Aadith Lasar

LY CORE-2

2203262

MACHINE LEARNING LAB-3

Aim:- Prepare the Correlated dataset of your own example or Download the any one of the UCI ML data Set in which find the correlated data and Find the best fit line for the data

Objectives:-

1. To learn linear regression
2. To learn and understand correlated datasets

Theory:-

Linear Regression:

Linear regression is a statistical approach for modelling relationship between a dependent variable with a given set of independent variables.

Simple linear regression is an approach for predicting a **response** using a **single feature**.
It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x).

Multiple linear regression attempts to model the relationship between **two or more features** and a response by fitting a linear equation to observed data. Clearly, it is nothing but an extension of Simple linear regression.

Logistic regression:

Logistic regression is another technique borrowed by machine learning from the field of statistics.

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.
1 / (1 + e^-value)

Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

Correlated datasets:

They follow the property of Correlation

Correlation is a statistical measure that indicates the extent to which two or more variables fluctuate together. A positive correlation indicates the extent to which those variables increase or decrease in parallel; a negative correlation indicates the extent to which one variable increases as the other decreases.

A correlation coefficient is a statistical measure of the degree to which changes to the value of one variable predict change to the value of another. When the fluctuation of one variable reliably predicts a similar fluctuation in another variable, there's often a tendency to think that means that the change in one causes the change in the other. However, correlation does not imply causation. There may be, for example, an unknown factor that influences both variables similarly.

Code:-

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
data = pd.read_csv("timed_delivery.csv")
data.head()
data.isnull().sum()
```

```
data.skew()
data.kurtosis()
data.corr()
data.info()
data.describe()
data.boxplot(["Delivery"])
data.boxplot(['Sorting'])
plt.scatter(data.Sorting, data.Delivery, marker="+") plt.xlabel("Sorting Time")
plt.ylabel("Delivery Time")
import seaborn as sns
sns.distplot(data['Sorting']) sns.distplot(data['Delivery'])
sns.pairplot(data)
import statsmodels.formula.api as smf
model = smf.ols("Delivery~Sorting", data=data).fit() sns.regplot(x="Sorting",
y="Delivery", data=data)
model.params
print(model.tvalues, '\n', model.pvalues)
rsqrt=(model.rsquared, model.rsquared_adj)
rsqrt
newdata = pd.Series([1, 11, 5, 13, 15])
newdata
data_pred = pd.DataFrame(newdata, columns = ['Sorting'])
model.predict(data_pred)
model.summary()
```

Output:-

jupyter Timed_delivery Last Checkpoint: 28 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Help | Notebook saved | Trusted | Python 3 (ipykernel)

Run | Code

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
#Aadith Lasar
#2203262
#LY-CORE2
```

In [3]:
```
data = pd.read_csv("timed_delivery.csv")
```

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

Out[4]:

	Delivery	Sorting
0	21.00	10
1	13.50	4
2	19.75	6
3	24.00	9
4	29.00	10

In [5]:
```
data.isnull().sum()
```

Out[5]:
```
Delivery    0
Sorting     0
dtype: int64
```

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

Out[6]:
```
Delivery    0.352390
Sorting     0.047115
dtype: float64
```

In [7]:
```
data.kurtosis()
```

Out[7]:
```
Delivery    0.317960
Sorting    -1.148455
dtype: float64
```

In [8]:
```
data.corr()
```

Out[8]:

	Delivery	Sorting
Delivery	1.000000	0.825997
Sorting	0.825997	1.000000

In [9]:
```
data.info()
data.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Delivery  21 non-null     float64
 1   Sorting   21 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 464.0 bytes
```

Out[9]:

	Delivery	Sorting
count	21.000000	21.000000
mean	16.790952	6.190476
std	5.074901	2.542028
min	8.000000	2.000000
25%	13.500000	4.000000
50%	17.830000	6.000000
75%	19.750000	8.000000
max	29.000000	10.000000

In [10]:
```
data.boxplot(['Sorting'])
```

Out[10]:
```
<Axes: >
```

jupyter Timed_delivery Last Checkpoint: 28 minutes ago (autosaved)

