Name: Ankit Sharma

Email: kumarankitx022@gmail.com

Mob no.: +91-7677241423

ASSIGNMENT – 02

TOPIC: INFERENTIAL STATISTICS AND HYPOTHESIS TESTING

Q.1. Write notes about significance level/ value and p-value.

Q.2. Create a spearman correlation coefficient matrix.

SIGNIFICANCE LEVEL/ VALUE:

The significance level, also denoted as alpha or α, is a measure of the strength of the evidence that must be present in your sample before you will reject the null hypothesis and conclude that the effect is statistically significant. The person conducting the hypothesis test determines the significance level before conducting the experiment.

The significance level is the probability of rejecting the null hypothesis when it is true. For example, a significance level of 0.05 indicates a 5% risk of concluding that a difference exists when there is no actual difference. Lower significance levels indicate that you require stronger evidence before you will reject the null hypothesis.

Common choices for the level of significance are $\alpha = 0.05$ and $\alpha = 0.01$.

P-VALUE:

P-value provides a convenient basis for drawing conclusions in hypothesis-testing applications. The *p*-value is a measure of how likely the sample results are, assuming the null hypothesis is true; the smaller the *p*-value, the less likely the sample results. If the *p*-value is less than α, the null hypothesis can be rejected; otherwise, the null hypothesis cannot be rejected. The *p*-value is often called the observed level of significance for the test. In other words, the evidence in your sample is strong enough to be able to reject the null hypothesis at the level.

In [1]:

```
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
%matplotlib inline
```

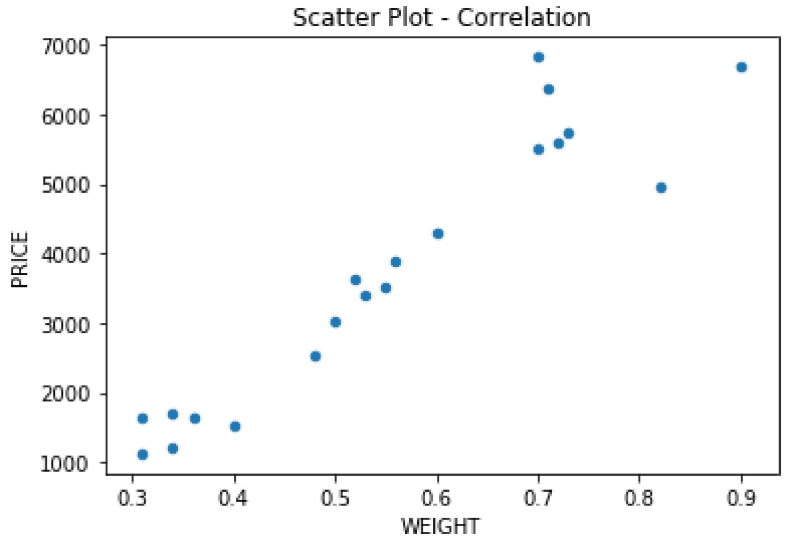
In [2]:

```
diamond = pd.read_excel('datasets/Inferential_Statistics.xlsx', sheet_name = 'Sheet1')
```

Correlation Coefficients

In [3]:

```
ScatterPlot = diamond.plot(kind='scatter',x='WEIGHT',y='PRICE',title='Scatter Plot - Correl
```



In [4]:

```
diamond['WEIGHT'].corr(diamond['PRICE']) #positive correlation
```

Out[4]:

```
0.9457713913032358
```

Correlation Coefficient Matrix

Here all the diagonal values will be 1 while correlation coefficient will be there for all the combination of the numerical variables. There are two methods of calculating Correlation Coefficient and its matrix – Pearson and Spearman.
diamond.corr(method= 'pearson')

In [5]:

```
diamond.corr(method= 'spearman')
```

Out[5]:

	IDNO.	WEIGHT	PRICE
IDNO.	1.000000	0.928115	0.921805
WEIGHT	0.928115	1.000000	0.925104
PRICE	0.921805	0.925104	1.000000

In [6]:

```
diamond.corr(method= 'pearson')
```

Out[6]:

	IDNO.	WEIGHT	PRICE
IDNO.	1.000000	0.893409	0.925019
WEIGHT	0.893409	1.000000	0.945771
PRICE	0.925019	0.945771	1.000000

ASSIGNMENT – 03

TOPIC: DATA - PREPROCESSING

Q.1. To find the details about ColumnTransformer and OneHotEncoder.

