NAME: GHODASARA KRIYANSHI
EMAIL: kriyanshighodasara@gmail.com
TOPIC: INTERNSHIP PROJECT
ROLE: DATA SCIENCE

LIBRARIES IMPORTED DURING TASK

- NumPy
- Pandas
- Matplotlib
- Seaborn
- Sklearn
- Labelencoder
- Train_test_split
- Standardscaler
- KNeighborsclassifier

TASK-1
TITANIC SURVIVAL PREDICTION

- Use the Titanic dataset to build a model that predicts whether a passenger on the Titanic survived or not. This is a classic beginner project with readily available data.
- The dataset typically used for this project contains information about individual passengers, such as their age, gender, ticket class, fare, cabin, and whether or not they survived.

Solution

```
import pandas as pd
con=pd.read_csv('/task-1 dataset.csv')
con.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

```
[ ] import statsmodels.api as sm
    Y=con['Survived']
    X=con['PassengerId']
    X.head()
```

```
0    892
1    893
2    894
3    895
4    896
Name: PassengerId, dtype: int64
```

The dataset has been imported and by using head we get the information about the columns.

```
X=sm.add_constant(X)
X.head()
```

	const	PassengerId
0	1.0	892
1	1.0	893
2	1.0	894
3	1.0	895
4	1.0	896

Constant has been added.

OLS method has been applied, std error has been calculated.

```
model=sm.OLS(Y,X,missing='drop')
model_result=model.fit()
model_result.summary()
```

OLS Regression Results

Dep. Variable:	Survived	R-squared:	0.001
Model:	OLS	Adj. R-squared:	-0.002
Method:	Least Squares	F-statistic:	0.2249
Date:	Sat, 06 Jan 2024	Prob (F-statistic):	0.636
Time:	14:37:53	Log-Likelihood:	-287.11
No. Observations:	418	AIC:	578.2
Df Residuals:	416	BIC:	586.3
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.4656	0.216	2.152	0.032	0.040	0.891
PassengerId	-9.267e-05	0.000	-0.474	0.636	-0.000	0.000

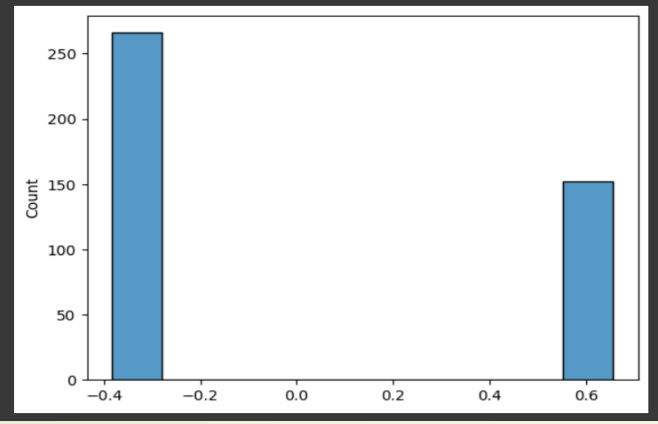
Omnibus:	2486.513	Durbin-Watson:	2.172
Prob(Omnibus):	0.000	Jarque-Bera (JB):	71.324
Skew:	0.567	Prob(JB):	3.25e-16
Kurtosis:	1.323	Cond. No.	1.02e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.02e+04. This might indicate that there are strong multicollinearity or other numerical problems.

```
[ ] from scipy import stats
    mu, std=stats.norm.fit(model_result.resid)
    mu, std
```

