MACHINE LEARNING ASSIGNMENT 1

GANESHREDDY ANNAPAREDDY

Person Number : 50442295 — ganeshre@buffalo.edu

SIMPLE LINEAR REGRESSION

In line with the observational data, regression models describe the relationship between the variables. Although logistic and nonlinear regression models employ a curved line, linear regression methods just don't. You may estimate a dependent variable's change as an independent variable or set of independent variables changes using regression. To determine the association between two quantitative variables, Simple Linear Regression is performed.

Formula :

$$y = \beta_0 + \beta_1 X + \varepsilon$$

Here :

- y is the dependent variable's (y) expected value for any given value of the independent variable (x)
- The intercept, or expected value of y when x is 0, is represented by β_0.
- The regression coefficient, or β_1, indicates how much we anticipate y to change when x rises.
- X is an independent variable.
- ε stands for the estimate error, or the degree of variance in our estimation of the regression coefficient.

Coding :

I have taken the Linear regression data set from the Kaggle an open source platform . The link https://www.kaggle.com/datasets/andonians/random-linear-regression . This is a CSV file containing 700 data pairs makes up the training dataset (x,y). The x-values are integers in the range of 0 to 100. The Excel function NORMINV(RAND(), x, 3), has been used to get the equivalent y-values. Therefore, x should be the best estimate y. A CSV file containing 300 data pairs makes up the test dataset.

STEP 1: Loading the data as df.

```
df
[8]
```

	X	Y
0	1	3.888889
1	2	4.555556
2	3	5.222222
3	4	5.888889
4	5	6.555556
...	...	...
295	296	200.555556
296	297	201.222222
297	298	201.888889
298	299	1.888889
299	300	1.888889

300 rows × 2 columns

*Here we can see the x, y columns with 300 rows

STEP 2: I am training and testing the x & y values .

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size = 1/4, random_state = 0)
[18]
```

STEP 3: I am importing the linear regression from "sklearn.linear_model" and fitting the model to training set.

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor = regressor.fit(X_train, Y_train)
```
[20]

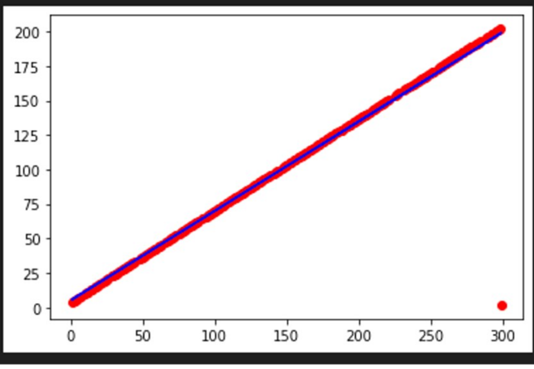
After that I am going to predict the results.

```
Y_pred = regressor.predict(X_test)
```
[21]

STEP 4: I am going to visualize the data using the scatter plot for both (x_train, y_train) and (x_test, y_test)

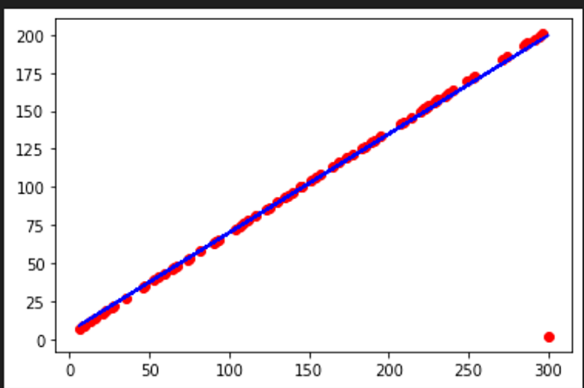
```
plt.scatter(X_train , Y_train, color = 'red')
plt.plot(X_train , regressor.predict(X_train), color ='blue')
```
[22]

```
[<matplotlib.lines.Line2D at 0x1ce9e3eb0a0>]
```




```
plt.scatter(X_test , Y_test, color = 'red')
plt.plot(X_test , regressor.predict(X_test), color ='blue')
```
[23]

```
[<matplotlib.lines.Line2D at 0x1cea04d4310>]
```



MULTIPLE LINEAR REGRESSION

Multiple linear regression, usually referred to as multiple regression which is a statistical method for predicting the result of a response variable using a number of explanatory factors. Modelling the linear connection between the explanatory independent factors and response dependent variables is the aim of multiple linear regression. Because it takes into account several explanatory variables, multiple regression is essentially an extension of conventional least-squares regression.

Formula :

$$y_i = \beta_0 x_0 + \beta_1 x_1 + \ldots\ldots\ldots + \beta_x x_x$$

Here :

- i= number of observations
- y_i is dependent variable
- β_0 is y intercept
- X is an explanatory variable
- β_x is slope coefficient for each explanatory variable

The Multiple linear regression is usually based on few of the assumptions which are

- yi observations are randomly selected and randomly from the population
- The residuals must have a normal distribution with a mean and variance of 0.
- The connection between the dependent and independent variables is linear.
- The independent variables don't have a lot of correlation with one another.

Coding :

I have taken the Linear regression data set from the Kaggle an open source platform . The link https://www.kaggle.com/datasets/farhanmd29/50-startups . This dataset contains information about 50 business startups, including 17 in each of the three states of New York, California, and Florida. Profit, R&D spending, administration spending, and marketing spending are the variables considered in the dataset.

STEP 1: Loading the data as df.

```
In [25]: df
Out[25]:
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96
10	101913.08	110594.11	229160.95	Florida	146121.95

STEP 2: I am importing one hot encoding and then I am training and testing the x & y values .

```
In [27]: from sklearn.preprocessing import OneHotEncoder
```

```
In [28]: from sklearn.preprocessing import LabelEncoder, OneHotEncoder
         labelencoder = LabelEncoder()
         X[: , 3] = labelencoder.fit_transform(X[ : , 3])
         ##onehotencoder = OneHotEncoder(categorical_features = [3])
         ##X = onehotencoder.fit_transform(X).toarray()
```

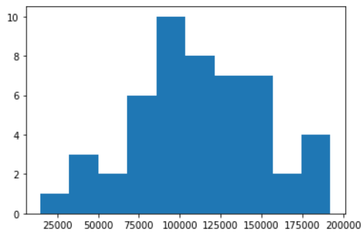
```
In [31]: from sklearn.model_selection import train_test_split
         X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
```

STEP 3: I am importing the linear regression from "sklearn.linear_model" and fitting the model to training set.

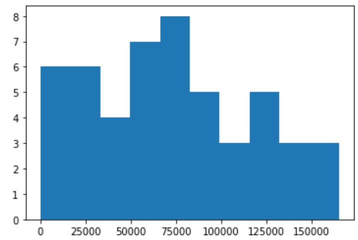
```
In [36]: from sklearn.linear_model import LinearRegression
         regressor = LinearRegression()
         regressor.fit(X_train, Y_train)

Out[36]: ▾ LinearRegression
         LinearRegression()
```

STEP 4: I am going to get the box plot and individual plots for R&D Spend, Administration, Marketing Spend and Profit.

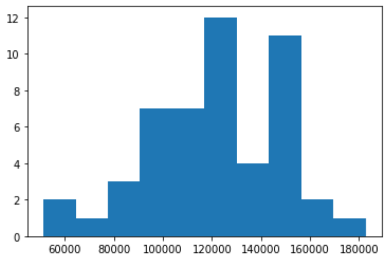
```
In [53]: plt.hist(data=df,x='Profit')
         plt.show()
```



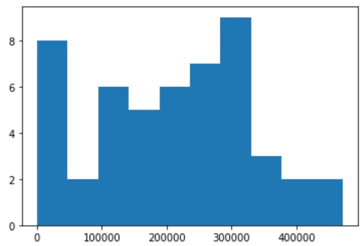

```
In [41]: plt.hist(data=df,x='R&D Spend')
         plt.show()
```



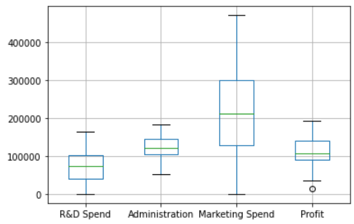

```
In [42]: plt.hist(data=df,x='Administration')
         plt.show()
```




```
In [43]: plt.hist(data=df,x='Marketing Spend')
         plt.show()
```




```
In [39]: df.boxplot()
Out[39]: <AxesSubplot: >
```



STEP 5: Getting the mean absolute error, mean squared error and r square value of the data set.

```
In [55]: Rsquare=r2_score(Y_test,y_pred)
         print('The R-Square value of model is:', Rsquare)

         The R-Square value of model is: 0.3161625677198351
```

```
In [56]: MAE=mean_absolute_error(Y_test,y_pred)
         print('The MAE value of model is:', MAE)

         The MAE value of model is: 24725.681223294312
```

```
In [57]: MSE=mean_squared_error(Y_test,y_pred)
         print('The MSE value of model is:', MSE)

         The MSE value of model is: 874553943.123917
```

LOGISTIC REGRESSION

The probability of a target variable is predicted using the supervised learning classification technique known as logistic regression. Since the dependent variable's nature is dichotomous, there are only two viable classes. Simply said, the dependent variable is a binary variable, with data recorded as either 1 (which represents success/yes) or 0 (which represents failure/no). A logistic regression model makes mathematical predictions about P(Y=1) as a function of X. One of the most basic machine learning algorithms, it may be used to a number of categorization issues, including spam identification, diabetes prediction, cancer diagnosis, etc.

Logistic Regression Assumptions:

- When using binary logistic regression, the result of interest is represented by factor level 1 and the target variables are required to always be binary.
- The independent variables in the model must be independent of one another in order to prevent multi-collinearity.
- Our model must contain relevant variables.
- For logistic regression, a high sample size is recommended.

Coding :

I have taken the Linear regression data set from the Kaggle an open source platform . The link https://www.kaggle.com/datasets/heptapod/titanic. This is a Titanic's Merged train and test data. Removed the 'ticket' and 'cabin' attributes. Moved the 'Survived' attribute to the last column . Added extra zero columns for categorical inputs to be better suited for One-Hot-Encoding. Substituted the values of 'Sex' and 'Embarked' attributes with binary and categorical values respectively. Filled the missing values in 'Age' and 'Fare' attributes with the median of the data.

STEP 1: Loading the data as df.

```
In [53]: df = pd.read_csv('C:/Users/annap/anaconda3/libs/titanic.csv')
         X = df.iloc[:, [2, 3]].values
         y = df.iloc[:, 4].values

In [54]: df
Out[54]:
```

	Passengerid	Age	Fare	Sex	sibsp	zero	zero.1	zero.2	zero.3	zero.4	...	zero.12	zero.13	zero.14	Pclass	zero.15	zero.16	Embarked	zero.17
0	1	22.0	7.2500	0	1	0	0	0	0	0	...	0	0	0	3	0	0	2.0	0
1	2	38.0	71.2833	1	1	0	0	0	0	0	...	0	0	0	1	0	0	0.0	0
2	3	26.0	7.9250	1	0	0	0	0	0	0	...	0	0	0	3	0	0	2.0	0
3	4	35.0	53.1000	1	1	0	0	0	0	0	...	0	0	0	1	0	0	2.0	0
4	5	35.0	8.0500	0	0	0	0	0	0	0	...	0	0	0	3	0	0	2.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1304	1305	28.0	8.0500	0	0	0	0	0	0	0	...	0	0	0	3	0	0	2.0	0
1305	1306	39.0	108.9000	1	0	0	0	0	0	0	...	0	0	0	1	0	0	0.0	0
1306	1307	38.5	7.2500	0	0	0	0	0	0	0	...	0	0	0	3	0	0	2.0	0
1307	1308	28.0	8.0500	0	0	0	0	0	0	0	...	0	0	0	3	0	0	2.0	0
1308	1309	28.0	22.3583	0	1	0	0	0	0	0	...	0	0	0	3	0	0	0.0	0

1309 rows × 28 columns

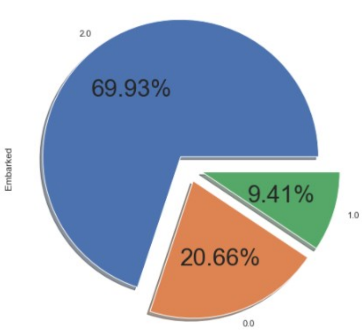
STEP 2: I am training and testing the x & y values .

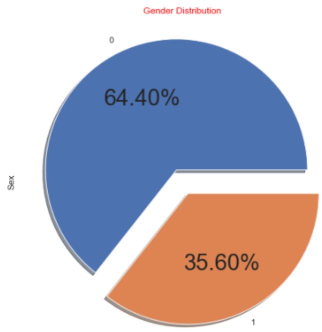
```
In [55]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

STEP 3: I am importing the logistic regression from "sklearn.linear_model" and fitting the model to training set.

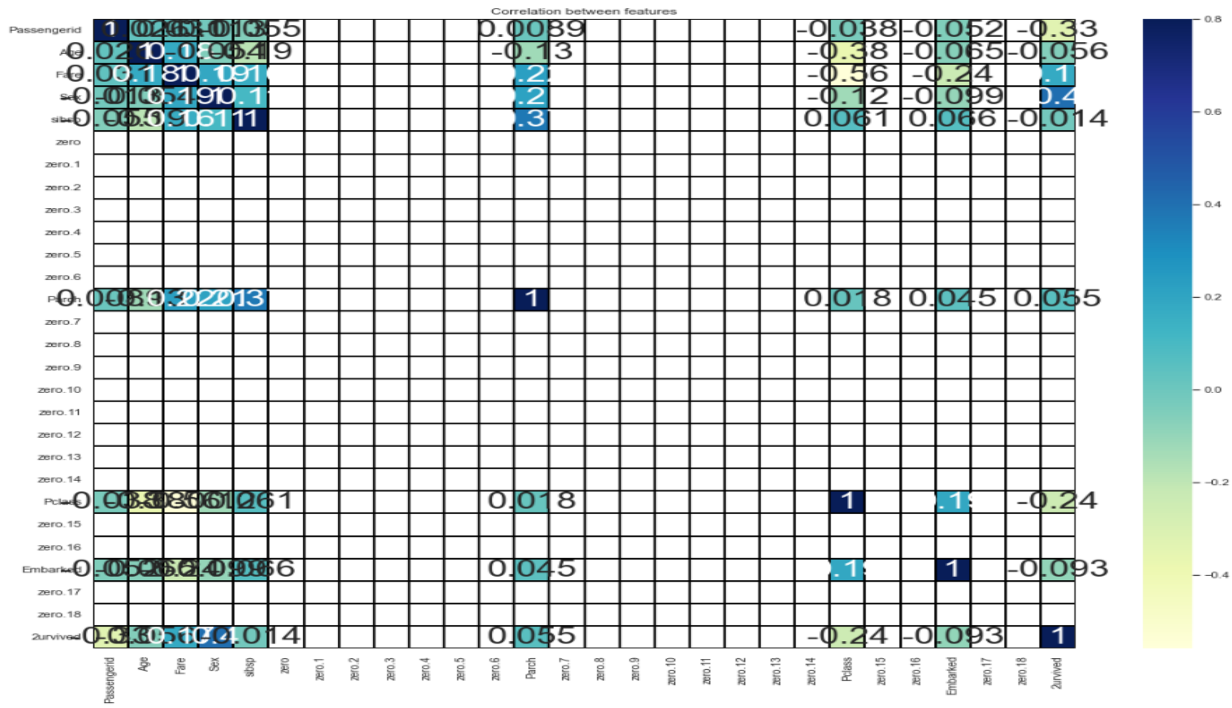
```
In [57]: from sklearn.linear_model import LogisticRegression
         classifier = LogisticRegression()
         classifier.fit(X_train, y_train)

Out[57]: ▾ LogisticRegression
         LogisticRegression()
```

STEP 4: I am going to get the pie plot for the people who embarked the journey and their gender report.





STEP 4: I am going to get the chart for correlation between the features.



STEP 4: I am going to get the accuracy for the logical regression model.

```
In [78]: from sklearn.metrics import accuracy_score
         from sklearn.linear_model import LogisticRegression
         logreg = LogisticRegression()
         logreg.fit(X_train, y_train)
         Y_pred = logreg.predict(X_test)

         log_train = round(logreg.score(X_train, y_train) * 100, 2)
         log_accuracy = round(accuracy_score(Y_pred, y_test) * 100, 2)

         print("Training Accuracy     :",log_train)
         print("Model Accuracy Score :",log_accuracy)

         Training Accuracy     : 67.69
         Model Accuracy Score : 65.24
```

STEP 4: I am going to get the confusion matrix.

```
In [60]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test, y_pred)

In [61]: cm

Out[61]: array([[208,  10,   0,   0,   0,   0,   0],
                [ 82,   6,   0,   0,   0,   0,   0],
                [  9,   1,   0,   0,   0,   0,   0],
                [  3,   1,   0,   0,   0,   0,   0],
                [  3,   0,   0,   0,   0,   0,   0],
                [  1,   0,   0,   0,   0,   0,   0],
                [  4,   0,   0,   0,   0,   0,   0]], dtype=int64)
```