Artificial Intelligence & Machine Learning LAB Manual

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

1. write a program to demonstrate python NumPy and pandas' functions.

```
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(m(5))
print(data)

list(map(lambda x:x*x ,data['Maths']))

a=list(filter(lambda x:x%2,data['Maths']))

from functools import reduce
b=reduce(lambda x,y:x+y,data['Science'])
print(b)
```

```
     Maths  English  Science  History
sum     24       20       23       20
min      7        4        7        5
max      9       10        8        9
15
   Maths  English  Science  History
0      9        4        8        9
1      8       10        7        6
2      7        6        8        5
23
```

2.Data visualization in python using matplotlib for MTCARS dataset:

Create the following plots to visualize/summarize the data and customize appropriately.

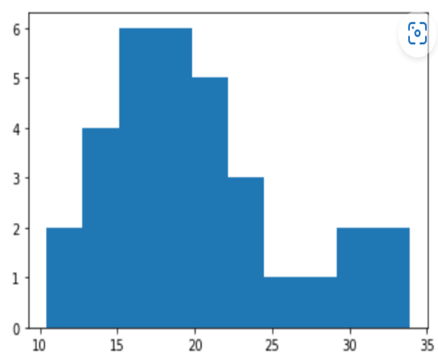
1. **histogram to check the frequency distribution of the variable 'mpg' (Miles per gallon) and note down the interval having the highest frequency.**
2. **scatter plot to determine the relation between weight of the car and mpg.**
3. **bar plot to check the frequency distribution of transmission type 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

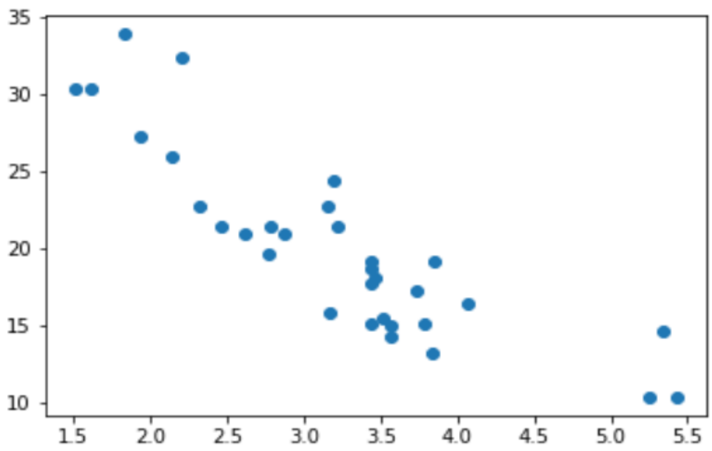
```
print(plt.hist(x=df['mpg']))
```

```
(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>)
```



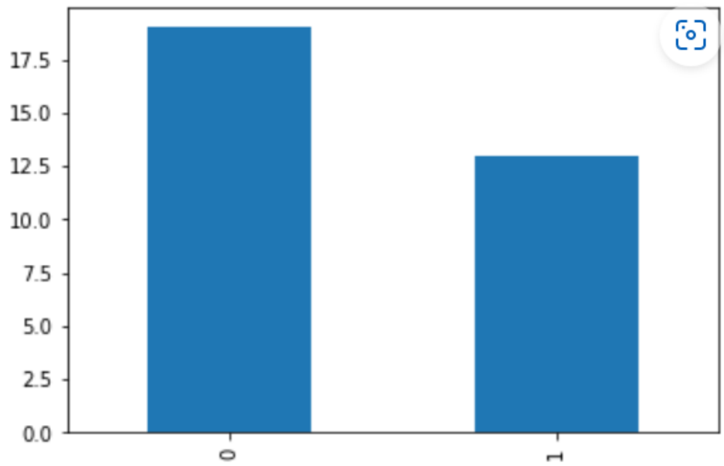

```
print(plt.scatter(x='wt',y='mpg',data=df))
```

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



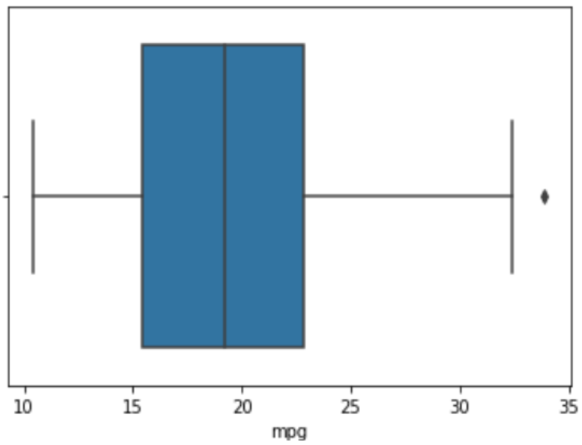
print(df['am'].value_counts().plot(kind='bar'))

```
<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')
```