```
(2.640099728414846e-16, 0.4809157121331379 4)
```

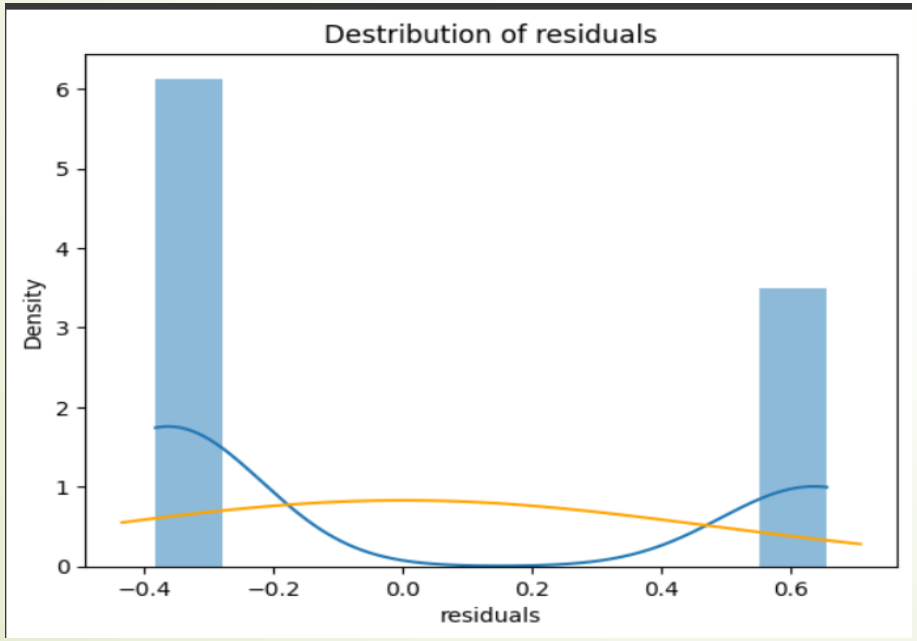
```
import seaborn as sns
sns.histplot(model_result.resid);
```



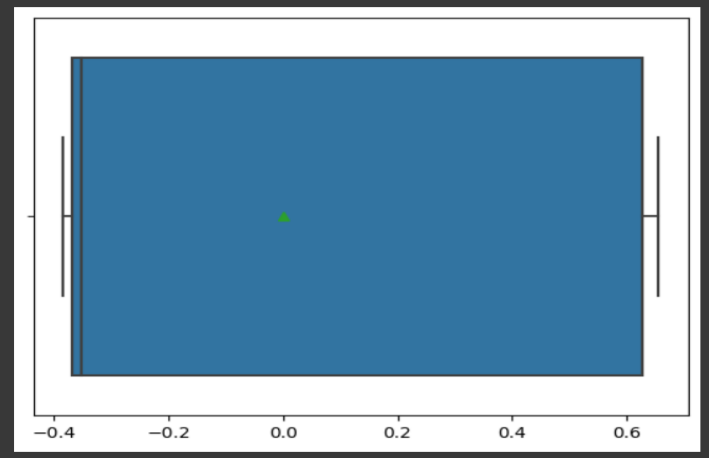
Histogram plot has been applied

Residuals and density graph with the combination of bar graph and line graph. Box plot has been also plotted with the residuals.

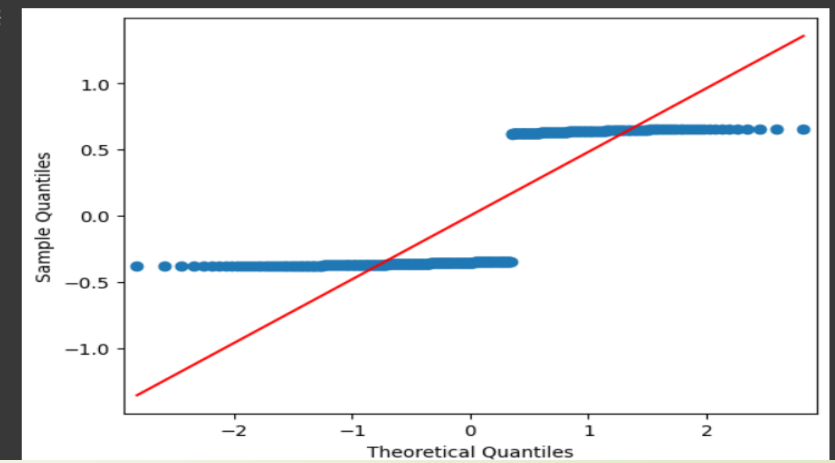
```
import matplotlib.pyplot as plt
import numpy as np
fig, ax=plt.subplots()
sns.histplot(x=model_result.resid, ax=ax, stat='density', linewidth=0,kde=True)
ax.set(title='Destribution of residuals',xlabel='residuals')
xmin, xmax=plt.xlim()
x=np.linspace(xmin, xmax, 100)
p=stats.norm.pdf(x, mu, std)
sns.lineplot(x=x, y=p, color='orange', ax=ax)
plt.show()
```

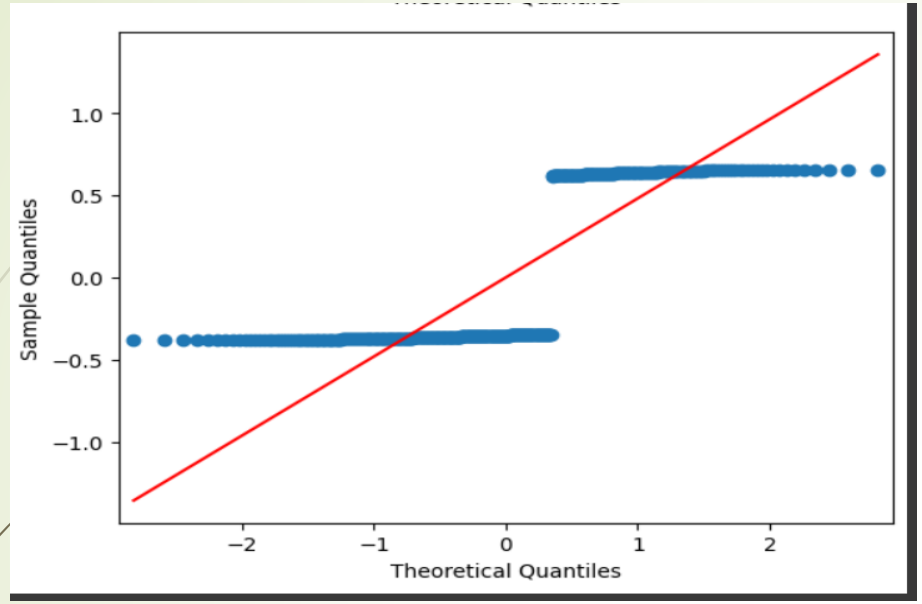



