ML | Multiple Linear Regression (Backward Elimination Technique)

Multiple Linear Regression is a type of regression where the model depends on several independent variables(instead of only on one independent variable as seen in the case of Simple Linear Regression). Multiple Linear Regression has several techniques to build an effective model namely:

- All-in
- Backward Elimination
- Forward Selection
- Bidirectional Elimination

infofeedacc

In this article, we will implement multiple linear regression using the backward elimination technique.
Backward Elimination consists of the following steps:

- Select a significance level to stay in the model (eg. SL = 0.05)
- Fit the model with all possible predictors
- Consider the predictor with the highest P-value. If P>SL, go to point d.
- Remove the predictor
- Fit the model without this variable and repeat the step c until the condition becomes false.

Let us suppose that we have a dataset containing a set of expenditure information for different companies. We would like to know the profit made by each company to determine which company can give the best results if collaborated with them. We build the regression model using a step by step approach.

Step 1 : Basic preprocessing and encoding

```
# import the necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

# import the dataset
df = pd.read_csv('50_Startups.csv')

# first five entries of the dataset
df.head()

# split the dataframe into dependent and independent variables.
x = df[['R&D Spend', 'Administration', 'Marketing Spend', 'State']]
y = df['Profit']
x.head()
y.head()

# since the state is a string datatype column we need to encode it.
x = pd.get_dummies(x)
x.head()
```

Dataset

	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

The set of independent variables after encoding the state column

	R&D Spend	Administration	Marketing Spend	State_California	State_Florida	State_New York
0	165349.20	136897.80	471784.10	0	0	1
1	162597.70	151377.59	443898.53	1	0	0
2	153441.51	101145.55	407934.54	0	1	0
3	144372.41	118671.85	383199.62	0	0	1
4	142107.34	91391.77	366168.42	0	1	0

Step 2 : Splitting the data into training and testing set and making predictions

```
x_train, x_test, y_train, y_test = train_test_split(
        x, y, test_size = 0.3, random_state = 0)
```

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(x_train, y_train)
pred = lm.predict(x_test)
```

```
pred.reshape(15,1)#to visulaize the data better
```

```
array([[104282.76472171],
       [132536.88499212],
       [133910.85007767],
       [ 72584.77489417],
       [179920.9276189 ],
       [114549.31079233],
       [ 66444.43261347],
       [ 98404.96840122],
       [114499.82808602],
       [169367.50639895],
       [ 96522.6253998 ],
       [ 88040.6718287 ],
       [110949.99405525],
       [ 90419.1897851 ],
       [128020.46250064]])
```

```
y_test
```

```
28    103282.38
11    144259.40
10    146121.95
41     77798.83
2     191050.39
27    105008.31
38     81229.06
31     97483.56
22    110352.25
4     166187.94
33     96778.92
35     96479.51
26    105733.54
34     96712.80
18    124266.90
Name: Profit, dtype: float64
```

We can see that our predictions our close enough to the test set but how do we find the most important factor contributing to the profit.

Here is a solution for that.

We know that the equation of a multiple linear regression line is given by y=b1+b2*x+b3*x'+b4*x"+……. where b1, b2, b3, … are the coefficients and x, x', x" are all independent variables.

Since we don't have any 'x' for the first coefficient we assume it can be written as a product of b and 1 and

hence we append a column of ones. There are libraries that take care of it but since we are using the stats model library we need to explicitly add the column.

Step 3 : Using the backward elimination technique

```
import statsmodels.regression.linear_model as sm
# add a column of ones as integer data type
x = np.append(arr = np.ones((50, 1)).astype(int),
              values = x, axis = 1)
# choose a Significance level usually 0.05, if p>0.05
#  for the highest values parameter, remove that value
x_opt = x[:, [0, 1, 2, 3, 4, 5]]
ols = sm.OLS(endog = y, exog = x_opt).fit()
ols.summary()
```

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.008e+04	6952.587	7.204	0.000	3.61e+04	6.41e+04
x1	0.8060	0.046	17.369	0.000	0.712	0.900
x2	-0.0270	0.052	-0.517	0.608	-0.132	0.078
x3	0.0270	0.017	1.574	0.123	-0.008	0.062
x4	41.8870	3256.039	0.013	0.990	-6520.229	6604.003
x5	240.6758	3338.857	0.072	0.943	-6488.349	6969.701

highest value

This figure shows the highest valued parameter

And now we follow the steps of the **backward elimination and start eliminating unnecessary parameters.**