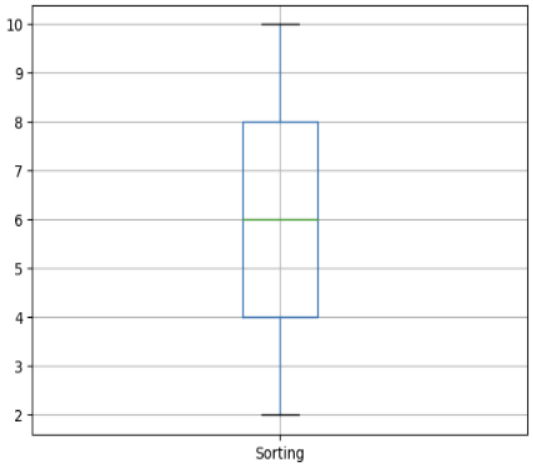
Logout

File Edit View Insert Cell Kernel Help

Trusted

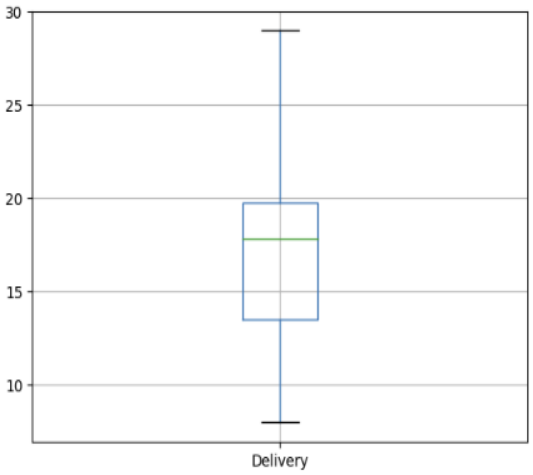
Python 3 (ipykernel)

Run Code



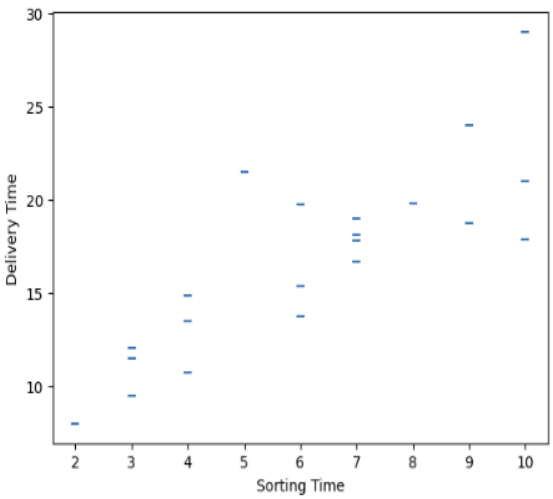
In [11]:
```
data.boxplot(["Delivery"])
```

Out[11]: <Axes: >



In [12]:
```
plt.scatter(data.Sorting, data.Delivery, marker="_")
plt.xlabel("Sorting Time")
plt.ylabel("Delivery Time")
```

Out[12]: Text(0, 0.5, 'Delivery Time')



In [13]:
```
import seaborn as sns
sns.distplot(data['Sorting'])
```

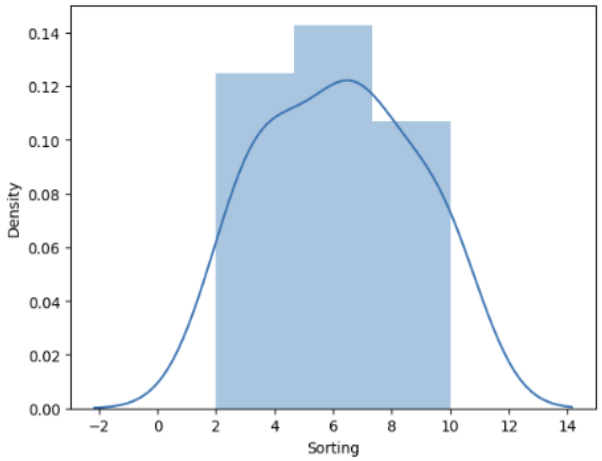
jupyter Timed_delivery Last Checkpoint: 29 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Help Trusted Python 3 (ipykernel)