```
sns.boxplot(x=model_result.resid, showmeans=True);
```




```
sm.qqplot(model_result.resid, line='s')
```

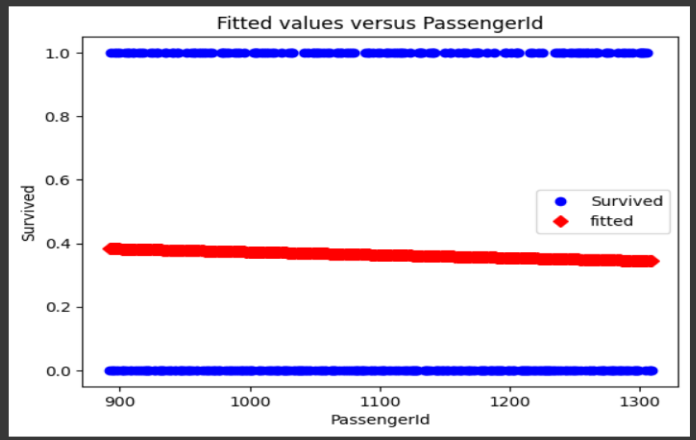




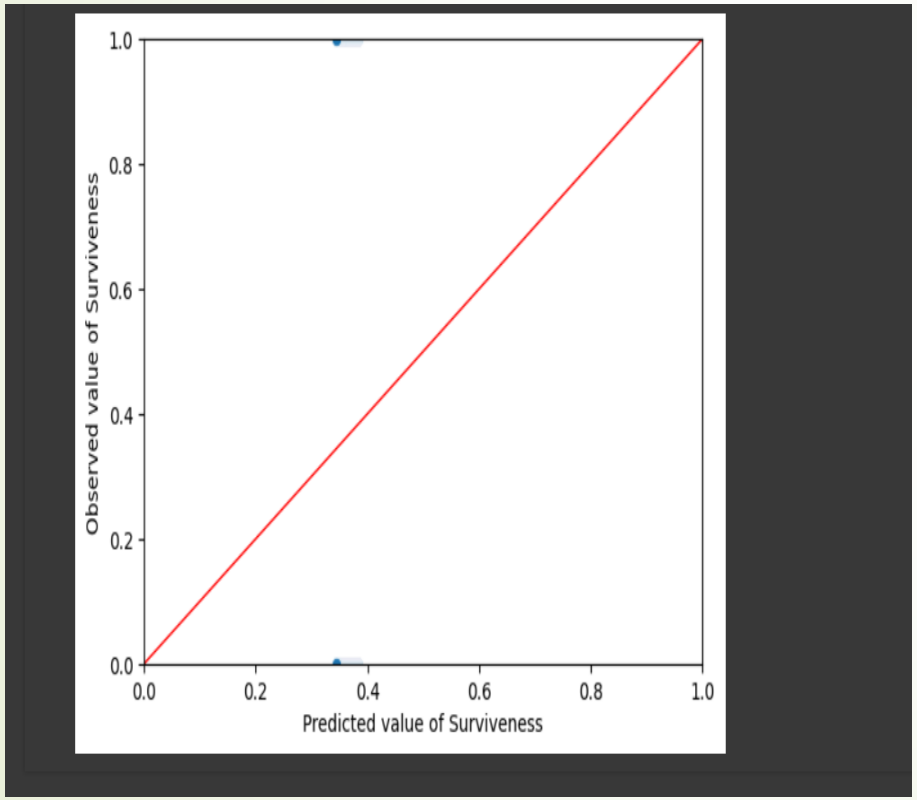

```
[ ] model_result.fittedvalues
```

```
0      0.382958
1      0.382865
2      0.382772
3      0.382680
4      0.382587
          ...
413    0.344686
414    0.344593
415    0.344500
416    0.344408
417    0.344315
Length: 418, dtype: float64
```

```
sm.graphics.plot_fit(model_result,1,vlines=False);
```




```
Y_max = Y.max()
Y_min = Y.min()

ax = sns.scatterplot(x=model_result.fittedvalues, y=Y)
ax.set(ylim=(Y_min, Y_max))
ax.set(xlim=(Y_min, Y_max))
ax.set_xlabel("Predicted value of Surviveness")
ax.set_ylabel("Observed value of Surviveness")

X_ref = Y_ref = np.linspace(Y_min, Y_max, 100)
plt.plot(X_ref, Y_ref, color='red', linewidth=1)
plt.show()
```



Predicted value by using the observed value. The method applied is linear regression and it was affected just by one factor.

TASK-3
IRIS FLOWER CLASSIFICATION

- The Iris flower dataset consists of three species: setosa, versicolor,and virginica. These species can be distinguished based on their measurements. Now, imagine that you have the measurements of Iris flowers categorized by their respective species. Your objective is to train a machine learning model that can learn from these measurements and accurately classify the Iris flowers into their respective species.
- Use the Iris dataset to develop a model that can classify iris flowers into different species based on their sepal and petal measurements. This dataset is widely used for introductory classification task.

SOLUTION

```
[20] import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
     import sklearn

[21] dataset = pd.read_csv('/content/task-flower.csv')
     X = dataset.iloc[:, [1, 2, 3]].values
     y = dataset.iloc[:, -1].values

[22] from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     X[:,0] = le.fit_transform(X[:,0])

[23] from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

     from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_train = sc.fit_transform(X_train)
     X_test = sc.transform(X_test)
```

The dataset has been uploaded. KNN method has been used by using machine learning which means that by using available data we can identify the type of flower.

```
[23] from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

```
[24] from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_train = sc.fit_transform(X_train)
     X_test = sc.transform(X_test)
```

```
[25] from sklearn.neighbors import KNeighborsClassifier
     classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
     classifier.fit(X_train, y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
[26] y_pred = classifier.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac = accuracy_score(y_test,y_pred)
```

TASK-4
SALES PREDICTION USING PYTHON

- Sales prediction involves forecasting the amount of a product that customers will purchase, taking into account various factors such as advertising expenditure, target audience segmentation, and advertising platform selection.
- In businesses that offer products or services, the role of a Data Scientist is crucial for predicting future sales. They utilize machine learning techniques in Python to analyse and interpret data, allowing them to make informed decisions regarding advertising costs. By leveraging these predictions, businesses can optimize their advertising strategies and maximize sales potential. Let's embark on the journey of sales prediction using machine learning in Python.

SOLUTION

```
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

```
[15] url='/content/advertising-csv.csv'
     names=['TV','Radio','Newspaper','Sales']
     dataset=read_csv(url,names=names)
```

```
[16] print(dataset.shape)

     (200, 4)
```

First importing all the libraries and use the dataset. Number of column and rows are calculated.

