**Multicollinearity in Linear Regression**

**What is Multicollinearity?**

Multicollinearity refers to a scenario when the multiple independent variables(predictors) in dataset have strong correlation between other predictors apart from correlation with target or response variable(Y).

**Impact of Multicollinearity**

The Multicollinearity causes problems regression analysis. Some of the impacts are as follows,

1. Reduces the statistical significance of independent variables

2. Increases variance in the error or residuals

3. High correlation of X variables will cause problems in calculating slope(w)

**Types of Multicollinearity**

There are 2 types of Multicollinearity,

Perfect Multicollinearity

Imperfect Multicollinearity

1. **Perfect Multicollinearity**

Perfect Multicollinearity is when one or more independent variables have linear relationship with other independent variables. This is simply when the correlation of two explanatory variables is exactly 1.

2**. Imperfect Multicollinearity**

Imperfect Muticollinearity implies when the correlation is not exactly 1.

However, if the correlation(r) is near 0.5 < r < 0.9 then it is called High Multicollinearity.

If the correlation(r) is less than r < 0.5 then it is known as Low Multicollinearity.

How to solve Multicollinearity

The multicollinearity can be solved by feature interaction or dimensionality reduction method.

1. Remove some highly correlated features

2. Create interactive features by combining them

3. Perform principal component analysis to select important features.

**What kind of problem we have to solve**

Lets say you have two independent features -> Age and experience and you have to predict the salary based on those two values.

Linear Regression Best-fit line formula => Salary = B0 + B1 \* (Age) + B2 \* (Experience)

here, B0 = intercept, B1 and B2 are coefficients / slopes

Now, if you see, there can be a possibility that the age and experience variable themselves have a high correlation value (>90%) i.e. age and experience are internally correlated with each other. This affects the output 'salary', the features 'age' and 'experience' will be almost same thus implying that we are providing the same information to the output feature 'salary' which we want to compute.

This is the problem that we have to resolve.

**In this Case, we will use the Multiple Linear Regression technique 'Ordinary Least Squared'**

-Equation of Linear Regression best fit line for this dataset is: y = B0 \* 1 + B1 \* (TV) + B2 \* (Radio) + B3 \* (Newspaper)

- Whenever computing Ordinary Least Squared (OLS) => we need to compute B0(i.e. the intercept) also.

- But we dont have the B0 value here => So we will add a column for B0 value and all values in that column will be equal to 1

- To add a constant value column for B0 with all values = 1 => We will use the statsmodel library

