JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Analysis and Implementation of Linear and Multiple Regression for Data Visualization

B.G.Prasanthi [1]
Department of Computer ScienceSt Joseph's University Bangalore, India nitai2009@gmail.com

Sara Kutty T K [2]
Department of Computer ScienceSt Joseph's University Bangalore, India sarajobythomas@yahoo.co.in

***Abstract*—Artificial Intelligence is broader term consisting of many Machine Learning models and Deep Learning neural networks as its components for the machine to interact and behave just like humans. Machine Learning, Artificial Intelligence, and Deep Learning concepts, principles, types and the application is discussed in this paper. The purpose of this topic is to help others easily understand the various terms, methods/ techniques, and technologies people can use for machine learning purposes. Few of the machine learning models such as simple linear regression, multiple linear regression are researched and done in this paper. In this paper, the various machine learning algorithms, methods, and tools have been tested and tried in Visual Studio Code using Python. The outputs for all the various codes have also been given besides the code.**

Keywords—Artificial Intelligence, Machine Learning, Deep Learning, Simple Linear Regression, Multiple Linear Regression

I. Introduction

Artificial Intelligence (AI) is a combination of two words, "Artificial" meaning something man-made or non-natural things, and then the word "Intelligence" which refers to one's ability to think and decide what should be done in the given circumstances. Artificial Intelligence can generally be classified into three levels or categories: Artificial Narrow Intelligence, Artificial General Intelligence and Artificial Super Intelligence

Artificial Narrow Intelligence (ANI), generally known as weak AI or Narrow AI, is the sort of AI that we have so far been able to properly develop and implement. It is a form of AI that is trained to undertake and behave for specified tasks. Weak AI pertains to machines that specialize in a particular task and operates under a confined set of constraints and limitations [1].

Artificial General Intelligence (AGI) is the form of AI that can accomplish work effectively to accomplish goals in about the same capacity that a human can. There are currently no AI that properly qualify as a General AI. These are still preliminary research and experimentation. Some predict that General AI will really be achievable by 2040, whereas others doubt that AGI could ever be realized.

Artificial Super Intelligence (ASI) is a hypothetical AI, that has surpassed the intelligence of humans and can execute tasks better than us. Such systems are evolved forms of General AI.

II. Machine Learning

Machine learning is a component of Artificial Intelligence and Computer Science that focuses on the use of data and algorithms to emulate the way in which people learn, with both the goal and method/process of steadily improving accuracy [2]. Machine learning was described by Arthur Lee Samuel, an American pioneer in the fields of computer programming, machine learning, and artificial intelligence, in 1959 as a "Field of study that grants computers the ability to learn without being explicitly programmed."

A. *Working of Machine Learning Algorithms*

Machine learning algorithms create a model based on training data, which is used to train the algorithm how to make accurate predictions or operate in certain circumstances. All of this can be taught automatically by the algorithm without it being specifically coded to do so. This algorithm's training phase will be repeated until the algorithm can successfully complete all of its objectives according to its constraints [3]. The working of machine learning algorithm is represented in "Fig 1". Data with similar traits are grouped by the algorithm; this grouping is called clusters. These prove helpful in the study of these groups, which can be applied to the entire data within a cluster more or less. Once the algorithm analyses and comes up with the probability distribution of the input, it can be used to generate new data. This proves to be very helpful in cases of missing data.

III. Data Preprocessing for Machine Learning

Data are pre-processed to a particular format before the machine learning takes place so that the machine learning can be performed in the right way this is done because there may be times when in the dataset there are missing data and to tackle this there are multiple ways to solve and handle it to prevent any errors when training the machine learning models.

Sometimes the observation can be deleted and ignored when there are only a few missing data in a big dataset, but if there are cases where there are many missing data then maybe the average (mean) or median of the dataset observations are used [4].

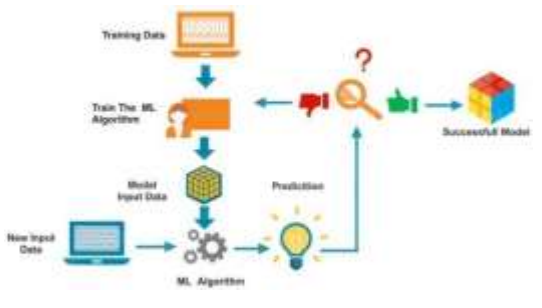
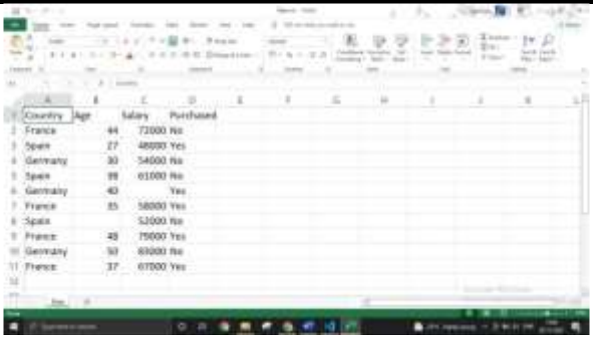


Fig. 1: Working of Machine Learning Algorithm



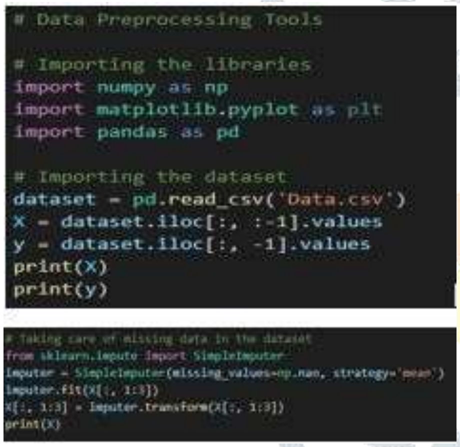
Country	Age	Salary	Purchased
France	44	72000	No
Spain	27	48000	Yes
Germany	30	54000	No
Spain	38	61000	No
Germany	40		Yes
France	35	58000	Yes
Spain		52000	No
France	48	79000	Yes
Germany	50	83000	No
France	37	67000	Yes

Fig 2 : Sample Dataset

In the dataset shown in “Fig 2”, dependent variables are the features which are the columns that are to be predicted based on the independent features. Thus two features are created; X for the independent features such as Country, age, and salary; and Y the dependent variable which has to be predicted. The missing data in the dataset is taken care using the Python code represented in “Fig 3”.

A. *Encoding the Categorical Data*

There are times when data such as the country column has to be encoded into something the computer understands . The country field in the dataset is encoded as shown in “Fig 4”. The onehot encoder was used in the code. The dependent variable was encoded as 0 and 1 as there were only two outputs- yes or no [5].



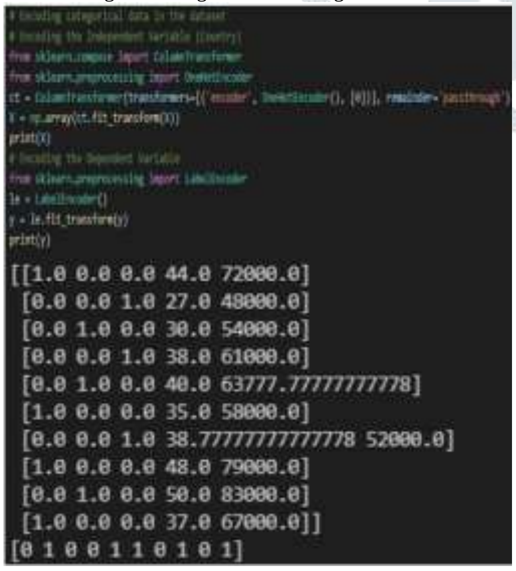
```
# Data Preprocessing Tools

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
print(X)
print(y)
```

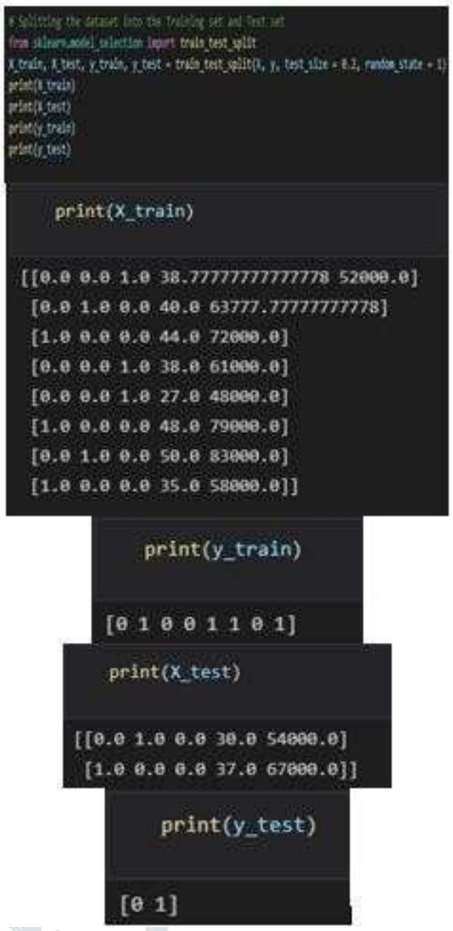
```
# Taking care of missing data in the dataset
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(X[:, 1:3])
X[:, 1:3] = imputer.transform(X[:, 1:3])
print(X)
```

Fig 3: Taking care of Missing Data



```
# Encoding categorical data in the dataset
# Encoding the Independent Variable (Country)
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
print(X)
# Encoding the Dependent Variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
print(y)
[[1.0 0.0 0.0 44.0 72000.0]
 [0.0 0.0 1.0 27.0 48000.0]
 [0.0 1.0 0.0 30.0 54000.0]
 [0.0 0.0 1.0 38.0 61000.0]
 [0.0 1.0 0.0 40.0 63777.77777777778]
 [1.0 0.0 0.0 35.0 58000.0]
 [0.0 0.0 1.0 38.77777777777778 52000.0]
 [1.0 0.0 0.0 48.0 79000.0]
 [0.0 1.0 0.0 50.0 83000.0]
 [1.0 0.0 0.0 37.0 67000.0]]
[0 1 0 0 1 1 0 1 0 1]
```

Fig.4 : Encoding the Categorical Data



```
# Splitting the dataset into the Training set and Test set
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 = 1)
print(X_train)
print(X_test)
print(y_train)
print(y_test)
```

```
print(X_train)

[[0.0 0.0 1.0 38.77777777777778 52000.0]
 [0.0 1.0 0.0 40.0 63777.77777777778]
 [1.0 0.0 0.0 44.0 72000.0]
 [0.0 0.0 1.0 38.0 61000.0]
 [0.0 0.0 1.0 27.0 48000.0]
 [1.0 0.0 0.0 48.0 79000.0]
 [0.0 1.0 0.0 50.0 83000.0]
 [1.0 0.0 0.0 35.0 58000.0]]
```

```
print(y_train)

[0 1 0 0 1 1 0 1]
```

```
print(X_test)

[[0.0 1.0 0.0 30.0 54000.0]
 [1.0 0.0 0.0 37.0 67000.0]]
```

```
print(y_test)

[0 1]
```

Fig 5 : Splitting the data into two sets- training set and test set

B. *Splitting the data into two sets- Training set and Test set*

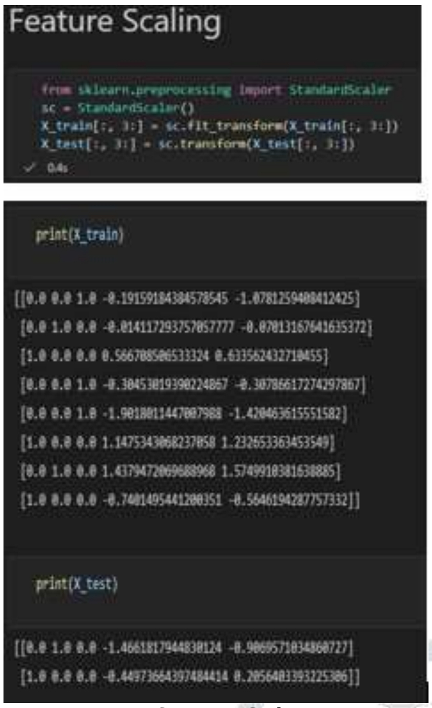
The data set is split into the training set which is to be used to train the machine learning algorithm on the observations available and then to test set which will be used for testing the model developed. Usually, the dataset is split, and then feature scaling is done [6][7]. As usual, it’s better to generate and keep the test set separately so it can be excluded when feature scaling is done. It is widely used that 80% of the observation are to be used for the training set and the remaining 20% for the test set.

C. *Feature Scaling*

The technique of standardizing the range of values of all the independent variables or features of data is known as feature scaling. Data normalization or standardization is another name for this technique. The goal is to reduce the dataset to a narrower, more precise range so that the machine can be trained more efficiently and precisely. There are several methods for normalizing data; two of them are illustrated in table 1. “Fig 6” shows the feature scaling done on the dataset.

TABLE I. METHODS FOR NORMALIZING DATA

Standardisation	Normalisation
Xstand= x-mean(x) / Standard deviation (x)	Xnorm= x-min(x) / (max(x) – min(x))
This will result in all the values between -3 and +3	While in this the result of all the values will be between 0 and 1.



Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:, 3:] = sc.fit_transform(X_train[:, 3:])
X_test[:, 3:] = sc.transform(X_test[:, 3:])
✓ 0.4s
```

```
print(X_train)
```

```
[[0.0 0.0 1.0 -0.19159184384578545 -1.0781259408412425]
 [0.0 1.0 0.0 -0.014117293757057777 -0.07013167641635372]
 [1.0 0.0 0.0 0.566708506533324 0.633562432710455]
 [0.0 0.0 1.0 -0.30453019390224867 -0.30786617274297867]
 [0.0 0.0 1.0 -1.9018011447007988 -1.420463615551582]
 [1.0 0.0 0.0 1.1475343068237058 1.232653363453549]
 [0.0 1.0 0.0 1.4379472069688968 1.5749910381638885]
 [1.0 0.0 0.0 -0.7401495441200351 -0.5646194287757332]]
```

```
print(X_test)
```

```
[[0.0 1.0 0.0 -1.4661817944830124 -0.9069571034860727]
 [1.0 0.0 0.0 -0.44973664397484414 0.2056403393225306]]
```

Fig 6: Feature Scaling

IV. Linear Regression

A. *Simple Linear Regression*

Simple Linear Regression is a method for estimating real values (house prices, phone calls, total sales, and so forth) based on a continuous variable (s). By fitting the optimal line, a link between independent and dependent variables can be established., The regression line is the best fit line, and it is represented by the linear equation Y= a *X + b. Here X is the Independent variable, Y is the Dependent Variable, a is the Slope and b the Intercept. The a and b coefficients are calculated by minimizing the sum of squared differences in distance between data points and the regression line.

The dataset and libraries are imported similarly as shown earlier and is shown in "Fig 7" and "Fig 8". The LinearRegression() class builds the simple linear regression model, thus the object is created using this class and the parameters are passed.

An example to find out how much would a person of 30 years of experience have as his/ her salary is represented in "Fig 9". The 30 years value was inputted into a double pair of [] brackets, as the predict method always takes a 2D array as its parameter for inputs. To find the simple linear regression model equation, first find the co-efficient and intercept by using the regression object. Hence the equation of the simple linear regression model is

Salary = 9345.942 * Years of Experience + 26816.192

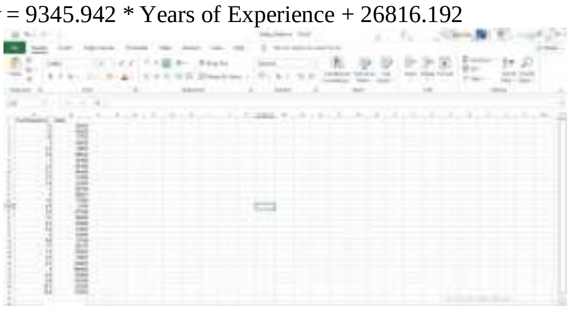
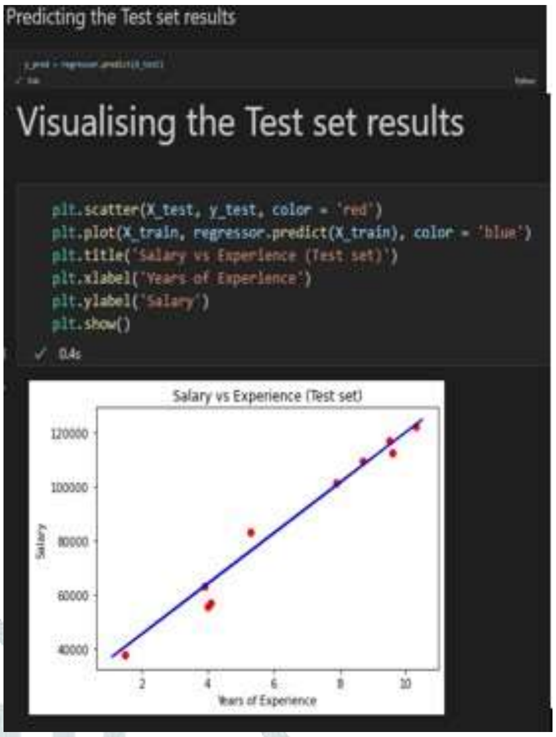


Fig 7 : Sample Data set



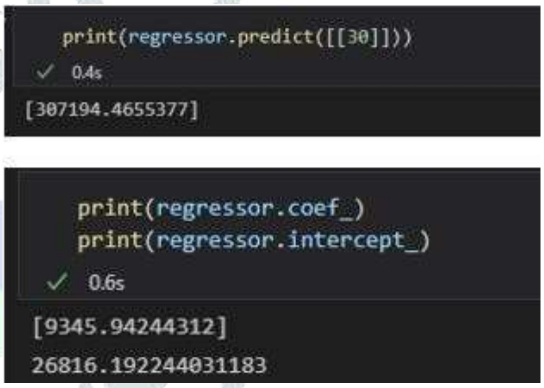
Predicting the Test set results

```
y_pred = regressor.predict(X_test)
```

Visualising the Test set results

```
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
✓ 0.4s
```

Fig 8 : Visualising the test set results



```
print(regressor.predict([[30]]))
✓ 0.4s
[307194.4655377]
```

```
print(regressor.coef_)
print(regressor.intercept_)
✓ 0.6s
[9345.94244312]
26816.192244031183
```

Fig 9: Example Prediction

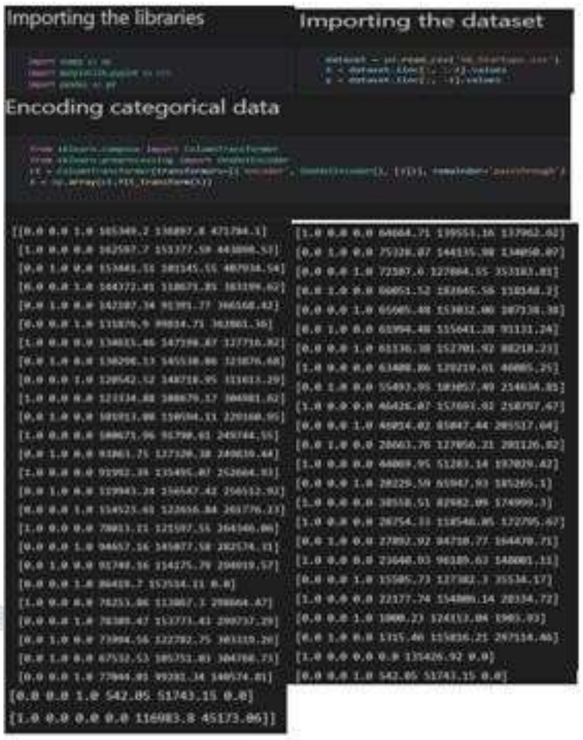
B. *Multiple Linear Regression*

Multiple Linear Regression is identical to the simple linear regression equation, however there are more than one independent variable and just one dependent variable in multiple linear regression. In such a type of regression, we will be able to see how the dependent variable will change for different independent variables. In short multiple linear regression is used to analyze and estimate the relationship between two or more independent variables and one dependent variable. The sample dataset used is shown in table 2. "Fig 10" shows the code written in Python for importing the libraries and the dataset; Encoding the categorical data; splitting the dataset into training set and test set; and then training the multiple linear regression model on the training set [8].

The multiple Regression function automatically detects and handles the multiple independent variables and chooses the best or most statistically significant for the linear regression.

TABLE II. SAMPLE DATASET

R&D Spend	Administration	Marketing Spend	State	Profit
165349.2	136897.8	471784.1	New York	192261.83
162597.7	151377.59	443898.53	California	191792.06
153441.51	101145.55	407934.54	Florida	191050.39
144372.41	118671.85	383199.62	New York	182901.99
142107.34	91391.77	366168.42	Florida	166187.94
131876.9	99814.71	362861.36	New York	156991.12
134615.46	147198.87	127716.82	California	156122.51
130298.13	145530.06	323876.68	Florida	155752.6
120542.52	148718.95	311613.29	New York	152211.77
123334.88	108679.17	304981.62	California	149759.96
101913.08	110594.11	229160.95	Florida	146121.95
100671.96	91790.61	249744.55	California	144259.4
93863.75	127320.38	249839.44	Florida	141585.52
91992.39	135495.07	252664.93	California	134307.35
119943.24	156547.42	256512.92	Florida	132602.65
114523.61	122616.84	261776.23	New York	129917.04
78013.11	121597.55	264346.06	California	126992.93
94657.16	145077.58	282574.31	New York	125370.37
91749.16	114175.79	294919.57	Florida	124266.9
86419.7	153514.11	0	New York	122776.86
76253.86	113867.3	298664.47	California	118474.03
78389.47	153773.43	299737.29	New York	111313.02
73994.56	122782.75	303319.26	Florida	110352.25
67532.53	105751.03	304768.73	Florida	108733.99
77044.01	99281.34	140574.81	New York	108552.04
64664.71	139553.16	137962.62	California	107404.34
75328.87	144135.98	134050.07	Florida	105733.54
72107.6	127864.55	353183.81	New York	105008.31
66051.52	182645.56	118148.2	Florida	103282.38
65605.48	153032.06	107138.38	New York	101004.64
61994.48	115641.28	91131.24	Florida	99937.59
61136.38	152701.92	88218.23	New York	97483.56
63408.86	129219.61	46085.25	California	97427.84
55493.95	103057.49	214634.81	Florida	96778.92
46426.07	157693.92	210797.67	California	96712.8
46014.02	85047.44	205517.64	New York	96479.51
28663.76	127056.21	201126.82	Florida	90708.19
44069.95	51283.14	197029.42	California	89949.14
20229.59	65947.93	185265.1	New York	81229.06
38558.51	82982.09	174999.3	California	81005.76
28754.33	118546.05	172795.67	California	78239.91
27892.92	84710.77	164470.71	Florida	77798.83
23640.93	96189.63	148001.11	California	71498.49
15505.73	127382.3	35534.17	New York	69758.98
22177.74	154806.14	28334.72	California	65200.33
1000.23	124153.04	1903.93	New York	64926.08
1315.46	115816.21	297114.46	Florida	49490.75
0	135426.92	0	California	42559.73
542.05	51743.15	0	New York	35673.41
0	116983.8	45173.06	California	14681.4



Importing the libraries

```
[illegible]
```

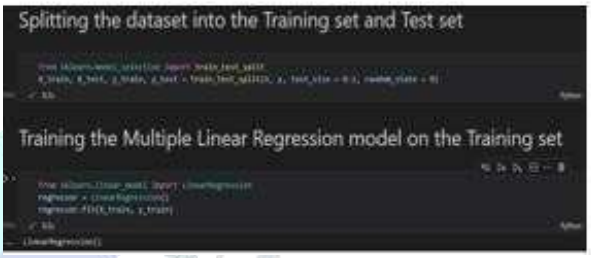
Importing the dataset

```
[illegible]
```

Encoding categorical data

```
[illegible]
```

```
[illegible]
[0.0 0.0 1.0 542.05 51743.15 0.0]
[1.0 0.0 0.0 0.0 116983.8 45173.06]]
```



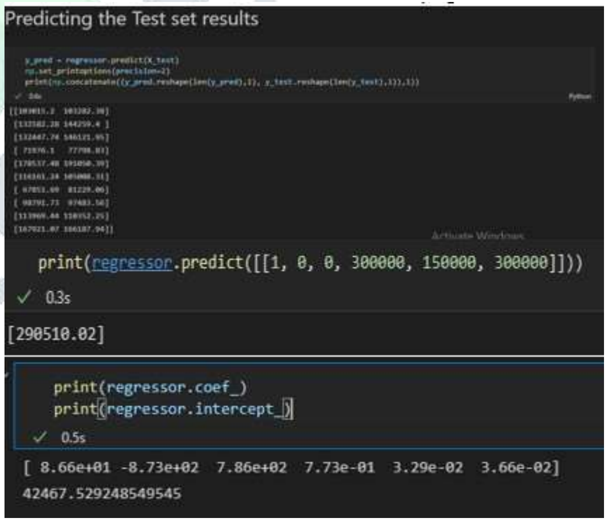
Splitting the dataset into the Training set and Test set

```
[illegible]
```

Training the Multiple Linear Regression model on the Training set

```
[illegible]
```

Fig 10: Training the Multiple Linear regression model



Predicting the Test set results

```
[illegible]
```

```
print(regressor.predict([[1, 0, 0, 300000, 150000, 300000]]))
✓ 0.3s
[290510.02]
```

```
print(regressor.coef_)
print(regressor.intercept_)
✓ 0.5s
[ 8.66e+01 -8.73e+02  7.86e+02  7.73e-01  3.29e-02  3.66e-02]
42467.529248549545
```

Fig 11: Predicting the test set results

Prediction on the test set results is shown in "Fig 11". The np.concatenate() function is used to concatenate two horizontal or vertical vectors. The 1 passed at the end refers to a horizontal concatenation. Here the second column(y_test) is the real profits on the test set, while the first column (y_pred) is the predicted profits. Making a single prediction (for example the profit of a startup with R&D Spend = 300000, Administration Spend = 150000, Marketing Spend = 300000 and State = 'California'). Therefore, the equation of our multiple linear regression model is :

Profit=86.6*Dummy State 1−873*Dummy State 2 +786*Dummy State 3+0.773*R&D Spend +0.0329*Administration+0.0366*Marketing Spend +42467.53

To get these coefficients the "coef_" and "intercept_" attributes were used from regressor object. Attributes in Python are different than methods and usually return a simple value or an array of values. Among the most fundamental and widely used unsupervised machine learning is K-means clustering. A cluster is a collection of data points that have been grouped together due to particular commonalities. You'll specify a goal number, k, for the number of centroids required in the dataset. A centroid is a fictional or genuine position that symbolizes the cluster's center. By lowering the in-cluster sum of squares, each pixel is assigned to one of the clusters. To put it differently, the K-means algorithm finds k centroids and then allocates every data point to the closest centroid while keeping the centroids as small as practicable. The average of the data, or determining the centroid, is just what the 'means' in K-means which can show the different clusters for visualization [9][10].

V. CONCLUSION

The linear and multiple regression model was implemented using a sample data set and found that multiple regression is better than linear regression considering the parameters correlation and use of algorithms has proved with the experimental results taking secondary dataset ESVMarket data .Multiple linear regression is used to estimate the relationship between two or more independent variables instead of one independent variable .This can be implemented using a multiple means clustering method with specified k clusters for visualization of customers using the multiple regression with their income and profit.

REFERENCES

[1] Ferrari M, Quaresima V. A brief review on the history of human functional near-infrared spectroscopy (fNIRS) development and fields of application. Neuroimage. 2012 Nov 1;63(2):921-35. doi: 10.1016/j.neuroimage.2012.03.049. Epub 2012 Mar 28. PMID: 22510258.

[2] Scholkmann F, Kleiser S, Metz AJ, Zimmermann R, Mata Pavia J, Wolf U, Wolf M. A review on continuous wave functional near-infrared spectroscopy and imaging instrumentation and methodology. Neuroimage. 2014 Jan 15;85 Pt 1:6-27. doi: 10.1016/j.neuroimage.2013.05.004. Epub 2013 May 16. PMID: 23684868.

[3] Huppert TJ, Diamond SG, Franceschini MA, Boas DA. HomER: a review of time-series analysis methods for near-infrared spectroscopy of the brain. Appl Opt. 2009 Apr 1;48(10):D280-98. doi: 10.1364/ao.48.00d280. PMID: 19340120; PMCID: PMC2761652.

[4] Koh PH, Glaser DE, Flandin G, Kiebel S, Butterworth B, Maki A, Delpy DT, Elwell CE. Functional optical signal analysis: a software tool for near-infrared spectroscopy data processing incorporating statistical parametric mapping. J Biomed Opt. 2007 Nov-Dec;12(6):064010. doi: 10.1117/1.2804092. PMID: 18163826.

[5] Li H, Tak S, Ye JC. Lipschitz-Killing curvature based expected Euler characteristics for p-value correction in fNIRS. J Neurosci Methods. 2012 Feb 15;204(1):61-67. doi: 10.1016/j.jneumeth.2011.10.016. Epub 2011 Oct 30. PMID: 22074819.

[6] Tomer Fekete, Denis Rubin, Joshua M. Carlson, Lilianne R. Mujica-Parodi, , The NIRS Analysis Package: Noise Reduction and Statistical Inference, September 2, 2011, https://doi.org/10.1371/journal.pone.0024322

[7] Tak S, Ye JC. Statistical analysis of fNIRS data: a comprehensive review. Neuroimage. 2014 Jan 15;85 Pt 1:72-91. doi: 10.1016/j.neuroimage.2013.06.016. Epub 2013 Jun 15. PMID: 23774396.

[8] Olive, David & Watagoda, Lasanthi & Rupasinghe Arachchige Don, Hasthika. (2015). Visualizing and Testing the Multivariate Linear Regression Model. International Journal of Statistics and Probability. 4. 10.5539/ijsp.v4n1p126.

[9] Deepali Patil, Aarti Puthran, MOVIES ON OTT ANALYSIS USING MULTIPLE REGRESSION AND RANDOM FOREST IN R 2021 IJCRT,| Volume 9, Issue 6 June 2021, ISSN: 2320-2882

[10] I. Keka and B. Çiço, "Data Visualization as Helping Technique for Data Analysis, Trend Detection and Correlation of Variables Using R Programming Language," 2019 8th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 2019, pp. 1-4, doi: 10.1109/MECO.2019.8760004.