```
# remove the 4th column as it has the highest value
x_opt = x[:, [0, 1, 2, 3, 5]]
ols = sm.OLS(endog = y, exog = x_opt).fit()
ols.summary()

# remove the 5th column as it has the highest value
x_opt = x[:, [0, 1, 2, 3]]
ols = sm.OLS(endog = y, exog = x_opt).fit()
ols.summary()

# remove the 3rd column as it has the highest value
x_opt = x[:, [0, 1, 2]]
ols = sm.OLS(endog = y, exog = x_opt).fit()
ols.summary()

# remove the 2nd column as it has the highest value
x_opt = x[:, [0, 1]]
ols = sm.OLS(endog = y, exog = x_opt).fit()
ols.summary()
```

Summary after removing the first unnecessary parameter.

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.011e+04	6647.870	7.537	0.000	3.67e+04	6.35e+04
x1	0.8060	0.046	17.606	0.000	0.714	0.898
x2	-0.0270	0.052	-0.523	0.604	-0.131	0.077
x3	0.0270	0.017	1.592	0.118	-0.007	0.061
x4	220.1585	2900.536	0.076	0.940	-5621.821	6062.138

So if we continue the process we see that we are left with only one column at the end and that is the R&D spent.We can conclude that the company which has maximum expenditure on the R&D makes the highest profit.

	0	1
0	1.0	165349.20
1	1.0	162597.70
2	1.0	153441.51
3	1.0	144372.41
4	1.0	142107.34

With this, we have solved the problem statement of finding the company for collaboration. Now let us have a brief look at the parameters of the OLS summary.

- **R square** – It tells about the goodness of the fit. It ranges between 0 and 1. The closer the value to 1, the better it is. It explains the extent of variation of the dependent variables in the model. However, it is biased in a way that it never decreases(even on adding variables).
- **Adj Rsquare –** This parameter has a penalising factor(the no. of regressors) and it always decreases or stays identical to the previous value as the number of independent variables increases. If its value keeps increasing on removing the unnecessary parameters go ahead with the model or stop and revert.
- **F statistic** – It is used to compare two variances and is always greater than 0. It is formulated as $v1^2/v2^2$. In regression, it is the ratio of the explained to the unexplained variance of the model.
- **AIC and BIC –** AIC stands for Akaike's information criterion and BIC stands for Bayesian information criterion Both these parameters depend on the likelihood function L.
- **Skew** – Informs about the data symmetry about the mean.
- **Kurtosis** – It measures the shape of the distribution i.e.the amount of data close to the mean than far away from the mean.

- **Omnibus – D'Angostino's test**. It provides a combined statistical test for the presence of skewness and kurtosis.
- **Log-likelihood** – It is the log of the likelihood function.

PARAMETER	VALUE
R SQAURED	HIGHER THE BETTER
ADJ R SQAURED	HIGHER THE BETTER
F-STATISTIC	HIGHER THE BETTER
AIC	LOWER THE BETTER
BIC	LOWER THE BETTER
SKEW	LOWER THE BETTER
KURTOSIS	HIGHER THE BETTER

This image shows the preferred relative values of the parameters.

References:- Machine Learning course by Kirill Eremenko and Hadelin de Ponteves.

Recommended Posts:

Understanding Logistic Regression

Multiple Linear Regression using R

Regression and Classification | Supervised Machine Learning

Linear Regression using PyTorch

Identifying handwritten digits using Logistic Regression in PyTorch

Simple Linear-Regression using R

Linear Regression Using Tensorflow

ML | Linear Regression

Gradient Descent in Linear Regression

Mathematical explanation for Linear Regression working

ML | Boston Housing Kaggle Challenge with Linear Regression

ML | Normal Equation in Linear Regression

Python | Implementation of Polynomial Regression

Python | Decision Tree Regression using sklearn

ML | Logistic Regression using Tensorflow

Choco_Chips

Check out this Author's contributed articles.

Improved By : shubham_singh, devansh007kaushik

Article Tags : Machine Learning

Practice Tags : Machine Learning

1

0

No votes yet.

To-do Done

Feedback/ Suggest Improvement Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments