heart_disease_logr
egression.csv")
data.head(10)

#converting String to Integer using label encoder
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
data=data.apply(lambda x:le.fit_transform(x))

x = data.drop(['HeartDisease'],axis=1)
y = data['HeartDisease']

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=0)

from sklearn.linear_model import LogisticRegression
log_regression = LogisticRegression()
log_regression.fit(x_train,y_train)
y_pred = log_regression.predict(x_test)

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

```
print("confusion_matrix: ",confusion_matrix(y_test, y_pred))
print("classification_report:")
print(metrics.classification_report(y_test, y_pred))
```

Output:

```
confusion_matrix:  [[ 91  22]
                    [ 24 139]]
```

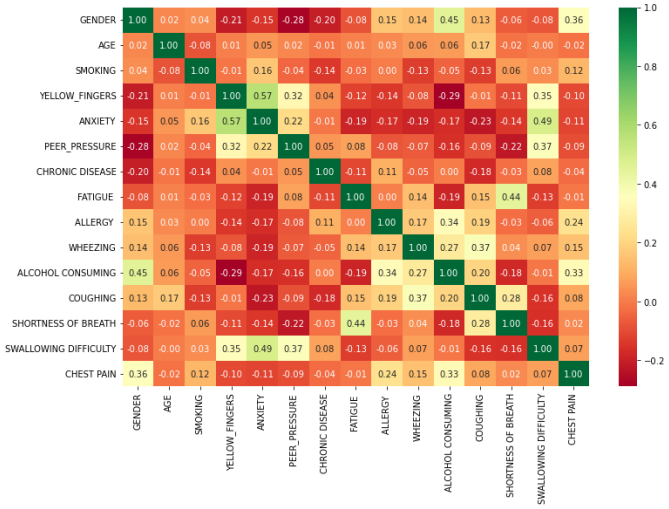
classification_report:

	precision	recall	f1-score	support
0	0.79	0.81	0.80	113
1	0.86	0.85	0.86	163
accuracy			0.83	276
macro avg	0.83	0.83	0.83	276
weighted avg	0.83	0.83	0.83	276

10.Build predictive Machine Learning model to Detect Lung Cancer using Support Vector Machine

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data=pd.read_csv("LUNG_CANCER.csv")
data.head()

x= data. drop(labels=['LUNG_CANCER'],axis=1))
y=data['LUNG_CANCER'].values.reshape(-1,1)

sns.heatmap(corrmat,annot=True,fmt='.2f',cmap='RdYlGn',ax=ax)
plt.show()

from sklearn.model_selection import train_test_split
x_train, x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
from sklearn.svm import SVC
sv=SVC ( )
sv.fit(x_train,y_train)
y_pred=sv.predict(x_test)

from sklearn.metrics import classification_report,accuracy_score
print ("Classification_report\n", classification_report(y_test,y_pred))
print('Accuracy',accuracy_score(y_test,y_pred))
```

output:



Classification_report	precision	recall	f1-score	support
NO	0.80	0.44	0.57	9
YES	0.91	0.98	0.95	53
accuracy			0.90	62
macro avg	0.86	0.71	0.76	62
weighted avg	0.90	0.90	0.89	62

Accuracy 0.9032258064516129

11.Build a supervised machine learning program for Credit Card Fraud Detection using Random Forest Classifier.

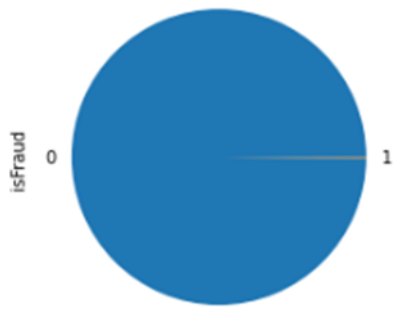
```
import pandas as pd
df=pd. read_csv("creditcard.csv")

print(df.head())
print(df.isna().sum())

del df['nameOrig']
del df['nameDest']
df['isFraud'].value_counts(). plot(kind='pie')

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['type'] = le.fit_transform(df['type'])
df.head()

x=df.iloc[ : , : -1]
y=df.iloc[: , -1]

from sklearn.model_selection import train_test_split
x_train,x_test, y_train, y_test = train_test_split(x_train,y_train,test_size=0.2)

from sklearn.ensemble import RandomForestClassifier
rm=RandomForestClassifier()
```

```
rm.fit(x_train,y_train)
y_pred=rm.predict(x_test)

from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
print(confusion_matrix(y_pred,y_test))
print(accuracy_score(y_pred,y_test))
print(classification_report(y_pred,y_test))
```

output:

```
step      type    amount  ... oldbalanceDest  newbalanceDest  isFraud
0     1   PAYMENT   9839.64  ...           0.0             0.0        0
1     1   PAYMENT   1864.28  ...           0.0             0.0        0
2     1  TRANSFER    181.00  ...           0.0             0.0        1
3     1  CASH_OUT    181.00  ...       21182.0             0.0        1
4     1   PAYMENT  11668.14  ...           0.0             0.0        0

[5 rows x 10 columns]
step                0
type                0
amount              0
nameOrig            0
oldbalanceOrig      0
newbalanceOrig      0
nameDest            0
oldbalanceDest      0
newbalanceDest      0
isFraud             0
dtype: int64
```

```
<AxesSubplot:ylabel='isFraud'>
```



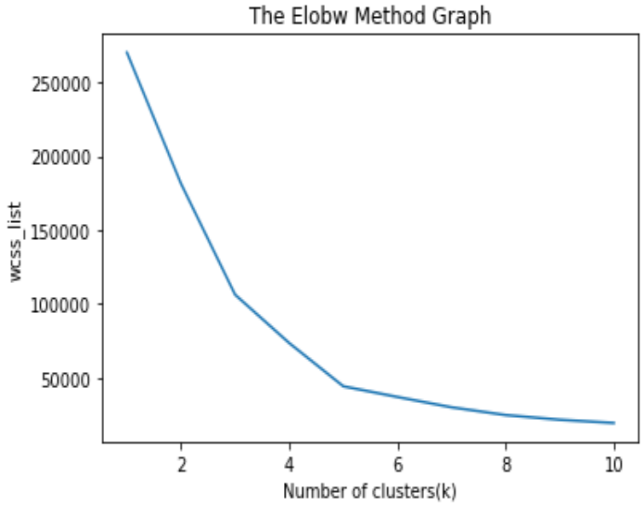

```
Name: isFraud, dtype: float64
 [[209455     15]
 [   228 209276]]
Accuracy 0.9994200117429721
                    precision    recall  f1-score   support
```

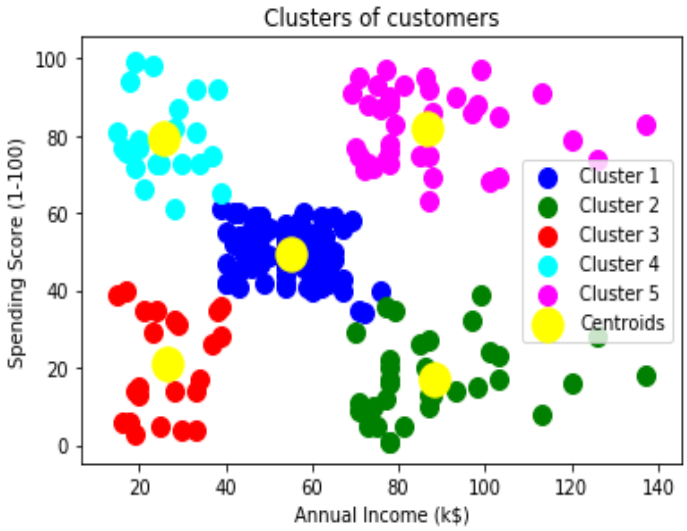
0	1.00	1.00	1.00	209470
1	1.00	1.00	1.00	209504
accuracy			1.00	418974
macro avg	1.00	1.00	1.00	418974
weighted avg	1.00	1.00	1.00	418974

12.Program to demonstrate K-means unsupervised clustering algorithm (mall customer dataset is used to group income v/s spending)

```
# Importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

 dataset = pd.read_csv('Mall_Customers_data.csv')
 x = dataset.iloc[:, [3, 4]].values

#finding optimal number of clusters using the elbow method
from sklearn.cluster import KMeans
wcss_list= []

#Using for loop for iterations from 1 to 10.
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)

mtp.plot(range(1, 11), wcss_list)
mtp.title('The Elobw Method Graph')
mtp.xlabel('Number of clusters(k)')
mtp.ylabel('wcss_list')
mtp.show()
```

```
#training the K-means model on a dataset
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x) #visulaizing the clusters

mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue',   label = 'Cluster 1')
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2')
mtp.scatter(x[y_predic t== 2, 0], x[y_predict == 2, 1], s = 100,c = 'red',     label = 'Cluster 3'
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan',   label = 'Cluster 4')
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')

mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c
 = 'yellow', label = 'Centroid')

mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```





The output image is clearly showing the five different clusters with different colors. The clusters are formed between two parameters of the dataset; Annual income of customer and Spending. We can change the colors and labels as per the requirement or choice. We can also observe some points from the above patterns,

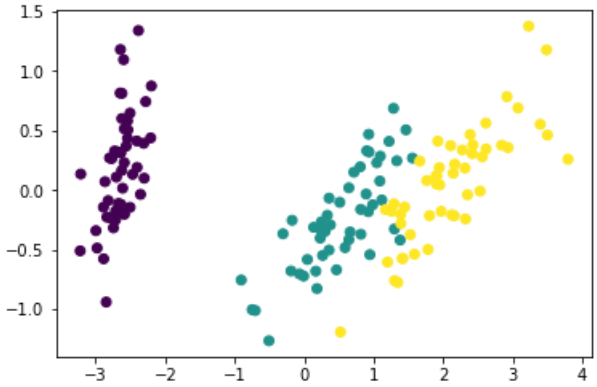
- **Cluster1**: shows the customers with average salary and average spending so we can categorize these customers as
- **Cluster2** shows the customer has a high income but low spending, so we can categorize them as **careful**.
- **Cluster3** shows the low income and low spending so they can be categorized as **sensible.**
- **Cluster4** shows the customers with low income with very high spending so they can be categorized as **careless**.
- **Cluster5** shows the customers with high income and high spending so they can be categorized as **target**, and these customers can be the most profitable customers for the mall owner.

13.Program to Demonstrate Dimensionality Reduction using principal component analysis (PCA) for iris dataset.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.decomposition import PCA

iris=datasets.load_iris()
x=iris.data
y=iris.target

print(x.shape)
print(y.shape)

pca=PCA(n_components=2)
pca.fit(x)
print(pca.components_)

x=pca.transform(x)
```

```
print("shape of X: after transformation",x.shape)
plt.scatter(x[:,0],x[:,1],c=y)

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

x_train, x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
res=DecisionTreeClassifier()
res.fit(x_train,y_train)

y_predict=res.predict(x_test)
print(accuracy_score(y_test,y_predict))
```

output:



Shape before PCA : (150, 4)

(150,)

[[0.36138659 -0.08452251 0.85667061 0.3582892]
[0.65658877 0.73016143 -0.17337266 -0.07548102]]

Shape after pca (150, 2)

Accuracy----→ 0.9666666666666667

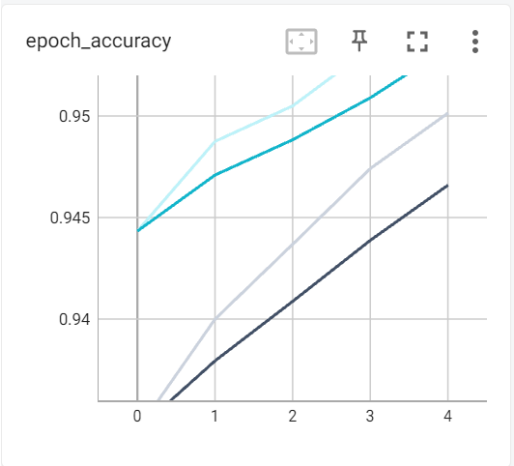
14.**Build a Convolutional Neural Networks (CNN) model for MNIST dataset with following conditions.**

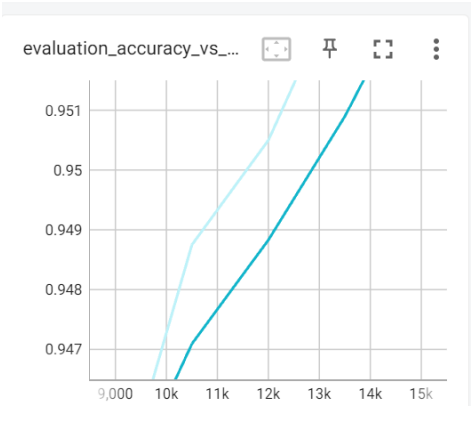
- **One Flatten () layer. o One Dense layer with 512 neurons using a ReLU as the activation function.**
- **A Dropout layer with the probability of retaining the unit of 20%.**
- **A final Dense layer, that computes the probability scores via the softmax function, for each of the 10 output labels.**
- **Show the losses and the final architecture on TensorBoard.**

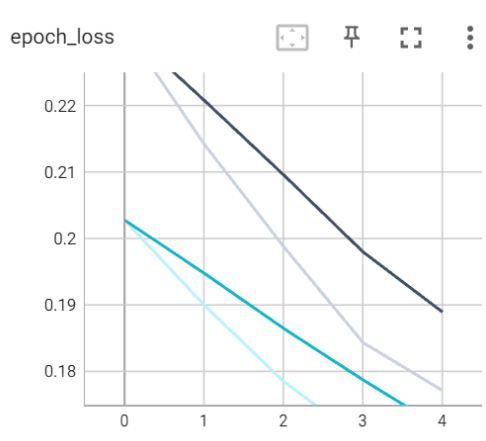
```
import tensorflow as tf
m=tf.keras.datasets.mnist

(x_train,y_train),(x_test,y_test)=m.load_data()
x_train,x_test=x_train/255,x_test/255

model=tf.keras.models.Sequential([
tf.keras.layers.Flatten(input_shape=(28,28)),
tf.keras.layers.Dense(512,activation='relu'),
tf.keras.layers.Dropout(0.2),
tf.keras.layers.Dense(10,activation='softmax')])

model.compile(optimizer='sgd',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

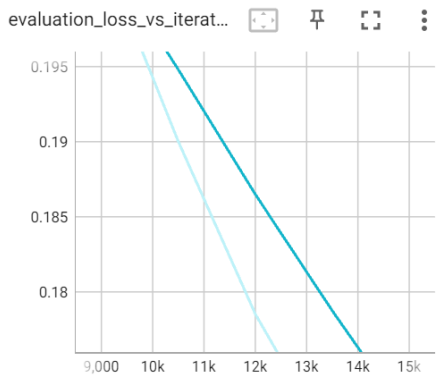
```
log="C:/Users/varsh/OneDrive/Desktop/log"
from tensorflow.keras.callbacks import TensorBoard

callbacks= [TensorBoard(
log_dir=log,
histogram_freq=1,
write_graph=True,
write_images=True,
update_freq='epoch',
profile_batch=2,
embeddings_freq=1)]

model.fit(x_train, y_train,epochs=5,validation_split=0.2,callbacks=callbacks)
model.save('m1.hs')
```

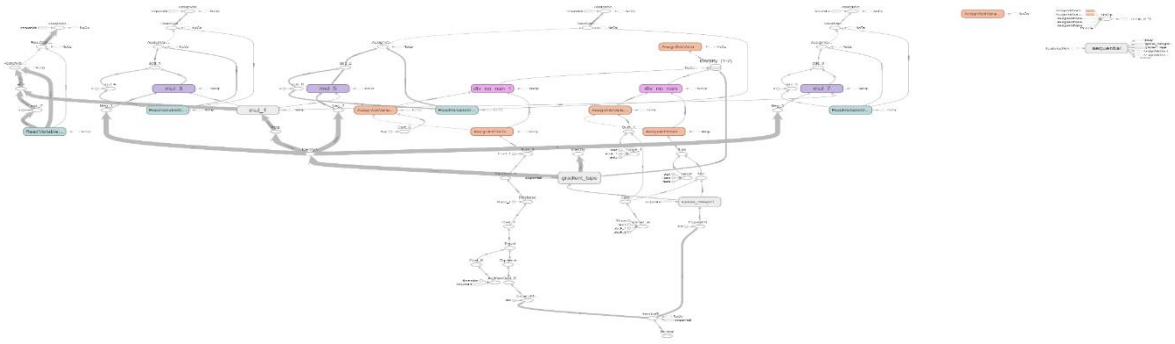
In CMD:
Type:
C:\Users\varsh>python
#Install python
C:\Users\varsh>pip3 install tensorboard
C:\Users\varsh>python -m tensorboard.main --
logdir="C:/Users/varsh/OneDrive/Desktop/log"--port=6006
#Copy the link and paste in google











```
import nltk
from nltk import sent_tokenize
from nltk import word_tokenize
from nltk.corpus import stopwords


text= "The first time you see The Second Renaissance it may look boring. Look at it at least twice and watch part 2. It will change your view of the matrix. Are the human people the ones who started the war? Is AI a bad thing?"
print(text)

#Tokenization
word_tocken = word_tokenize(text)
print(word_tocken)

#Normalization
#Punctuation Removal
elist= [ ]
for i in word_tocken:
    if i.isalpha():
        elist.append(i)
print(elist)

#Stop Words Removal
```

```
stopwords=stopwords.words("english")
print (stopwords)


elist1=[]
for i in elist:
    if i not in stopwords:
        elist1.append(i)
print(elist1)


#Parts of Speech (POS) Tagging
#Named Entity Recognition (NER)

from nltk import pos_tag
from nltk import ne_chunk
tag=nltk.pos_tag(elist1)
print(tag)

tree=nltk.ne_chunk(tag,binary=True)
print(tree)
tree.draw()

#Lemmatization
from nltk import WordNetLemmatizer
lemma= WordNetLemmatizer()
word_list=elist1
g=[]
for i in word_list:
    g.append(lemma.lemmatize(i))
print(g)

#Tf-IdfVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
x=vectorizer.fit_transform(g)
print(x.toarray())
```

output:

The first time you see The Second Renaissance it may look boring. Look at it at least twice and watch part 2. It will change your view of the matrix. Are the human people the ones who started the war? Is AI a bad thing?
['The', 'first', 'time', 'you', 'see', 'The', 'Second', 'Renaissance', 'it', 'may', 'look', 'boring', '.', 'Look', 'at', 'it', 'at', 'least', 'twice', 'and', 'watch', 'part', '2', '.', 'It', 'will', 'change', 'your', 'view', 'of', 'the', 'matrix', '.', 'Are', 'the', 'human', 'people', 'the', 'ones', 'who', 'started', 'the', 'war', '?', 'Is', 'AI', 'a', 'bad', 'thing', '?']

['The', 'first', 'time', 'you', 'see', 'The', 'Second', 'Renaissance', 'it', 'may', 'look', 'boring', 'Look', 'at', 'it', 'at', 'least', 'twice', 'and', 'watch', 'part', 'It', 'will', 'change', 'your', 'view', 'of', 'the', 'matrix', 'Are', 'the', 'human', 'people', 'the', 'ones', 'who', 'started', 'the', 'war', 'Is', 'AI', 'a', 'bad', 'thing']

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

['The', 'first', 'time', 'see', 'The', 'Second', 'Renaissance', 'may', 'look', 'boring', 'Look', 'least', 'twice', 'watch', 'part', 'It', 'change', 'view', 'matrix', 'Are', 'human', 'people', 'ones', 'started', 'war', 'Is', 'AI', 'bad', 'thing']
[('The', 'DT'), ('first', 'JJ'), ('time', 'NN'), ('see', 'VB'), ('The', 'DT'), ('Second', 'NNP'), ('Renaissance', 'NNP'), ('may', 'MD'), ('look', 'VB'), ('boring', 'VBG'), ('Look', 'NNP'), ('least', 'JJS'), ('twice', 'RB'), ('watch', 'JJ'), ('part', 'NN'), ('It', 'PRP'), ('change', 'VBZ'), ('view', 'NN'), ('matrix', 'NN'), ('Are', 'NNP'), ('human', 'JJ'), ('people', 'NNS'), ('ones', 'NNS'), ('started', 'VBD'), ('war', 'NN'), ('Is', 'NNP'), ('AI', 'NNP'), ('bad', 'JJ'), ('thing', 'NN')]
(S
  The/DT
  first/JJ
  time/NN
  see/VB
  The/DT
  (NE Second/NNP Renaissance/NNP)
  may/MD
  look/VB
  boring/VBG
  Look/NNP
  least/JJS
  twice/RB
  watch/JJ
  part/NN
  It/PRP
  change/VBZ
  view/NN
  matrix/NN
  Are/NNP
  human/JJ

```
people/NNS
ones/NNS
started/VBD
war/NN
Is/NNP
AI/NNP
bad/JJ
thing/NN)
```

16.Write a program to perform Sentimental Analysis using NLTK

```
from textblob import TextBlob
from textblob.classifiers import NaiveBayesClassifier
train = [
    ('I love this sandwich.', 'pos'),
    ('This is an amazing place!', 'pos'),
    ('I feel very good about these beers.', 'pos'),
    ('I do not like this restaurant', 'neg'),
    ('I am tired of this stuff.', 'neg'),
    ("I can't deal with this", 'neg'),
   ("My boss is horrible.", "neg")
   ]
cl = NaiveBayesClassifier(train)

print("The polarity of sentence I feel amazing is",cl.classify("I feel amazing!"))
blob = TextBlob("The beer is good. But the hangover is horrible. I can't drive",
classifier=cl)

for s in blob.sentences:
   print(s)
   print(s.classify())
```

output:

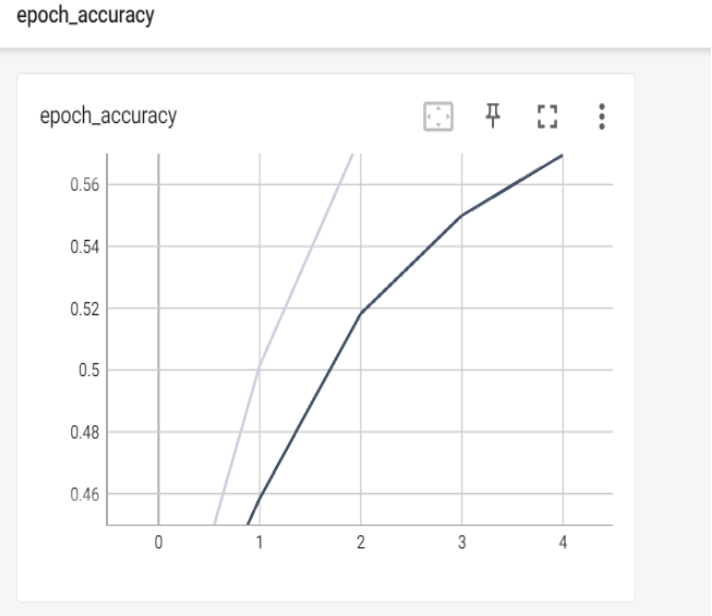
The polarity of sentence I feel amazing is pos
The beer is good.
pos
But the hangover is horrible.
neg
I can't drive
Neg

17. Build neural networks to predict diabetes using TensorFlow and keras

```
import pandas as pd
data = pd.read_csv("diabetes .csv")

x = data.drop("Outcome", axis=1)
y = data["Outcome"]

from keras.models import Sequential
from keras.layers import Dense

model = Sequential(
model.add(Dense(12, input_dim=8, activation="relu"))
model.add(Dense(12, activation="relu"))
model.add(Dense(1, activation="sigmoid"))

model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
log="C:/Users/Shilpa/Desktop/logs"

from tensorflow.keras.callbacks import TensorBoard
callbacks= [TensorBoard(
log_dir=log,
histogram_freq=1,
write_graph=True,
write_images=True,
```

```
update_freq='epoch',
profile_batch=2,
embeddings_freq=1)]

model.fit(x,y, epochs=10, batch_size=10,callbacks=callbacks)

_, accuracy = model.evaluate(x, y)
print("Model accuracy: %.2f"% (accuracy*100))
```

Output:

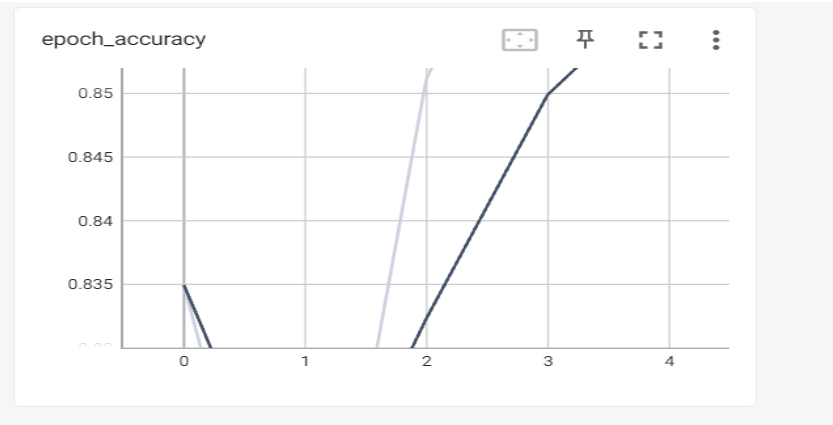


18. Build neural networks to predict lung cancer using TensorFlow and keras

```
import pandas as pd
data = pd.read_csv("survey_lung_cancer_tensorflow.csv")

x = data.drop("LUNG_CANCER", axis=1)
y = data["LUNG_CANCER"]

from keras.models import Sequential
from keras.layers import Dense

model = Sequential(
model.add(Dense(512, input_dim=15, activation="relu"))
model.add(Dense(512, activation="relu"))
model.add(Dense(1, activation="sigmoid"))

model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
```

```
log="C:/Users/Shilpa/Desktop/logs"
from tensorflow.keras.callbacks import TensorBoard

callbacks= [TensorBoard(
log_dir=log,
histogram_freq=1,
write_graph=True,
write_images=True,
update_freq='epoch',
profile_batch=2,
embeddings_freq=1)]

model.fit(x,y, epochs=5, batch_size=10,callbacks=callbacks)

_, accuracy = model.evaluate(x, y)
print("Model accuracy: %.2f"% (accuracy*100))
```

Output:




epoch_loss
epoch_loss
0.7
0.65
0.6
0.55
0
1
2
3
4