Q.2. Find details about LabelEncoder.

Q.3. When to use OneHotEncoder and LabelEncoder?

Ans. 1) COLUMN TRANSFORMER:

Applies transformers to columns of an array or pandas DataFrame.

This estimator allows different columns or column subsets of the input to be transformed separately and the features generated by each transformer will be concatenated to form a single feature space. This is useful for heterogeneous or columnar data, to combine several feature extraction mechanisms or transformations into a single transformer.

The ColumnTransformer constructor takes quite a few arguments,

- **Name**: a name for the column transformer, which will make setting of parameters and searching of the transformer easy.
- **Transformer**: here we're supposed to provide an estimator. We can also just "passthrough" or "drop" if we want. But since we're encoding the data we'll use the OneHotEncoder here. Remember that the estimator you use here needs to support fit and transform.
- **Column(s)**: the list of columns which you want to be transformed.

Creating an object for ColumnTransformer:

```
COlumnTransformer = ColumnTransformer([('encoder', OneHotEncoder(), [0])], remainder = 'passthrough').
```

As you can see from the snippet above, we'll name the transformer simply "encoder." We're using the OneHotEncoder() constructor to provide a new instance as the estimator. And then we're specifying that only the first column has to be transformed.

ONE HOT ENCODER (OneHotEncoder):

What one hot encoding does is, it takes a column which has categorical data, which has been label encoded or column transformed and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has what value.

It simply creates additional features based on the number of unique values in the categorical feature. Every unique value in the category will be added as a feature. In other words, it creates dummy variables making the value in the category as features.

Consider below example:

ID	Country	Population
1	Japan	127185332
2	U.S	326766748
3	India	1354051854
4	China	1415045928
5	U.S	326766748
6	India	1354051854

In our example, on applying OneHotEncoder we'll get four new columns, one for each country- Japan, U.S, India, and China.

For rows which have the first column value as Japan, the 'Japan' column will have a '1' and the other three columns will have '0's. Similarly, for rows which have the first column value as the U.S, the 'U.s' column will have a '1' and the other three columns will have '0's and so on.

ID	Country_Japan	Country_U.S	Country_India	Country_China	Population
1	1	0	0	0	127185332
2	0	1	0	0	326766748
3	0	0	1	0	1354051854
4	0	0	0	1	1415045928
5	0	1	0	0	326766748
6	0	0	1	0	1354051854

Ans. 2) LABEL ENCODER:

Label Encoding is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering.

Label Encoding in Python can be achieved using Sklearn Library. Sklearn provides a very efficient tool for encoding the levels of categorical features into numeric values. `LabelEncoder` encode labels with a value between 0 and n_classes-1 where n is the number of distinct labels. If a label repeats it assigns the same value to as assigned earlier.

- Encode target labels with value between 0 and n_classes-1.
- This transformer should be used to encode target values, *i.e.* y, and not the input X.

Consider below example:

ID	Country	Population
1	Japan	127185332
2	U.S	326766748
3	India	1354051854
4	China	1415045928
5	U.S	326766748
6	India	1354051854

This time we will try encoding this data using Label Encoder. On applying Label Encoder we will get a result as seen below,

ID	Country	Population
1	0	127185332
2	1	326766748
3	2	1354051854
4	3	1415045928
5	1	326766748
6	2	1354051854

That's all label encoding is about. But depending on the data, label encoding introduces a new problem. For example, we have encoded a set of country names into numerical data. This is actually categorical data and there is no relation, of any kind, between the rows.

The problem here is since there are different numbers in the same column, the model will misunderstand the data to be in some kind of order, 0 < 1 <2.

The model may derive a correlation like as the country number increases the population increases but this clearly may not be the scenario in some other data or the prediction set. To overcome this problem, we use One Hot Encoder.