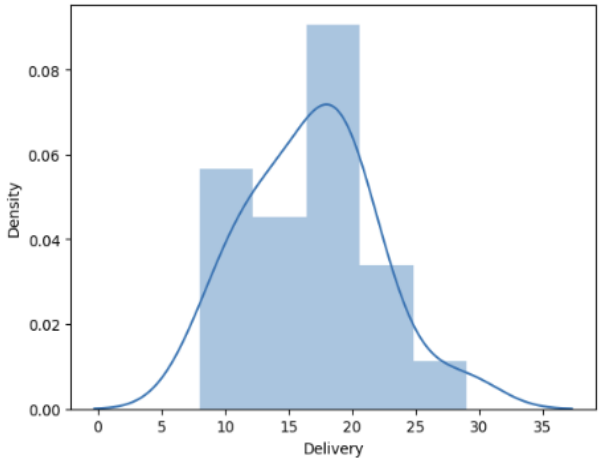
Run Code

Out[13]: <Axes: xlabel='Sorting', ylabel='Density'>



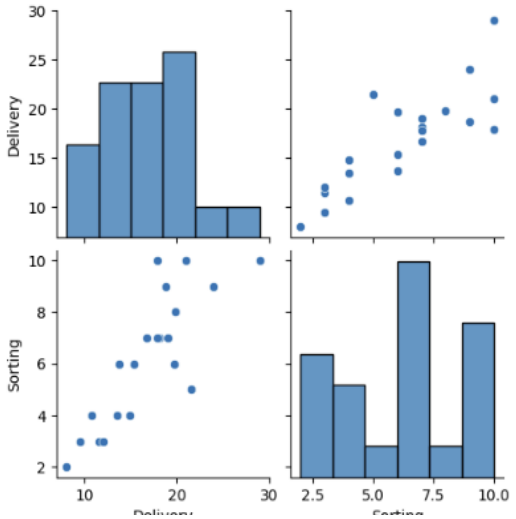
In [14]:
```
sns.distplot(data['Delivery'])
```

Out[14]: <Axes: xlabel='Delivery', ylabel='Density'>



In [15]:
```
sns.pairplot(data)
```

Out[15]: <seaborn.axisgrid.PairGrid at 0x295887b80>



jupyter Timed_delivery Last Checkpoint: 30 minutes ago (autosaved)

Logout

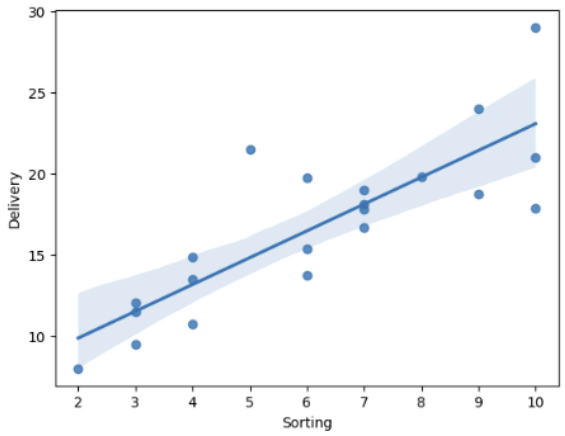
File Edit View Insert Cell Kernel Help

Trusted | Python 3 (ipykernel)

Run Code

In [16]:
```
import statsmodels.formula.api as smf
model = smf.ols("Delivery~Sorting", data=data).fit()
sns.regplot(x="Sorting", y="Delivery", data=data)
```

Out[16]: <Axes: xlabel='Sorting', ylabel='Delivery'>



In [17]:
```
model.params
```

Out[17]:
```
Intercept    6.582734
Sorting      1.649020
dtype: float64
```

In [18]:
```
rsqrt=(model.rsquared, model.rsquared_adj)
rsqrt
```

Out[18]: (0.6822714748417231, 0.6655489208860244)

In [19]:
```
newdata = pd.Series([1, 11, 5, 13, 15])
newdata
```

Out[19]:
```
0     1
1    11
2     5
3    13
4    15
dtype: int64
```

In [20]:
```
data_pred = pd.DataFrame(newdata, columns = ['Sorting'])
```

In [21]:
```
model.predict(data_pred)
```

Out[21]:
```
0     8.231754
1    24.721953
2    14.827833
3    28.019993
4    31.318032
dtype: float64
```

In [22]:
```
model.summary()
```

Out[22]:

OLS Regression Results

Dep. Variable:	Delivery	**R-squared:**	0.682
Model:	OLS	**Adj. R-squared:**	0.666
Method:	Least Squares	**F-statistic:**	40.80
Date:	Mon, 31 Jul 2023	**Prob (F-statistic):**	3.98e-06
Time:	01:28:17	**Log-Likelihood:**	-51.357
No. Observations:	21	**AIC:**	106.7
Df Residuals:	19	**BIC:**	108.8
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	6.5827	1.722	3.823	0.001	2.979	10.186
Sorting	1.6490	0.258	6.387	0.000	1.109	2.189

Omnibus:	3.649	**Durbin-Watson:**	1.248
Prob(Omnibus):	0.161	**Jarque-Bera (JB):**	2.086
Skew:	0.750	**Prob(JB):**	0.352
Kurtosis:	3.367	**Cond. No.**	18.3

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]: