Dependency Of Accuracy On The Ratio Of Train-Test Split.

Ganesh Sahu[1] , Dr. Ram Chandra Barik[2]

[1] Bachelor In Computer Application, Vikash School Of Business Management, Bargarh, India
[2] Assistant Professor, C. V. Raman Global University (Department of Computer Science and Engineering)

Abstract : Linear Regression is a machine learning algorithm based on supervised learning. It executes a regression operation. Regression uses independent variables to model a goal prediction value. It is mostly used to determine how variables and forecasting relate to one another, and to do so we generally use dependent variables and independent variables. Here to know the accuracy we split out the data set into train and test data, and to perform that we take a standard splitting ratio of 8:2 or 7:3.This paper verifies whether the change in the ratio of the train-test dataset has a relation with the resultant accuracy of the model. To perform this here we gonna take a simple example data set and test it which different ratios multiple times and determine the change in accuracy. We are using simple linear regression in python 3.6 using Jupiter notebook and sklearn.

Keywords: Regression, Simple Linear Regression, least square method, Train-Test split, Sklearn, Machine learning, Matplotlib.

1.1 Introduction

In a developing world where companies like Facebook created data at a rate of more than 500TB per day in 2012[7], Twitter produced data at a rate of 8TB per day in 2011[6], and Google, the largest Internet business, may have worked on data at a rate of 20PB per day in 2008 [8], they have to convert these raw data into meaningful and useable data. Machine learning [1–5] is frequently utilised in a variety of disciplines to resolve challenging issues that are not easily resolved using computational methods. One of the simplest and most used machine learning methods is linear regression. It is a method of performing predictive analysis using a mathematical approach.

Train-test split procedure:-

The train-test split procedure is used to estimate the performance of machine learning algorithms. They are used to make predictions on data and used to train the model.

It is a fast and easy procedure to perform. The results of which allow you to compare the performance of machine learning algorithms for your predictive modelling problem, Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced[9].

1.2 train test spilt evaluation:-

The train-test split is a technique for evaluating the performance of a machine-learning algorithm.

It can be used for classification or regression problems and can be used for any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.[9]

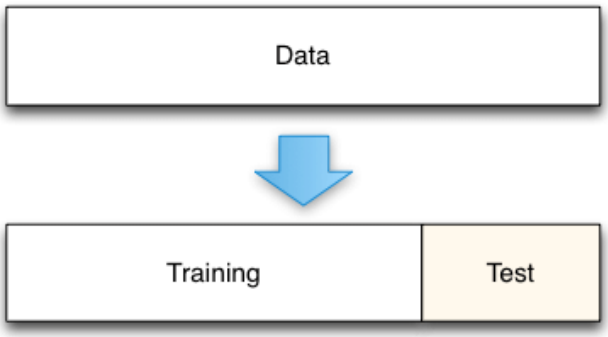

fig:1

Python:-

Python is one of the most popular object-oriented programming languages among developers and also For this experiment, we are going to use python to perform the necessary coding.

The Python library provides base-level items, so developers do not have to write code from scratch every time. Machine learning requires continuous data processing, and Python libraries allow us to access, process and transform our data. These are some of the most extensive libraries available for AI and ML and from which some of them we are going to use. [10].

- **Scikit-learn** to handle basic ML algorithms such as clustering, logistic and linear regression, regression, and classification. [10].
- **Pandas** are used for advanced structure and data analysis. It allows you to merge and filter data and collect data from other external sources (such as Excel). [10].
- **Keras** is used for deep learning. In addition to the computer's CPU, it also uses the GPU, allowing rapid calculations and prototyping. [10].
- **TensorFlow** is used to manipulate deep understanding by building, training, and using artificial neural networks using substantial data sets. [10].

Jupyter notebook :-

Computation notebooks have been used as electronic lab notebooks to document procedures, data, calculations, and findings. Jupyter notebooks provide an interactive computational environment for developing data science applications.[11]

The Jupyter Notebook project supports dozens of programming languages, its name reflecting support for Julia (Ju), Python (Py), and R.[11]

Method :-

Here we observe, how we split our train and test data set using SKlearn model of the library of panda in machine learning. Using the train-test split method Usually, when we have data set like this Sometimes we used to train the model using all data sets. But that's not a good approach. The perfect strategy is to split the data set into 2 parts Where we use part of the sample for actual training And use the remaining sample for testing our model. The reason is, we want to use those samples For testing the model that our Model has not seen before. So for example here if we use the first 8 models of my sample to train the model and then use the remaining set to test, then I get a good idea on the accuracy of the model Cause the model has not seen this sample before.

Mileage	Age(yrs)	Sell Price($)
69000	6	18000
35000	3	34000
57000	5	26100
22500	2	40000
46000	4	31500
59000	5	26750
52000	5	32000
72000	6	19300
91000	8	12000
67000	6	22000

fig:2

	Mileage	Age(yrs)	Sell Price($)
Training (80%)	69000	6	18000
	35000	3	34000
	57000	5	26100
	22500	2	40000
	46000	4	31500
	59000	5	26750
	52000	5	32000
	72000	6	19300
Test	91000	8	12000
	67000	6	22000

fig:3

	A	B	C
1	Mileage	Age(yrs)	Sell Price($)
2	69000	6	18000
3	35000	3	34000
4	57000	5	26100
5	22500	2	40000
6	46000	4	31500
7	59000	5	26750
8	52000	5	32000
9	72000	6	19300
10	91000	8	12000
11	67000	6	22000
12	83000	7	18700
13	79000	7	19500
14	59000	5	26000
15	58780	4	27500
16	82450	7	19400
17	25400	3	35000
18	28000	2	35500
19	69000	5	19700
20	87600	8	12800
21	52000	5	28200
22			
23			
24			

fig:4

The data set which we are using for This example is this BMW Car prices data set. Here we have two variables: the mileage and the age of the car and the price that it was sold for and here mileage and age are our independent variables and the selling price is the dependent variable. In the Jupiter notebook here we have our file into a data frame which looks like this *fig4*.

```
In [1]: import pandas as pd
        df = pd.read_csv("carprices.csv")
        df.head()
```

Out[1]:

	Mileage	Age(yrs)	Sell Price($)
0	69000	6	18000
1	35000	3	34000
2	57000	5	26100
3	22500	2	40000
4	46000	4	31500

fig:6

Then we are using metalplotlib visualization to figure out the relationship between my dependent and independent variable as shown in *fig5*

```
In [2]: import matplotlib.pyplot as plt
        %matplotlib inline
```

fig:6

So here we have a plot of mileage versus the sale price and you can see a clear linear relationship herein *fig7* .we can draw A-line which can go through all these data points.

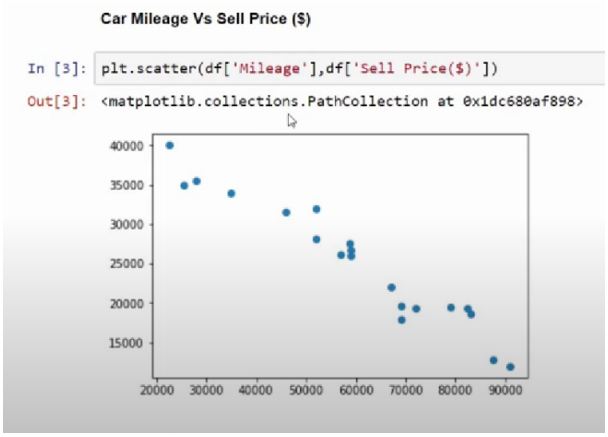

fig:7

Similarly, for car age and shell price, I have plotted another scattered plot and here also we can also apply a linear relationship.

```
In [5]: plt.scatter(df['Age(yrs)'],df['Sell Price($)'])
Out[5]: <matplotlib.collections.PathCollection at 0x1dc67e8e828>
```

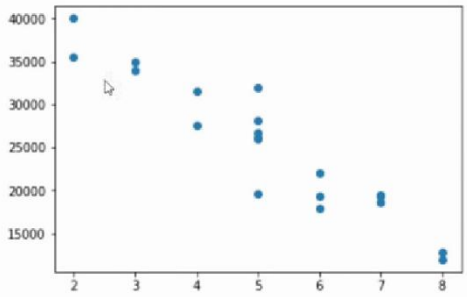

fig:8

So we are going to use the simple linear regression model, based on these visualizations I have prepared my X data and Y data here, So X is the mileage and age where Y is the selling price.

So, now what we do here is we are going to use **train test split** method from **sklearn.model_selection.** we are importing the train test split method and then using this method by supplying X and Y as input.

```
In [72]: from sklearn.model_selection import train_test_split
```

fig:9

Then we need to supply the ratio by which we are splitting. so here I want my test data size to be 20% and my training data size to be 80% .

```
: X_train, X_test, y_train, y_test =  train_test_split(X,y,test_size=0.2)
: len(X_train)
: 16
: len(X_test)
: 4
```

fig:10

As a result what we get is an x_train dataset, x_test dataset, y_train dataset, and y_test data set. So here we get four parameters back and if we look at the length of whatever we got back we see it is 80% of our total data size, our total data size here is 20 and based on 80% ratio My training dataset is 16 samples if we check test-data it would be 4 as it expected to be.

So now let's see the linear regression model and for that, we're going to import linear regression class and create my classifier which is nothing but an object of this class.

```
In [99]: from sklearn.linear_model import LinearRegression
         clf = LinearRegression()
```

fig:11

Then we use **Ffit** method to actually train our model.

```
In [100]: clf.fit(X_train,y_train)
Out[100]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
In [101]: clf.predict(X_test)
Out[101]: array([ 22186.75344538,  23040.49041558,  39712.3812494 ,  39055.40183091])
```

fig:12

So we need to use x_train,y_train and now our model is trained and we can call **predict** method on our actual test data So here our model is predicting a value for our x_test as shown in *fig13* and now if we see our y_test to check the similarity its kind of similar but not 100% accurate.

```
In [102]: y_test
Out[102]: 0     18000
          9     22000
          3     40000
          15    35000
          Name: Sell Price($), dtype: i
```

fig:13

So now we can see the accuracy of our model for that we are going to use the **score** method, what the score method does is it will use x_test and predict these values and compare it against the y_test values and calculate accuracy.

```
In [103]: clf.score(X_test,y_test)
Out[103]: 0.8924543740684386 4
```

fig:14

So, here the accuracy is 89% and this is just because of the nature of our data set.
So, as we get to see the accuracy of the predicted dataset now let's clear some doubts and check the relationship between the change in accuracy and to change in the train_test split ratio.

For this, we are going to perform many observations and we are going to divide them into two types
: without random_state
: with random_state

Here first we take different pairs of train-test ratios and going to perform five iterations with each ratio to note the difference in accuracy and whether it makes any order or relationship.

If we check the actual content of our x_train we can see that it's choosing the random samples it's not selecting the 1st 80% of the sample, which is perfect in some conditions but if we execute this method again and again you see the sample will change rapidly Now sometimes we don't want the sample to remain same and for that, you can use this random state method if we use random state method it's gonna use same samples each time we perform. So random state value of train data will always produce the same amount, as you can see that our x_train is not changing. Here we are going to perform our iteration in both state and observe the resultant.

Without random state:-

Operation 1:-

Firstly, here we are taking the test size to be 50% of the train data set and performing five operations and the results are the following:-

Iteration 1st:-

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.5)
from sklearn.linear_model import LinearRegression
clf = LinearRegression()
clf.fit(X_train, y_train)
clf.predict(X_test)
clf.score(X_test, y_test)
```

```
0.8592647778873638
```

Iteration 2nd:-

```
0.9311235029839575
```

Iteration 3rd:-

```
0.8685244175836442
```

Iteration 4th:-

```
0.9213246147390187
```

Iteration 5th :-

```
0.8815252687679208
```

Operation 2:-

Taking the test size to be 40% of the train data set and performing five operations and the results are the following:-

Iteration 1st:-

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.4)
from sklearn.linear_model import LinearRegression
clf = LinearRegression()
clf.fit(X_train, y_train)
clf.predict(X_test)
clf.score(X_test, y_test)
```

```
0.8401179684032325
```

Iteration 2nd :-

```
0.8350480562152451
```

Iteration 3rd :-

```
0.8481412097338992
```

Iteration 4th :-

```
0.8664397432320918
```

Iteration 5th:-

```
0.8268626252391797
```

Operation 3:-

Taking the test size to be 30% of the train data set and performing five operations and the results are the following:-

Iteration 1st:-

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
from sklearn.linear_model import LinearRegression
clf = LinearRegression()
clf.fit(X_train, y_train)
clf.predict(X_test)
clf.score(X_test, y_test)
```

```
0.8401358460687627
```

Iteration 2nd:

```
0.892870595800107
```

Iteration 3rd:-

```
0.9211815593357848
```

Iteration 4th:-

```
0.8176642983941782
```

Iteration 5th :-

```
0.9434791591750434
```

Operation 4:-

Taking the test size to be 20% of the train data set and performing five operations and the results are the following:-

Iteration 1st:-

```
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)
from sklearn.linear_model import LinearRegression
clf = LinearRegression()
clf.fit(X_train, y_train)
clf.predict(X_test)
clf.score(X_test, y_test)
```

```
0.8702555908196198
```

Iteration 2st:-

```
Out[92]: 0.5422311668922963
```

Iteration 3rd :-

```
Out[90]: 0.9058022591848761
```

Iteration 4th:-

```
0.8746525159680257
```

Iteration 5th:-

```
Out[93]: 0.927570017426757
```

Operation 5:-

Taking the test size to be 10% of the train data set and performing five operations and the results are the following:-

Iteration 1st :-

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.1)
from sklearn.linear_model import LinearRegression
clf = LinearRegression()
clf.fit(X_train, y_train)
clf.predict(X_test)
clf.score(X_test, y_test)
```

```
0.517534997535158
```

Iteration 2nd :-

```
Out[97]: 0.9835566254462367
```

Iteration 3rd :-

```
Out[96]: -22.834268518814124
```

Iteration 4th:-

```
Out[95]: 0.4881780522863778
```

Iteration 5th:-

```
Out[98]: 0.9807286528363944
```

Now, if we analyse all 5 operations then we get to know that there isn't much change in accuracy or we can say the average accuracy isn't showing any order or major change in respect to the change in the ratio of train data.

With random state:-

Operation 1 :-

When test_data size is 10 percent of the total data set :-

```
In [130]: from sklearn.model_selection import train_test_split
          X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.1,random_state=10)
          from sklearn.linear_model import LinearRegression
          clf = LinearRegression()
          clf.fit(X_train, y_train)
          clf.predict(X_test)
          clf.score(X_test, y_test)

Out[130]: -35.219534901001964
```

Operation 2 :-

When test_data size is 50 percent of the total data set :-

```
In [126]: from sklearn.model_selection import train_test_split
          X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.5,random_state=10)
          from sklearn.linear_model import LinearRegression
          clf = LinearRegression()
          clf.fit(X_train, y_train)
          clf.predict(X_test)
          clf.score(X_test, y_test)

Out[126]: 0.9092482551329326
```

Operation 3 :-

When test_data size is 40 percent of the total data set :-

```
In [120]: from sklearn.model_selection import train_test_split
          X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.4,random_state=10)
          from sklearn.linear_model import LinearRegression
          clf = LinearRegression()
          clf.fit(X_train, y_train)
          clf.predict(X_test)
          clf.score(X_test, y_test)

Out[120]: 0.9049019666314766
```

Operation 4 :-

When test_data size is 30 percent of the total data set :-

```
In [116]: from sklearn.model_selection import train_test_split
          X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=10)
          from sklearn.linear_model import LinearRegression
          clf = LinearRegression()
          clf.fit(X_train, y_train)
          clf.predict(X_test)
          clf.score(X_test, y_test)

Out[116]: 0.921242248377633
```

Operation 5 :-

When test_data size is 20 percent of the total data set :-

```
In [111]: 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=10)
          from sklearn.linear_model import LinearRegression
          clf = LinearRegression()
          clf.fit(X_train, y_train)
          clf.predict(X_test)
          clf.score(X_test, y_test)

Out[111]: 0.733233959309014
```

This is clearly showing that 20 and 10 percent is not showing the accurate result as it should be, but that's doesn't prove that its always going to show the same result its dependent to the data_set too.
Here in these 5 operations also we can't find any type of increasing or decreasing order in accuracy .

Conclusion:-

In this paper, we have tried to answer the widely ask question in data mining and in machine learning So, we have examined whether the accuracy of a model depends on the train_test split ratio .its may depend on large data sets that use in industries, but we can clearlly she that it is not for the small data sets like dataset, what students uses for academic purpose. It's not showing any dependency on the split but what is being more responsible is the dataset we are using.

References :-

1. S. Shalev-Shwartz and S. Ben-David, Understanding machine learning: From theory to algorithms: Cambridge university press, 2014.
2. K. P. Murphy, Machine learning: a probabilistic perspective: MIT press, 2012.
3. P. Domingos, "A few useful things to know about machine learning," Communications of the ACM, vol. 55, pp. 78-87, 2012.
4. D. Q. Zeebaree, H. Haron, A. M. Abdulazeez, and D. A. Zebari, "Machine learning and Region Growing for Breast Cancer Segmentation," in 2019 International Conference on Advanced Science and Engineering (ICOASE), 2019, pp. 88-93.
5. Bargarai, F., Abdulazeez, A., Tiryaki, V., & Zeebaree, D. (2020). Management of Wireless Communication Systems Using Artificial Intelligence-Based Software Defined Radio.
6. Krikorian R. Twitter by the numbers. 2010. http://www.slideshare.net/raffikrikorian/twitter-bythe-numbers.
7. Tam D. Facebook processes more than 500 TB of data daily.2012. http://www.cnet.com/news/facebook-processesmore-than-500-tb-of-data-daily/
8. Wikipedia,Petabyte.2014. http://en.wikipedia.org/wiki/Petabyte
9. *Jason Brownlee* PhD.,Train-Test Split for Evaluating Machine Learning Algorithms, July 24, 2020. https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/
10. Sujith Kumar ,Why Python is Best for AI, ML, and Deep Learning , June 23, 2021 https://www.rtinsights.com/why-python-is-best-for-ai-ml-and-deep-learning/#:~:text=The%20Python%20library%20provides%20base,available%20for%20AI%20and%20ML.
11. Ishu verma red head developer: Introductionto machine learning with Jupyter notebooks, May 21, 2021