Ans. 3) When to use a Label Encoding vs. One Hot Encoding:

This question generally depends on your dataset and the model which you wish to apply. But still, a few points to note before choosing the right encoding technique for your model:

We apply One-Hot Encoding when:

1. The categorical feature is **not ordinal** (like the countries above)
2. The number of categorical features is less so one-hot encoding can be effectively applied

We apply Label Encoding when:

1. The categorical feature is **ordinal** (like Jr. kg, Sr. kg, Primary school, high school)
2. The number of categories is quite large as one-hot encoding can lead to high memory consumption

ASSIGNMENT – 04

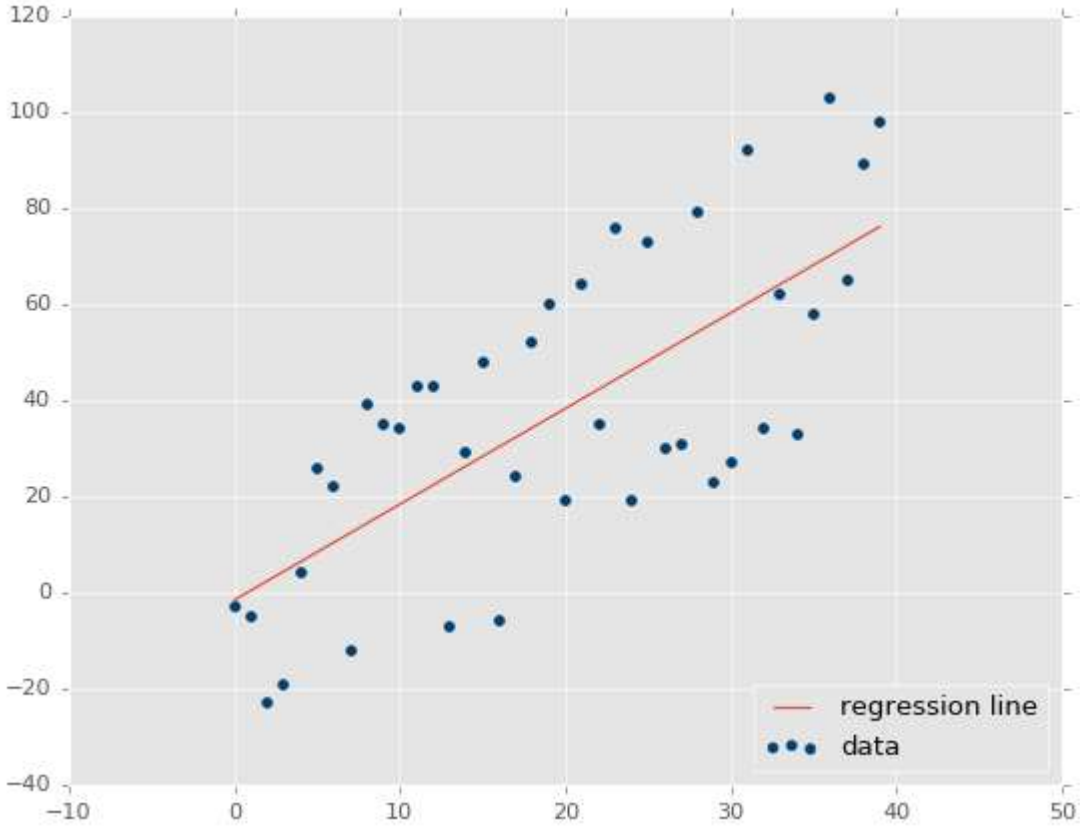
TOPIC: LINEAR REGRESSION

Q.1. Write notes on LinearRegression model of sklearn module.

Q.2. Operate with different 'random_state' values and create a report for the same.

Ans. 1) LINEAR REGRESSION MODEL OF SKLEARN:

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. If we plot the independent variable (x) on the x-axis and dependent variable (y) on the y-axis, linear regression gives us a straight line that best fits the data points, as shown in the figure below.



We know that the equation of a straight line is basically: Y = mx + c

Where 'c' is the intercept and 'm' is the slope of the line. So basically, the linear regression algorithm gives us the most optimal value for the intercept and the slope (in two dimensions). The y and x variables remain the same, since they are the data features and cannot be changed. The values that we can control are the intercept (c) and slope (m). There can be multiple

straight lines depending upon the values of intercept and slope. Basically what the linear regression algorithm does is it fits multiple lines on the data points and returns the line that results in the least error.

This same concept can be extended to cases where there are more than two variables. This is called multiple linear regression. For instance, consider a scenario where you have to predict the price of the house based upon its area, number of bedrooms, the average income of the people in the area, the age of the house, and so on. In this case, the dependent variable (target variable) is dependent upon several independent variables. A regression model involving multiple variables can be represented as:

Y = m1x1 + m2x2 + m3x3 + ...+ mnxn + c

This is the equation of a hyperplane. Remember, a linear regression model in two dimensions is a straight line; in three dimensions it is a plane, and in more than three dimensions, a hyperplane.

Example:

```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
>>> X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
array([[1, 1],
       [1, 2],
       [2, 2],
       [2, 3]])
>>> y = np.dot(X, np.array([1, 2])) + 3
array([ 6,  8,  9, 11])
>>> reg = LinearRegression().fit(X, y)
>>> reg.score(X, y)
1.0
>>> reg.coef_
array([1., 2.])
>>> reg.intercept_
3.0000000000000018
>>> reg.predict(np.array([[3, 5]]))
array([16.])
```

Checking the values of Coefficients and Intercept based on different random_state

In [12]:

```
import random as rd
reg = linear_model.LinearRegression()
i = rd.sample(range(50),10)
x,a,b = [],[],[]
for e in i:
    X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.25,
                                                        random_state = e)
    reg.fit(X_train,Y_train)
    Y_pred = reg.predict(X_test)
    x.append(e)
    a.append(reg.coef_.round(decimals = 2))
    b.append(reg.intercept_.round(decimals = 2))
    rand_check = pd.DataFrame({
        'random_state': np.array(x).flatten(),
        'Coefficients': np.array(a).flatten(),
        'Intercept': np.array(b).flatten(),
    })
rand_check
```

Out[12]:

	random_state	Coefficients	Intercept
0	5	9337.14	27048.91
1	26	9532.38	26073.12
2	20	9621.99	25747.08
3	29	9452.90	26034.89
4	2	9518.15	23866.27
5	31	9554.99	25021.98
6	21	9349.63	24602.12
7	14	9778.56	24345.06
8	25	9307.12	27066.13
9	36	9487.41	25982.16

ASSIGNMENT – 05

TOPIC: MULTIPLE LINEAR REGRESSION

Q.1. Write notes about the assumptions of linear regression.

Q.2. Calculate MAE, MSE, RMSE, R-Square of multiple linear regression model.

ASSUMPTIONS OF LINEAR REGRESSION:

1. **Linearity**: The relationship between X and the mean of Y is linear.
2. **Lack of multicollinearity**: a state of very high inter-correlations or inter-associations among the independent variables.
3. **Homoscedasticity**: The variance of residual is the same for any value of X and if there is a specific pattern, the data is **Heteroscedastic.**
4. **Independence of error (Independence)**: Observations are independent of each other.
5. **Multivariate normality (normality)**: For any fixed value of X, Y is normally distributed.

MAE, MSE, RMSE, R-Square of multiple linear regression:

In [14]:
```
from sklearn.metrics import r2_score, mean_squared_error

print("Mean absolute error: %.2f" % np.mean(np.absolute(pred_y - y_test)))
print("Mean square error: %.2f" %np.mean((pred_y - y_test))**2)
print("R2-score: %.2f" % r2_score(pred_y , y_test) )
print('Variance score: %.2f' % reg.score(X_test, y_test))
print('RMSE: %.2f' %np.sqrt(np.mean((pred_y - y_test) ** 2)))
#print('RMSE: %.2f' %np.sqrt(mean_squared_error(y_test, pred_y)))
```

```
Mean absolute error: 5701.10
Mean square error: 951660.65
R2-score: 0.95
Variance score: 0.95
RMSE: 7516.47
```