120A3051
Shreya Idate
Batch: E3

EXPERIMENT NO. 6

Aim: Implementation of

a. Statistical Hypothesis Test using Scipy /Sci-kit learn.
b. Perform linear regression to find out relation between variables.

Theory:

Hypothesis testing is a statistical method that is used in making statistical decisions using experimental data. Hypothesis Testing is basically an assumption that we make about the population parameter.

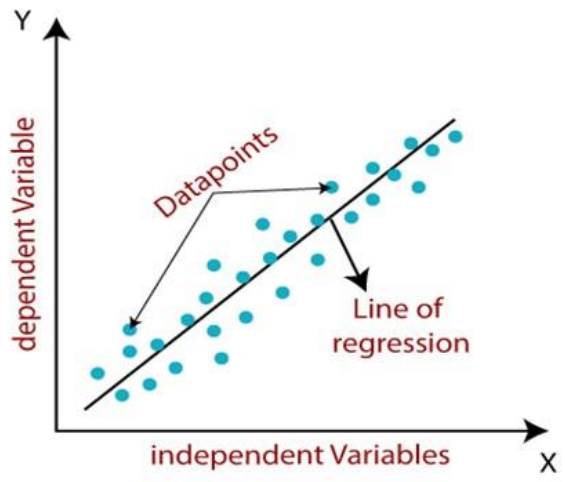
It is an essential procedure in statistics. A hypothesis test evaluates two mutually exclusive statements about a population to determine which statement is best supported by the sample data. When we say that a finding is statistically significant, it is because of hypothesis test.

Linear Regression

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

The linear regression model provides a sloped straight line representing the relationship between the variables.



Mathematically, we can represent a linear regression as:

$y= a0+a1x+ \varepsilon$

Here,

Y= Dependent Variable (Target Variable)

X= Independent Variable (predictor Variable)

a0= intercept of the line (Gives an additional degree of freedom)

a1 = Linear regression coefficient (scale factor to each input value).

ε = random error

The values for x and y variables are training datasets for Linear Regression model representation. The different values for weights or coefficient of lines (a0, a1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line. We can use the cost function to find the accuracy of the mapping function, which maps the input variable to the output variable. This mapping function is also known as Hypothesis function.

Program:

```
import numpy as np
import pandas as pd
```

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

```
#READ DATASET
```

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

```
train.head()
```

	x	y
0	24.0	21.549452
1	50.0	47.464463
2	15.0	17.218656
3	38.0	36.586398
4	87.0	87.288984

```
train.tail()
```

	x	y
695	58.0	58.595006
696	93.0	94.625094
697	82.0	88.603770
698	66.0	63.648685
699	97.0	94.975266

```
test = pd.read_csv("test.csv")
```

```
test.head()
```

	x	y
0	77	79.775152
1	21	23.177279
2	22	25.609262
3	20	17.857388
4	36	41.849864

```
test.tail()
```

	x	y
295	71	68.545888
296	46	47.334876
297	55	54.090637
298	62	63.297171
299	47	52.459467

```
train.sample
```

```
<bound method NDFrame.sample of          x          y
0     24.0   21.549452
1     50.0   47.464463
2     15.0   17.218656
3     38.0   36.586398
4     87.0   87.288984
5     36.0   32.463875
6     12.0   10.780897
7     81.0   80.763399
8     25.0   24.612151
9      5.0    6.963319
10    16.0   11.237573
11    16.0   13.532902
12    24.0   24.603239
```

```
train.shape
```

```
(700, 2)
```

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 2 columns):
x    700 non-null float64
y    699 non-null float64
dtypes: float64(2)
memory usage: 11.0 KB
```

```
train.isnull().count()
```

```
x    700
y    700
dtype: int64
```

```
train['x'].isnull().count()
```

```
700
```

```
train['x'].value_counts()
```

```
58.000000      16
49.000000      14
25.000000      13
24.000000      13
16.000000      13
50.000000      12
26.000000      12
75.000000      12
21.000000      11
54.000000      11
42.000000      11
48.000000      11
```

```
train['y'].value_counts()
```

```
48.093680    1
5.809947     1
82.889358    1
8.791140     1
26.928324    1
44.268005    1
85.860779    1
36.413996    1
2.116113     1
63.673482    1
```

```
train["x"].describe()
```

```
count     700.000000
mean       54.985939
std       134.681703
min         0.000000
25%        25.000000
50%        49.000000
75%        75.000000
max      3530.157369
Name: x, dtype: float64
```

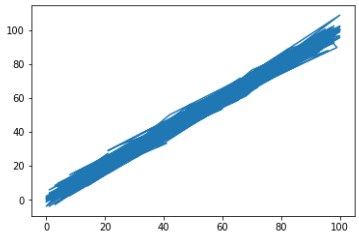
```
train["y"].describe()
```

```
count    699.000000
mean      49.939869
std       29.109217
min       -3.839981
25%       24.929968
50%       48.973020
75%       74.929911
max      108.871618
Name: y, dtype: float64
```

```
%matplotlib inline
plt.plot('x','y', data=train)
```

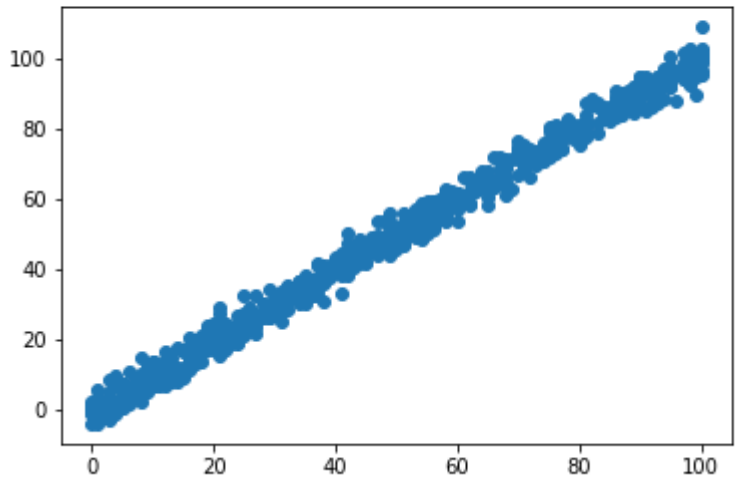
```
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\pyplot.py:3240: RuntimeWarning: Second argument 'y' is ambiguous: could b
e a color spec but is in data. Using as data.
Either rename the entry in data or use three arguments to plot.
  ret = ax.plot(*args, **kwargs)
```

```
[<matplotlib.lines.Line2D at 0x207badbde10>]
```



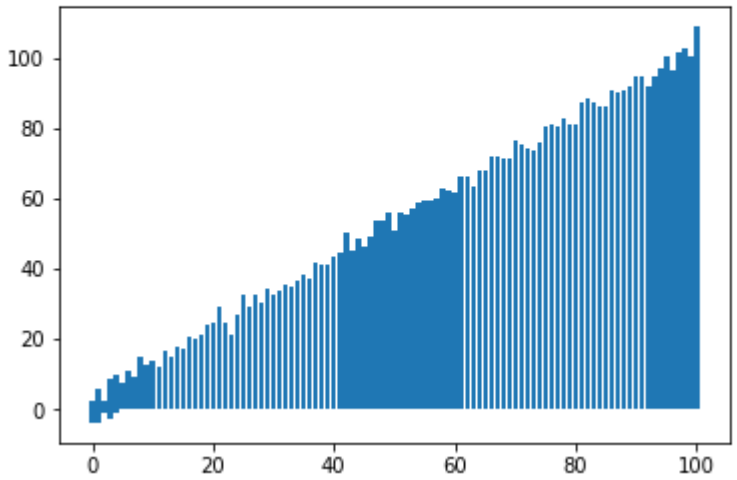

```
plt.scatter('x','y', data=train)
```

```
<matplotlib.collections.PathCollection at 0x207bb096518>
```




```
plt.bar('x','y', data=train)
```

```
<Container object of 700 artists>
```



```
#DROP NA
```

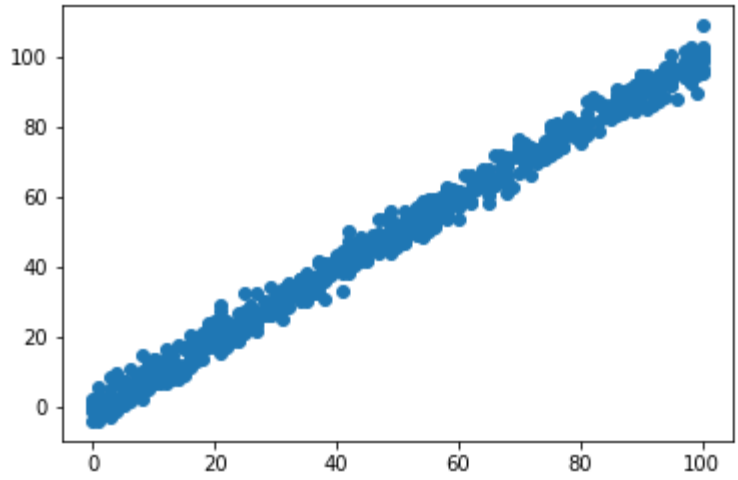
```
train = train.dropna()
```

```
train.shape
```

```
(699, 2)
```

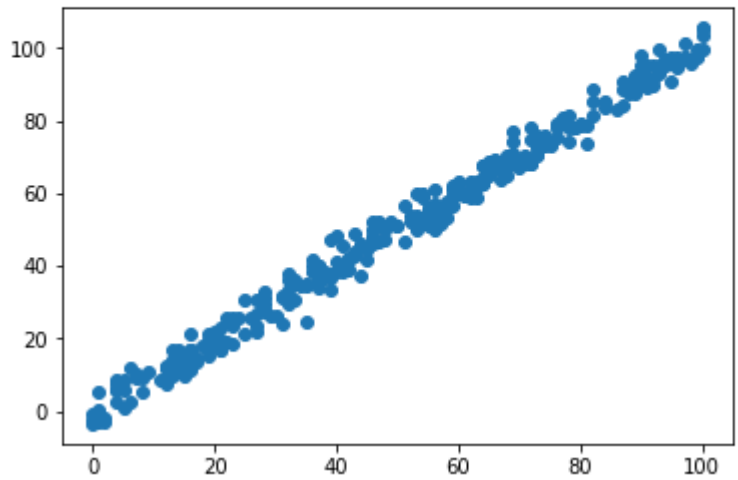
```
plt.scatter('x','y', data=train)
```

```
<matplotlib.collections.PathCollection at 0x207bbc39278>
```




```
plt.scatter('x','y', data=test)
```

```
<matplotlib.collections.PathCollection at 0x207bbe98898>
```



```
from sklearn import linear_model
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

slr = linear_model.LinearRegression()
```

```
x_train = np.array(train.iloc[:, :-1].values)
x_test = np.array(test.iloc[:, :-1].values)

y_train = np.array(train.iloc[:, :1].values)
y_test = np.array(test.iloc[:, :1].values)
```

```
slr.fit(x_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
# y=ax+b
# _coeff  (coefficient/slope/m)
# _intercept  (bias/intercept)

slr.coef_
```

```
array([[ 1.]])
```

```
slr.intercept_
```

```
array([ 0.])
```

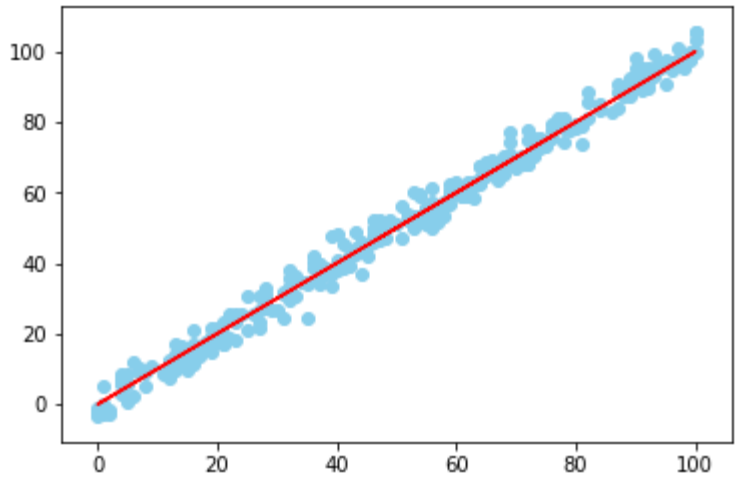
```
y_pred = slr.predict(x_test)
y_pred
```

```
array([[ 77.],
       [ 21.],
       [ 22.],
       [ 20.],
       [ 36.],
       [ 15.],
       [ 62.],
       [ 95.],
       [ 20.],
       [  5.],
       [  4.],
       [ 19.],
       [ 96.],
       [ 62.],
       [ 36.],
       [ 15.],
       [ 65.],
       [ 14.],
       [ 87.],
```

```
accuracy = slr.score(x_test,y_test)
print(accuracy)
```

```
1.0
```

```
plt.plot(x_test,y_pred, color ='red')
plt.scatter('x','y', data=test, color="skyblue")
plt.show()
```




```
mean_absolute_error(y_test, y_pred)
```

```
0.0
```

```
mean_squared_error(y_test, y_pred)
```

```
0.0
```

```
r2_score(y_test, y_pred)
```

```
1.0
```

Conclusion: Successfully performed linear regression.