```
print(dataset.head(20))
```

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
5	8.7	48.9	75.0	7.2
6	57.5	32.8	23.5	11.8
7	120.2	19.6	11.6	13.2
8	8.6	2.1	1.0	4.8
9	199.8	2.6	21.2	15.6
10	66.1	5.8	24.2	12.6
11	214.7	24.0	4.0	17.4
12	23.8	35.1	65.9	9.2
13	97.5	7.6	7.2	13.7
14	204.1	32.9	46.0	19.0
15	195.4	47.7	52.9	22.4
16	67.8	36.6	114.0	12.5
17	281.4	39.6	55.8	24.4
18	69.2	20.5	18.3	11.3
19	147.3	23.9	19.1	14.6

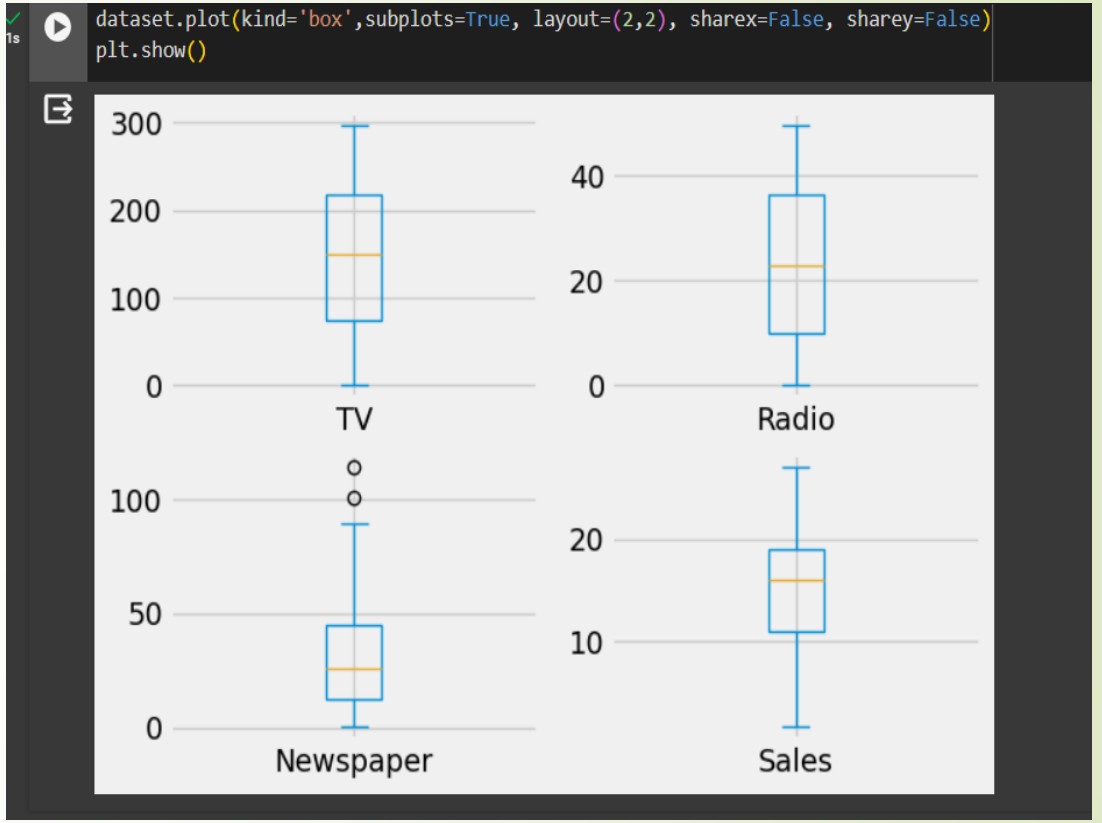
By using head we can upload the data in the form of columns.

```
print(dataset.describe())
```

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

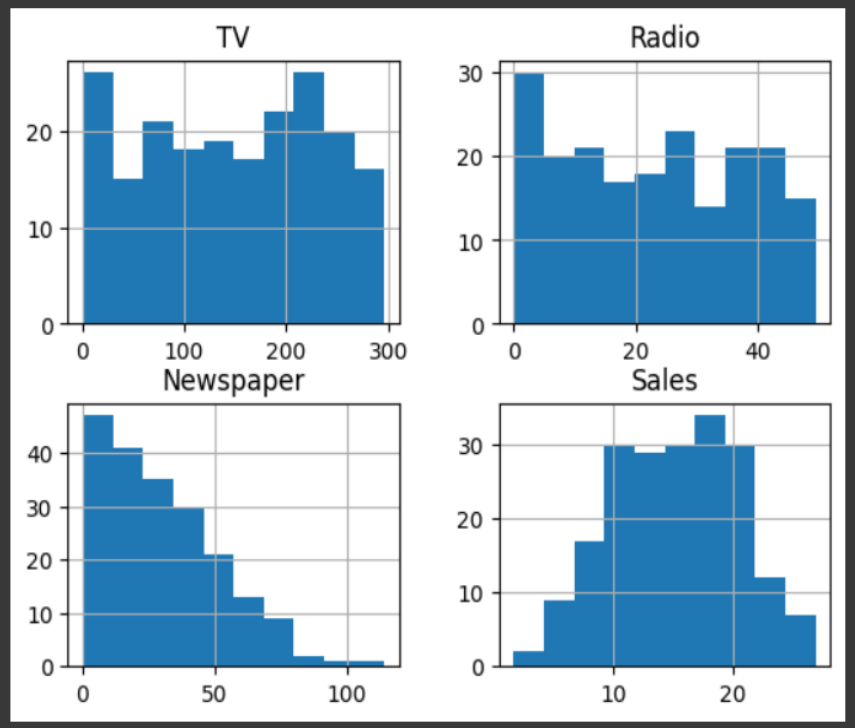
Dataset's descriptive analysis like mean, median, mode, count and quartile.

```
dataset.plot(kind='box',subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```



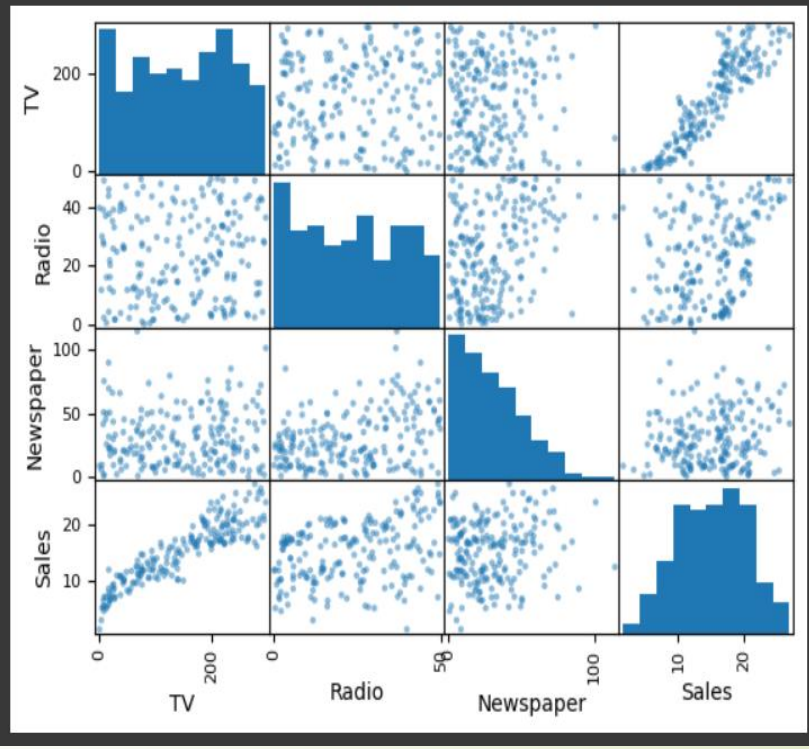
Box-plot of TV, radio, newspaper and sales are uploaded.

```
dataset.hist()
plt.show()
```



Histogram has been uploaded of TV, radio, newspaper and sales.

```
scatter_matrix(dataset)
plt.show()
```



Scatter plot of tv, radio newspaper and sales with sales, newspaper, radio and tv.

```
[45] array = dataset.values
     X = array[:,0:4]
     y = array[:,3]
     X_train, X_validation, Y_train, Y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

```
[64] models = []
     models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
     results = []
```

```
[51] import sklearn.linear_model as linear_model
```

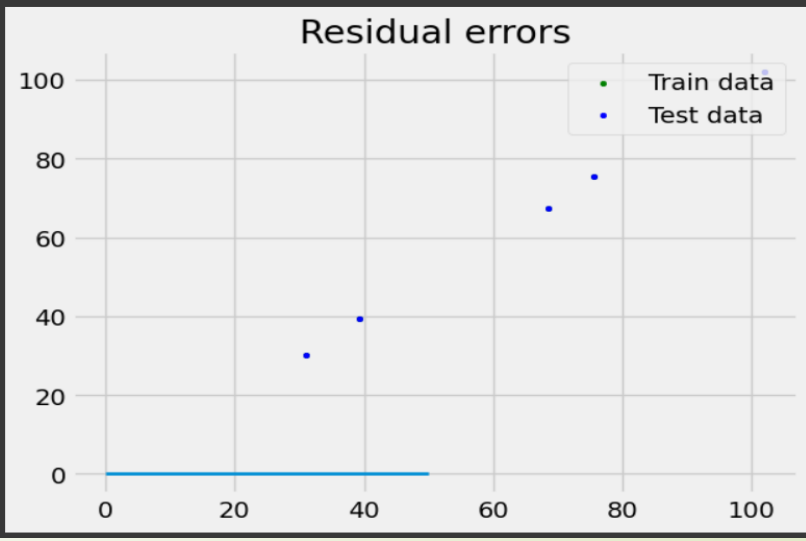
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
logr = linear_model.LinearRegression()
logr.fit(X, y)
```

```
LinearRegression
LinearRegression()
```

```
[62] y_pred = logr.predict(X)
     print('Accuracy of linear regression classifier on test set: {:.2f}'.format(logr.score(X, y)))

     Accuracy of linear regression classifier on test set: 0.99
```

```
[74] print('coefficients:', logr.coef_)
     print('Variance score: {}'.format(logr.score(X, Y_test)))

     coefficients: [ 0.02725994  0.05401091 -0.02058411  5.67893596]
     Variance score: -115.90835614678292
```

```
plt.style.use('fivethirtyeight')
plt.scatter(logr.predict(X),
            logr.predict(X) - y_train,
            color="green", s=10,
            label='Train data')
plt.scatter(logr.predict(X),
            logr.predict(X) - y_train,
            color="blue", s=10,
            label='Test data')
plt.hlines(y=0, xmin=0, xmax=50, linewidth=2)
plt.legend(loc='upper right')
plt.title("Residual errors")
plt.show()
```



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

from sklearn.preprocessing import LabelEncoder
from sklearn.impute import KNNImputer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score

warnings.filterwarnings('ignore')

df = pd.read_csv('/content/advertising-csv.csv')

X = df.iloc[:,1:2].values
y = df.iloc[:,2].values
X
```

```
array([[33.2],
       [ 5.7],
       [14.8],
       [ 1.9],
       [ 7.3],
       [49. ],
       [40.3],
       [25.8],
       [13.9],
       [ 8.4],
       [23.3],
       [39.7],
       [21.1],
       [11.6],
       [43.5],
       [ 1.3],
       [36.9],
       [18.4],
       [18.1],
       [35.8],
       [18.1],
       [36.8],
       [14.7],
       [ 3.4],
       [37.6],
       [ 5.2],
       [23.6],
       [10.6],
       [11.6],
       [20.9],
       [20.1],
       [ 7.1],
       [ 3.4],
       [48.9],
       [30.2],
       [ 7.8],
       [ 2.3],
       [10. ],
       [ 2.6],
       [ 5.4],
       [ 5.7],
       [43. ],
       [21.3],
       [45.1],
       [ 2.1],
       [28.7],
       [13.9],
       [12.1],
       [41.1],
       [10.8],
       [ 4.1],
       [42. ],
       [35.6],
       [ 3.7],
       [ 4.9],
       [ 9.3],
       [42. ],
       [ 8.6]])
```

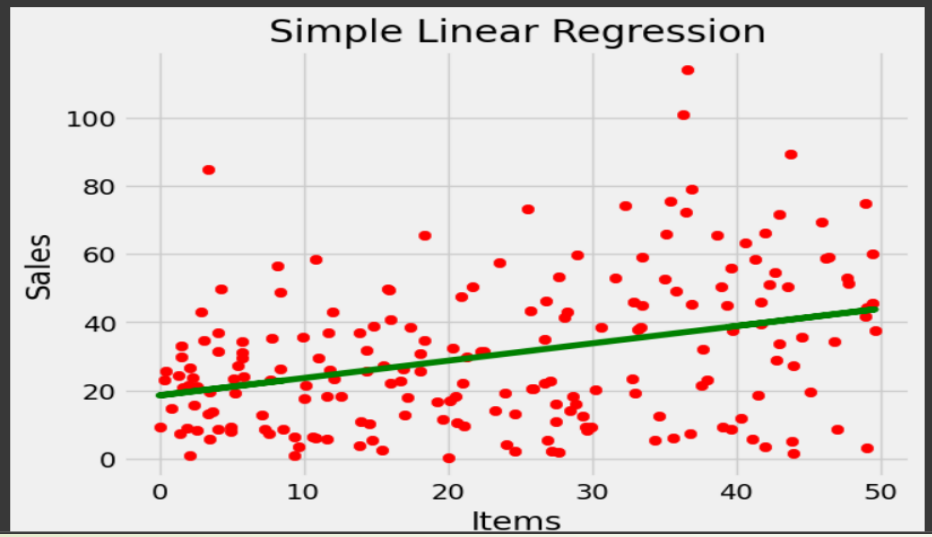
```
from sklearn.linear_model import LinearRegression
lin_reg=LinearRegression()
lin_reg.fit(X,y)
```

```
LinearRegression()
```

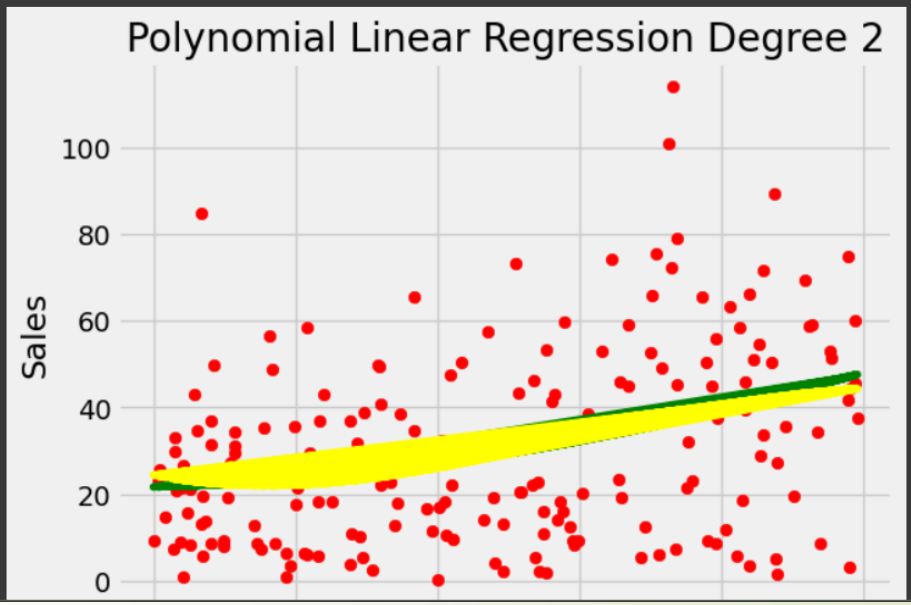
```
from sklearn.preprocessing import PolynomialFeatures
poly_reg2=PolynomialFeatures(degree=2)
X_poly=poly_reg2.fit_transform(X)
lin_reg_2=LinearRegression()
lin_reg_2.fit(X_poly,y)

poly_reg3=PolynomialFeatures(degree=3)
X_poly3=poly_reg3.fit_transform(X)
lin_reg_3=LinearRegression()
lin_reg_3.fit(X_poly3,y)
```

```
LinearRegression
LinearRegression()
```

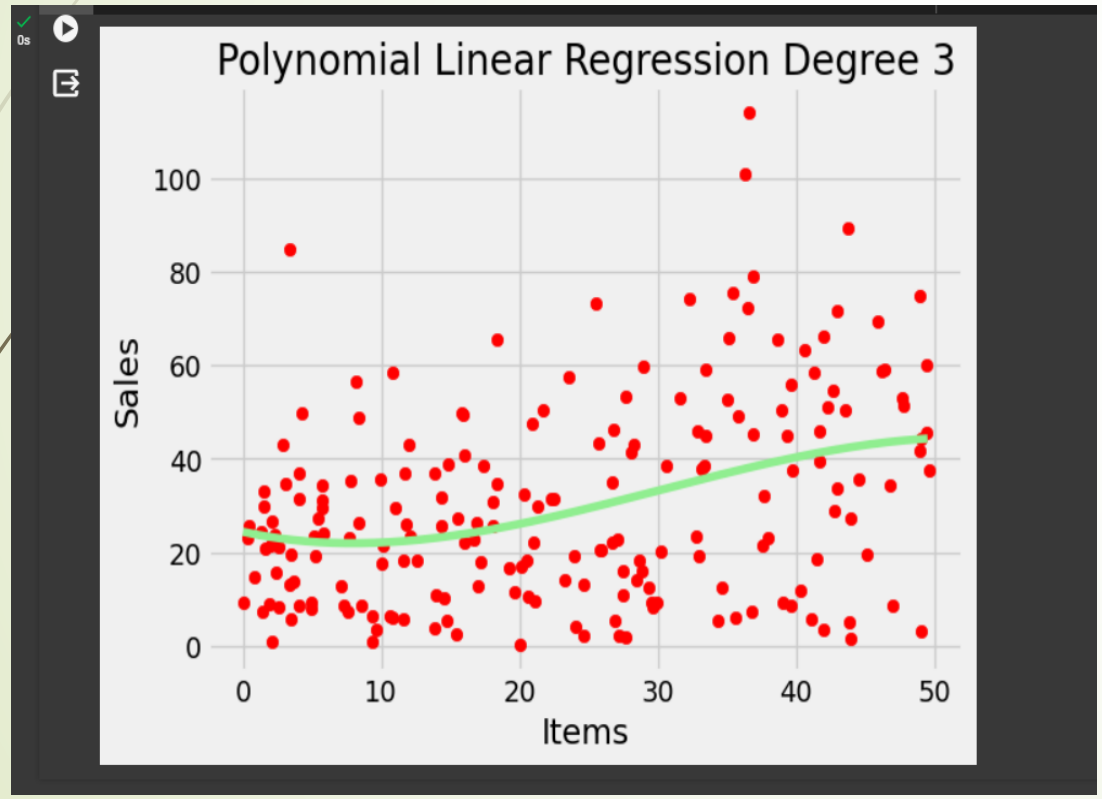
```
plt.scatter(X,y,color='red')
plt.plot(X,lin_reg.predict(X),color='green')
plt.title('Simple Linear Regression')
plt.xlabel('Items')
plt.ylabel('Sales')
plt.show()
```




```
plt.style.use('fivethirtyeight')
plt.scatter(X,y,color='red')
plt.plot(X,lin_reg_2.predict(poly_reg2.fit_transform(X)),color='green')
plt.plot(X,lin_reg_3.predict(poly_reg3.fit_transform(X)),color='yellow')
plt.title('Polynomial Linear Regression Degree 2')
plt.xlabel('Items')
plt.ylabel('Sales')
plt.show()
```



```
plt.style.use('fivethirtyeight')
X_grid=np.arange(min(X),max(X),0.1)
X_grid=X_grid.reshape((len(X_grid),1))
plt.scatter(X,y,color='red')
plt.plot(X_grid,lin_reg_3.predict(poly_reg3.fit_transform(X_grid)),color='lightgreen')
plt.title('Polynomial Linear Regression Degree 3')
plt.xlabel('Items')
plt.ylabel('Sales')
plt.show()
```



Linear regression has been applied with multiple values and it is 0.99 accurate to use this method.

THANKYOU