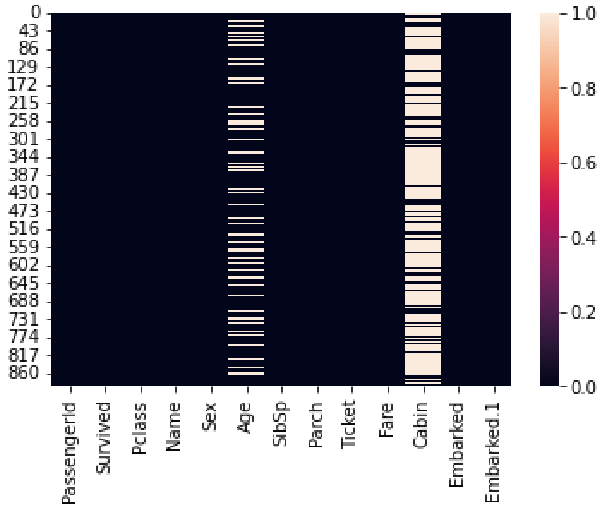
```
data.info()
```

```
<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
```

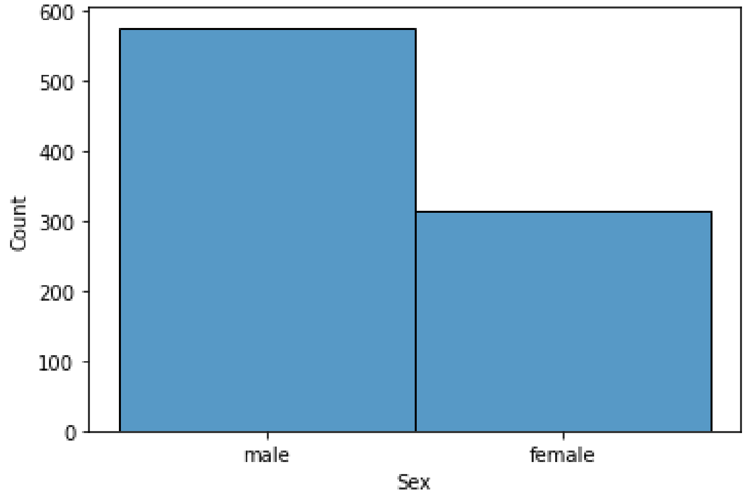
```
data.describe()
```

	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

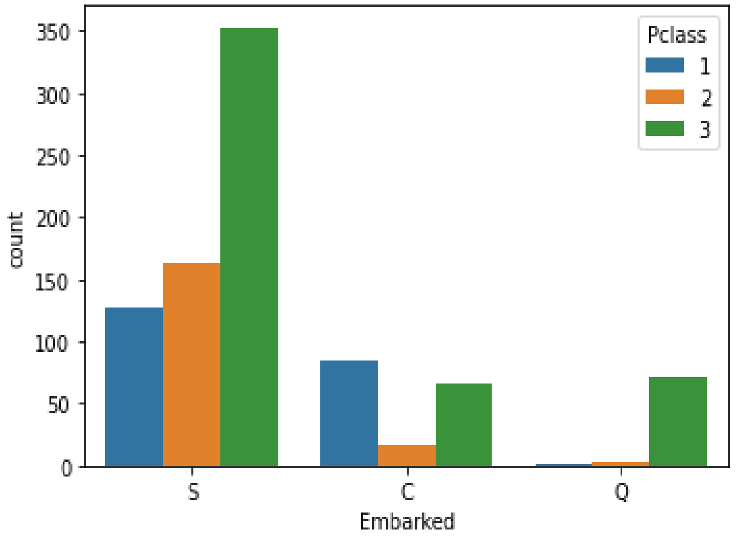
```
sns.heatmap(data.isna())
```



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



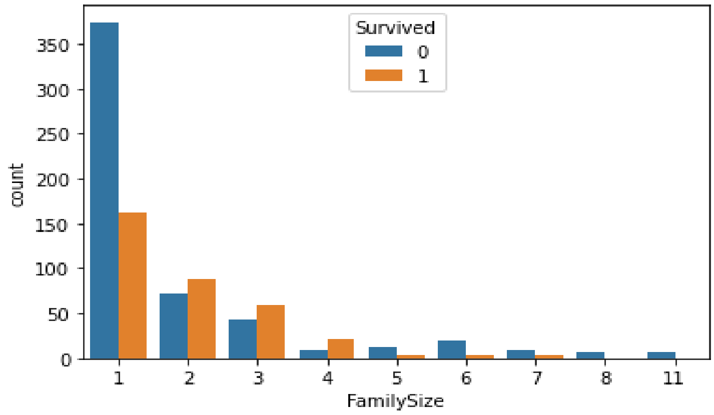

```
g=sns.countplot(x='Embarked', hue='Pclass', data=data)
```



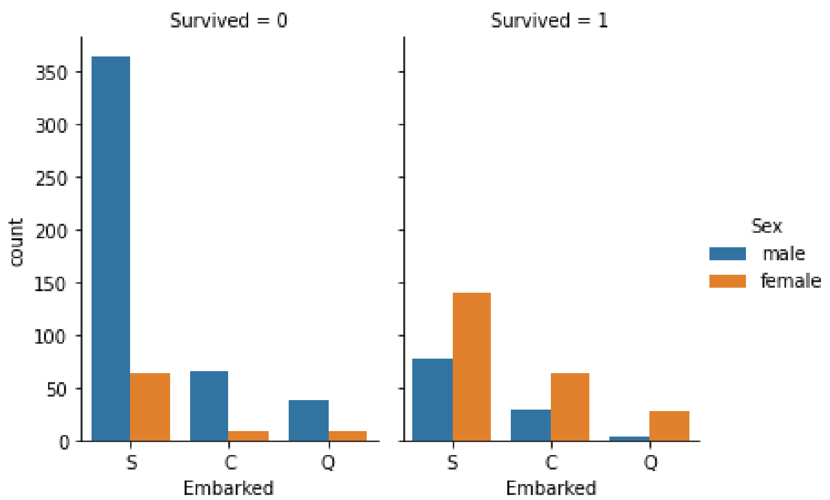

```
def add_family(df):
    df['FamilySize']=df['SibSp'] + df['Parch'] +1
    return df
data=add_family(data)
data.head(10)
```

	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
df=pd.read_csv("titanic.csv")
df.head()

#Checking missing values
df.isna().sum()

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

df['Embarked’] =df['Embarked'].astype('category')
df['Embarked’] =df['Embarked'].cat.codes

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

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

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

output:

```
0      2
1      0
2      2
3      2
4      2
      ..
886    2
887    2
888    2
889    0
890    1
Name: Embarked, Length: 891, dtype: int8
0      2
1      0
2      2
3      2
4      2
      ..
886    2
887    2
888    2
889    0
890    1
Name: Embarked, Length: 891, dtype: int8
```

output:

```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex        Age  \
0                              Braund, Mr. Owen Harris    male  22.000000
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.000000
2                               Heikkinen, Miss. Laina  female  26.000000
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.000000
4                             Allen, Mr. William Henry    male  35.000000
..                                                 ...     ...        ...
886                              Montvila, Rev. Juozas    male  27.000000
887                       Graham, Miss. Margaret Edith  female  19.000000
888           Johnston, Miss. Catherine Helen "Carrie"  female  29.699118
889                              Behr, Mr. Karl Howell    male  26.000000
890                                Dooley, Mr. Patrick    male  32.000000

     SibSp  Parch            Ticket     Fare  Embarked
0        1      0         A/5 21171   7.2500         2
1        1      0          PC 17599  71.2833         0
2        0      0  STON/O2. 3101282   7.9250         2
3        1      0            113803  53.1000         2
4        0      0            373450   8.0500         2
..     ...    ...               ...      ...       ...
886      0      0            211536  13.0000         2
887      0      0            112053  30.0000         2
888      1      2        W./C. 6607  23.4500         2
889      0      0            111369  30.0000         0
890      0      0            370376   7.7500         1

[891 rows x 11 columns]
```

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("athlete_events.csv")
data.head(10)

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

data.info()

data['Weight'].skew()
plt.hist(data['Weight'])
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']<lower,low
er,data['Weight']))

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

output:

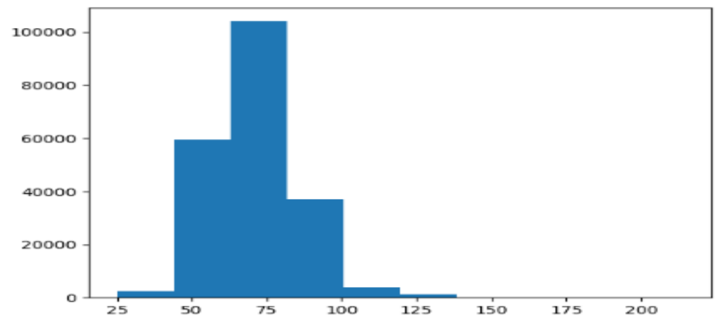
	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	NaN
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	NaN
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	NaN
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating	Speed Skating Women's 500 metres	NaN

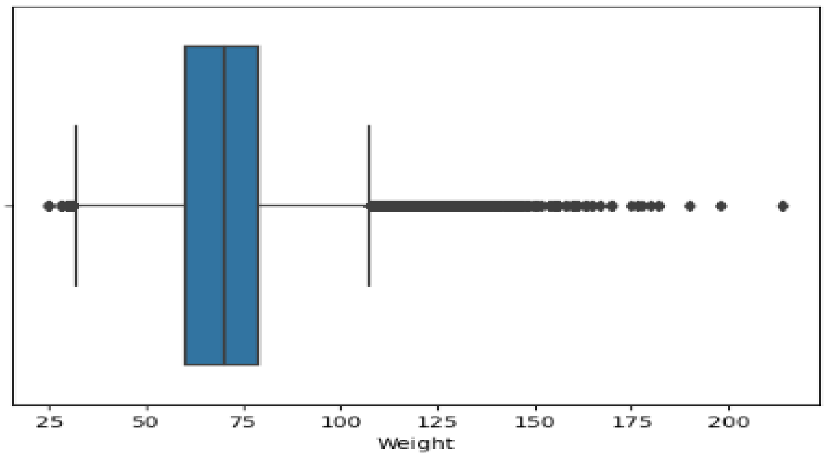
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271116 entries, 0 to 271115
Data columns (total 15 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   ID      271116 non-null  int64
 1   Name    271116 non-null  object
 2   Sex     271116 non-null  object
 3   Age     261642 non-null  float64
 4   Height  271116 non-null  float64
 5   Weight  208241 non-null  float64
 6   Team    271116 non-null  object
 7   NOC     271116 non-null  object
 8   Games   271116 non-null  object
 9   Year    271116 non-null  int64
 10  Season  271116 non-null  object
 11  City    271116 non-null  object
 12  Sport   271116 non-null  object
 13  Event   271116 non-null  object
 14  Medal   39783 non-null   object
dtypes: float64(3), int64(2), object(10)
memory usage: 31.0+ MB
```

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

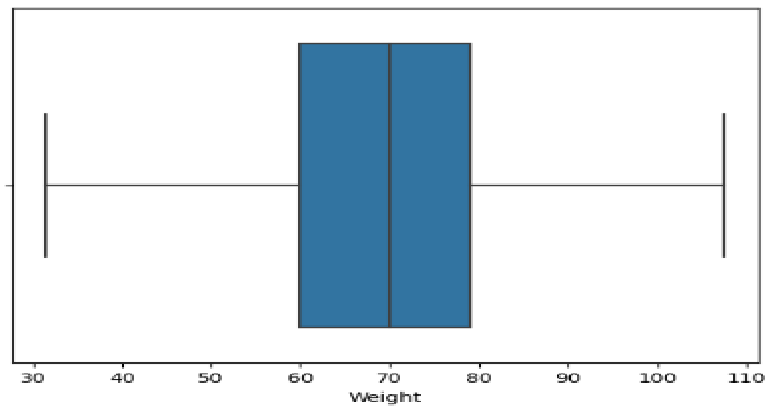
```
0.7971690270264297
```

```
(array([2.53900e+03, 5.95230e+04, 1.03919e+05, 3.68800e+04, 3.87600e+03,
        1.25300e+03, 2.00000e+02, 4.10000e+01, 7.00000e+00, 3.00000e+00]),
 array([ 25. ,  43.9,  62.8,  81.7, 100.6, 119.5, 138.4, 157.3, 176.2,
        195.1, 214. ]),
 <BarContainer object of 10 artists>)
```






```
0.390822515385823
```

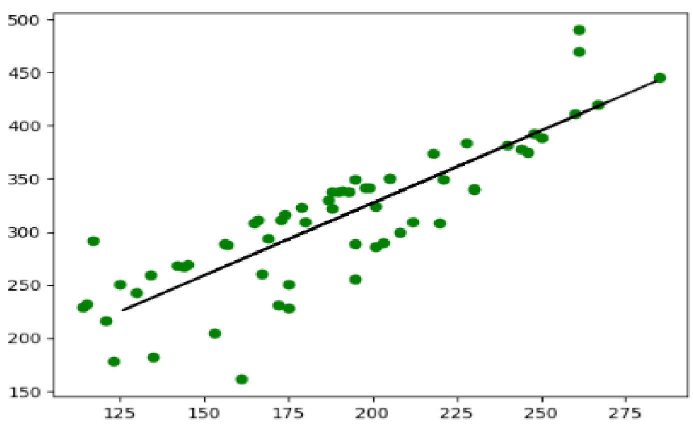


6.Build Simple Linear Regression Machine Learning Model to analysis relationship between CIE and SEE

```
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

```
 #   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---> 29.367841250713436
MSE---> 1593.0385277231082
score---> 0.44915939374374436
 RMSE---> 39.91288673753263
```



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

```
import pandas as pd
import numpy as np

df=pd.read_csv("Housing .csv")
df.head()

df=pd.get_dummies(df)
e=df.drop(['mainroad_no','guestroom_no','basement_yes','hotwaterheating_yes','aircon
ditioning_yes'],axis=1)

x=df.iloc[:,1:]
y=df.iloc[:,0]


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,y)
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:

```
MAE---> 740452.0626450425
MSE---> 974326300393.968
score---> 0.6410748483263938
 RMSE---> 987079.6829000018
```

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

```
import pandas as pd
data=pd.read_csv('Breast_cancer.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(xtrain,ytrain)

y_pred=model.predict(xtest)
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_sore_)
```

output:

```
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
None
The accuracy of the model built is 92.98245614035088
Fitting 10 folds for each of 648 candidates, totalling 6480 fits
```

```
C:\Users\SPTINT-29\anaconda4\lib\site-packages\sklearn\model_selection\_validation.py:372: FitFailedWarning:
720 fits failed out of a total of 6480.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
720 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\SPTINT-29\anaconda4\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\SPTINT-29\anaconda4\lib\site-packages\sklearn\tree\_classes.py", line 937, in fit
```

```
The accuracy of the model built is 87.71929824561403

{'criterion': 'entropy',
 'max_depth': 5,
 'min_samples_leaf': 2,
 'min_samples_split': 8}

 0.9627053140096619
```

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("HEART_DISEASE.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))
```

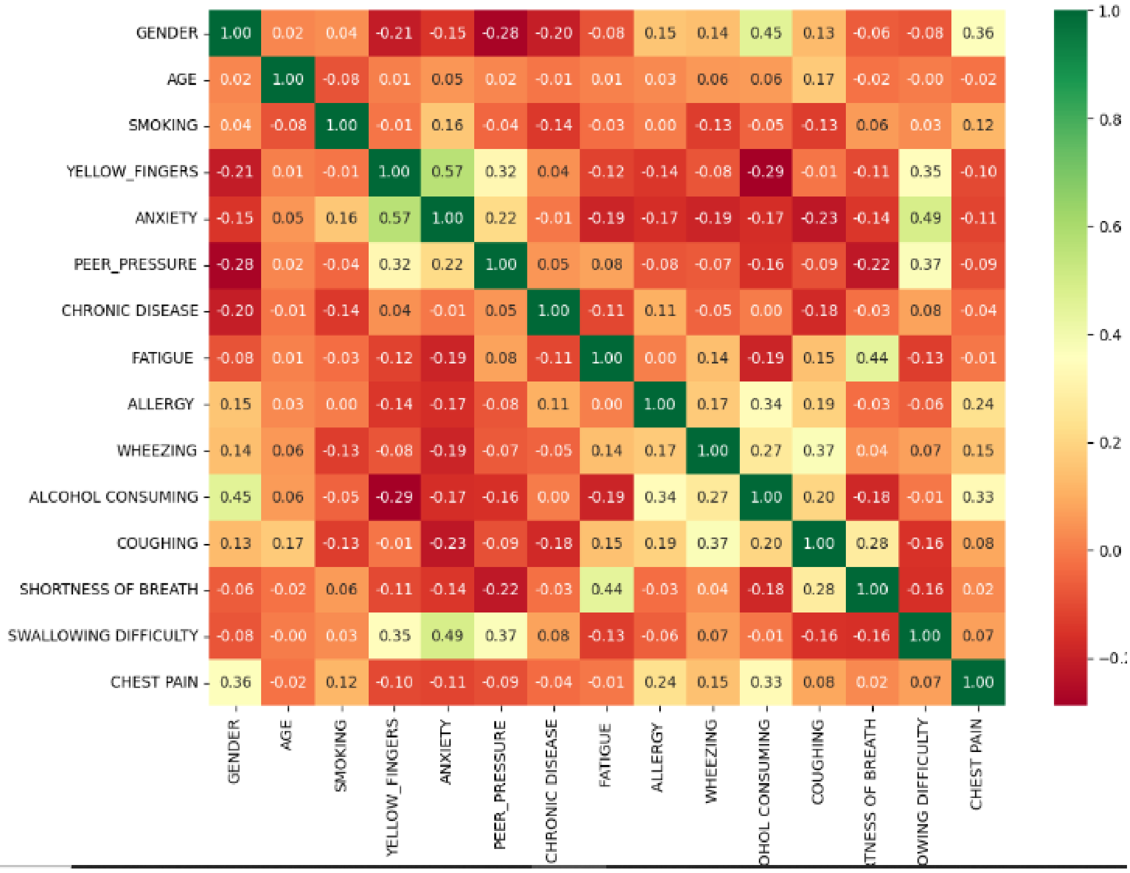
output:

```
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("C:/Users/SPTINT-29/Desktop/survey_lung_cancer_SVM.csv")
data.head()
data['GENDER']=data['GENDER'].map({'M':1,'F':0})

from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
new_data=scaler.fit_transform(data. drop(labels=['LUNG_CANCER'],axis=1))
corrmat=data.corr()

f,ax=plt.subplots(figsize=[12,8])
sns.heatmap(corrmat,annot=True,fmt='.2f',cmap='RdYlGn',ax=ax)
plt.show()

x=new_data
y=data['LUNG_CANCER'].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,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", classification_report(y_test,y_pred))
print('Accuracy',accuracy_score(y_test,y_pred))
```

output:




```
Classification_report           precision    recall  f1-score   support

         NO       0.62      0.42      0.50        12
        YES       0.87      0.94      0.90        50

   accuracy                           0.84        62
  macro avg       0.75      0.68      0.70        62
weighted avg       0.82      0.84      0.83        62

Accuracy 0.8387096774193549
```

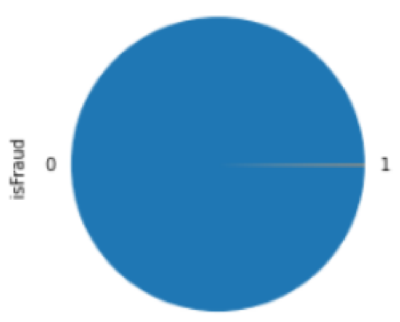
11.Build a supervised machine learning program for Credit Card Fraud Detection using Random Forest Classifier.

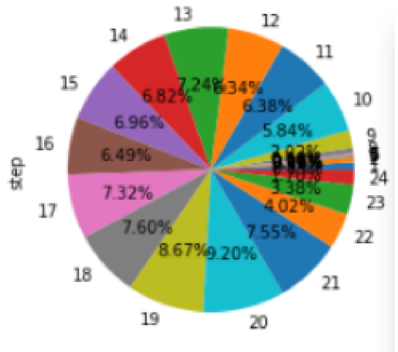
```
import pandas as pd
df=pd. read_csv("creditcard.csv")

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

del df['nameOrig']
del df['nameDest']

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

df['step']=df['step']%24+1
df=df.sort_values(by='step')
df['step'].value_counts(). sort_index().plot.pie(autopct='%1.2f%%')

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

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


pip install imblearn

from imblearn.over_sampling import SMOTE
sm=SMOTE (random_state=42)
x_sm,y_sm=sm.fit_resample(x,y)
print(f'''Shape of X before SMOTE: {x.shape}
Shape of X after SMOTE: {x_sm.shape}''')
print('\nBalance of positive and negative classes (%):')
y_sm.value_counts(normalize=True) * 100

from sklearn.model_selection import train_test_split
x_train,x_test, y_train, y_test = train_test_split(x_sm,y_sm,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  oldbalanceOrig  newbalanceOrig  oldbalanceDest  \
1024631     1     3  1134.00          4061.0         2927.00             0.0
1025443     1     3  1874.21          1026.0            0.00             0.0
1025444     1     3  7067.63         19101.0        12033.37             0.0
1025445     1     3  3403.72         21573.0        18169.28             0.0
1025446     1     3  1255.74         39699.0        38443.26             0.0

         newbalanceDest  isFraud
1024631             0.0        0
1025443             0.0        0
1025444             0.0        0
1025445             0.0        0
1025446             0.0        0
step              0
type              0
amount            0
oldbalanceOrig    0
newbalanceOrig    0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
dtype: int64
<AxesSubplot:ylabel='isFraud'>            <AxesSubplot:ylabel='step'>
```



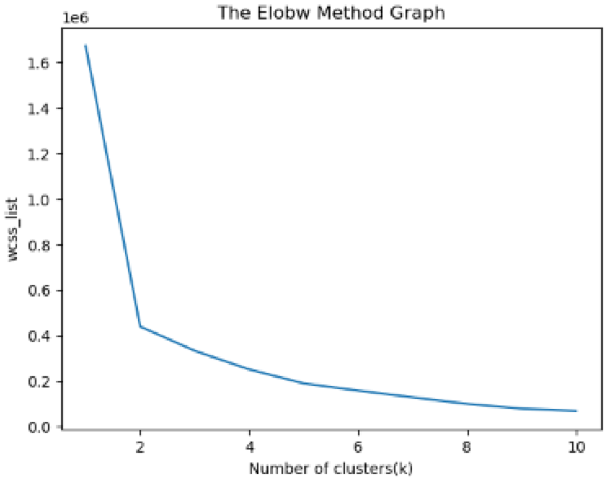


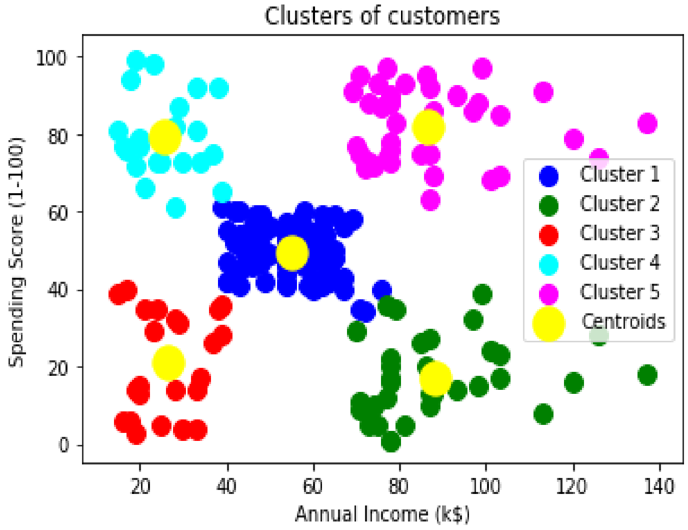

```
step   type   amount   oldbalanceOrig   newbalanceOrig   oldbalanceDest   newbalanceDest   isFrau
```

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_predict== 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()
```



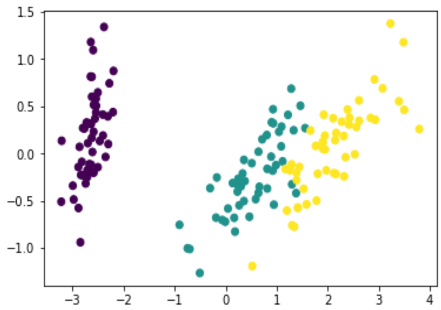


The output image is clearly showing the five different clusters with different colors. The clusters are formed between two parameters of the dataset; Annual income of customer and Spending. We can change the colors and labels as per the requirement or choice. We can also observe some points from the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




```
(150, 4)
(150,)
[[ 0.36138659 -0.08452251  0.85667061  0.3582892 ]
 [ 0.65658877  0.73016143 -0.17337266 -0.07548102]]
(150, 2)
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

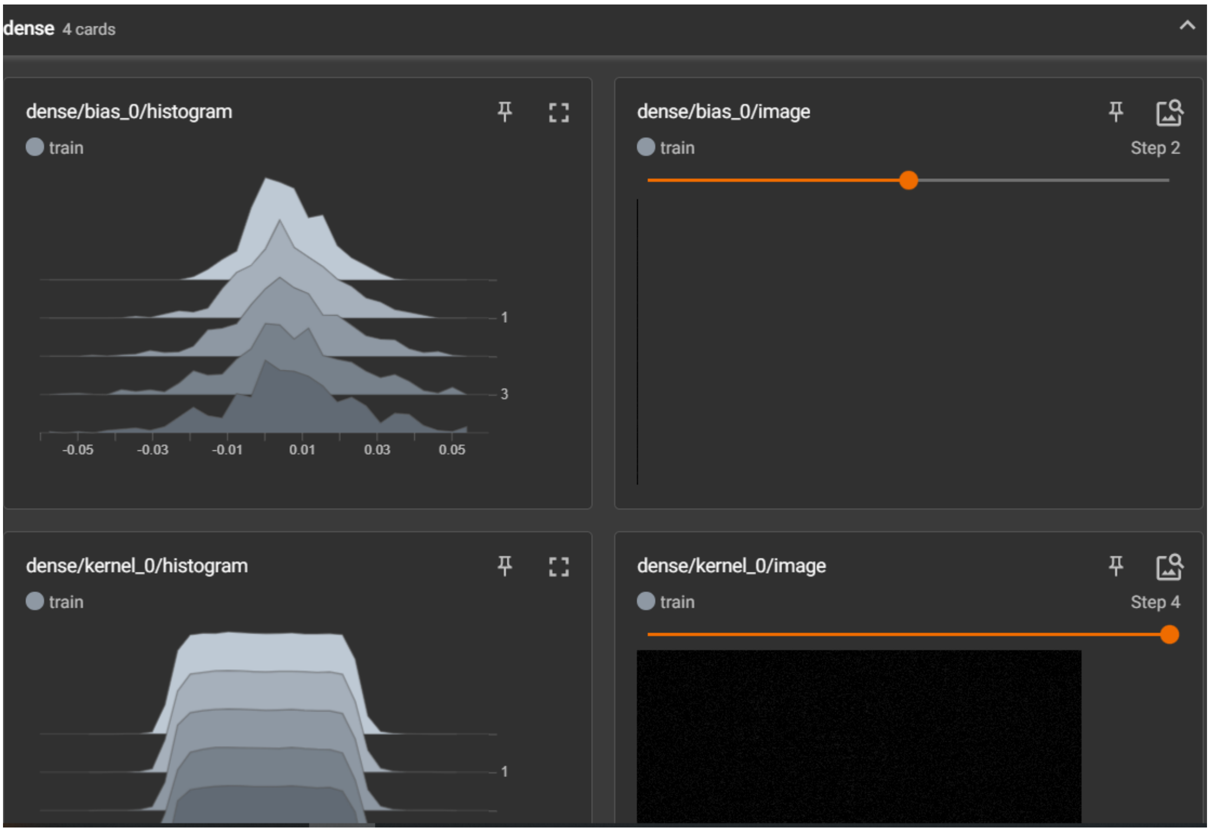
#Install python

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

```
Epoch 1/5
1500/1500 [==============================] - 17s 11ms/step - loss: 0.6977 - accuracy: 0.8210 - val_loss: 0.3615 - val_accuracy:
0.9057
Epoch 2/5
1500/1500 [==============================] - 14s 10ms/step - loss: 0.3689 - accuracy: 0.8969 - val_loss: 0.2958 - val_accuracy:
0.9188
Epoch 3/5
1500/1500 [==============================] - 14s 9ms/step - loss: 0.3128 - accuracy: 0.9116 - val_loss: 0.2619 - val_accuracy:
0.9280
Epoch 4/5
1500/1500 [==============================] - 14s 9ms/step - loss: 0.2786 - accuracy: 0.9214 - val_loss: 0.2383 - val_accuracy:
0.9342
Epoch 5/5
1500/1500 [==============================] - 14s 9ms/step - loss: 0.2520 - accuracy: 0.9299 - val_loss: 0.2190 - val_accuracy:
```



15.program to Build NLP pipeline for text processing using NLTK

```
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', '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'),
 ('definitely', 'RB'),
 ('watch', 'JJ'),
 ('part', 'NN'),
 ('It', 'PRP'),
 ('change', 'VBZ'),
 ('view', 'NN'),
 ('matrix', 'NN'),
 ('Are', 'NNP'),
```

```
['The', 'first', 'time', 'see', 'The', 'Second', 'Renaissance', 'may', 'look', 'boring', 'Look', 'least', 'twice', 'definitel
y', 'watch', 'part', 'It', 'change', 'view', 'matrix', 'Are', 'human', 'people', 'one', 'started', 'war', 'Is', 'AI', 'bad', 't
hing']
```

17. 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
```