**Example 1:**

|  |  |  |
| --- | --- | --- |
| 1 | Import related libraries | import pandas as pd  import statsmodels.api as sm |
| 2 | Read the dataset | df = pd.read\_csv("Advertising.csv")  df.head()   |  | **TV** | **Radio** | **Newspaper** | **Sales** | | --- | --- | --- | --- | --- | | 0 | 230.1 | 37.8 | 69.2 | 22.1 | | 1 | 44.5 | 39.3 | 45.1 | 10.4 | | 2 | 17.2 | 45.9 | 69.3 | 12.0 | | 3 | 151.5 | 41.3 | 58.5 | 16.5 | | 4 | 180.8 | 10.8 | 58.4 | 17.9 | |
| 3 | Splitting the data into independent and dependent features  X 🡺 independent variables => predict sales value based on these features  Y 🡺 dependent variables | X = df[['TV', 'Radio', 'Newspaper']]  y = df[['Sales']]  X.head()   |  | **TV** | **Radio** | **Newspaper** | | --- | --- | --- | --- | | 0 | 230.1 | 37.8 | 69.2 | | 1 | 44.5 | 39.3 | 45.1 | | 2 | 17.2 | 45.9 | 69.3 | | 3 | 151.5 | 41.3 | 58.5 | | 4 | 180.8 | 10.8 | 58.4 | |
| 4 |  | y.head()   | **Sales** | | --- | | 0 | 22.1 | | 1 | 10.4 | | 2 | 12.0 | | 3 | 16.5 | | 4 | 17.9 | |
| 5 | Whenever computing Ordinary Least Squared (OLS) => we need to compute B0(i.e. the intercept) also.  But we dont have the B0 value here => So we will add a column for B0 value and all values in that column will be equal to 1  To add a constant value column for B0 with all values = 1 => We will use the statsmodel library | X = sm.add\_constant(X)  X.head()   | **const** | **TV** | **Radio** | **Newspaper** | | --- | --- | --- | --- | | 0 | 1.0 | 230.1 | 37.8 | 69.2 | | 1 | 1.0 | 44.5 | 39.3 | 45.1 | | 2 | 1.0 | 17.2 | 45.9 | 69.3 | | 3 | 1.0 | 151.5 | 41.3 | 58.5 | | 4 | 1.0 | 180.8 | 10.8 | 58.4 | |
|  | -Fit an Ordinary Least Squared Model with intercept on TV and Radio.  -We will again be using the statsmodel library for this as it has the function OLS for creating the model.  -Inside the OLS method we have to give endog(output feature) and exog values(input features) as parameters | model = sm.OLS(y, X).fit()  model.summary()   |  |  |  |  | | --- | --- | --- | --- | | OLS Regression Results | | | | | Dep. Variable: | Sales | R-squared: | 0.903 | | Model: | OLS | Adj. R-squared: | 0.901 | | Method: | Least Squares | F-statistic: | 605.4 | | Date: | Tue, 16 Feb 2021 | Prob (F-statistic): | 8.13e-99 | | Time: | 16:29:57 | Log-Likelihood: | -383.34 | | No. Observations: | 200 | AIC: | 774.7 | | Df Residuals: | 196 | BIC: | 787.9 | | Df Model: | 3 |  |  | | Covariance Type: | nonrobust |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | coef | std err | t | P>|t| | [0.025 | 0.975] | | const | 4.6251 | 0.308 | 15.041 | 0.000 | 4.019 | 5.232 | | TV | 0.0544 | 0.001 | 39.592 | 0.000 | 0.052 | 0.057 | | Radio | 0.1070 | 0.008 | 12.604 | 0.000 | 0.090 | 0.124 | | Newspaper | 0.0003 | 0.006 | 0.058 | 0.954 | -0.011 | 0.012 |  |  |  |  |  | | --- | --- | --- | --- | | Omnibus: | 16.081 | Durbin-Watson: | 2.251 | | Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 27.655 | | Skew: | -0.431 | Prob(JB): | 9.88e-07 | | Kurtosis: | 4.605 | Cond. No. | 454. | |
|  | Plot independent features in terms of correlation  Through this table, we can see the correlation values among the various independent features:  \* Between TV and Radio => 0.054809  \* Between Radio and Newspaper => 0.354104  \* Between TV and Newspaper => 0.056648  This implies that none of the correlation values are >0.5.  Thus indicating that there is not much correlation between the independent features and thus no multicollinearity issue among the independent features | import matplotlib.pyplot as plt  X.iloc[:, 1:].corr()   | **TV** | **Radio** | **Newspaper** | | --- | --- | --- | | TV | 1.000000 | 0.054809 | 0.056648 | | Radio | 0.054809 | 1.000000 | 0.354104 | | Newspaper | 0.056648 | 0.354104 | 1.000000 | |
|  |  |  |

**Observations:**

When we want to detectmulticollinearity, we need **P value** and **std error** in **model.summary().**

This summary helps us understand whether there is multicollinearity / high correlation between the independent features.

According to the summary,

\* B0 = coeff of const = 4.6251

\* B1 = coeff of TV = 0.0544

\* [coeff value means that if we change the TV expenditure(i.e. input feature) by 1 unit, the change in sales(i.e. output) will be 0.0544]

\* B2 = coeff of Radio = 0.1070

\* B3 = coeff of Newspaper = 0.0003

\* B3 coeff => << 0.005 => this shows that we are making an unnecessary expenditure on Newspaper. Thus we can reduce that unnecessary expenditure done on Newspaper. Thus while creating the model, we can just drop this feature.

\* R-squared value = 0.903 => very close to 1 => the model has fitted very well

\* P value of const = 0

\* P value of TV = 0

\* P value of Radio = 0

\* P value of Newspaper = 0.954

=> Except the feature 'Newspaper' (P-value = 0.954), all the P values are less than 0.05

\* std error of const = 0.308

\* std error of TV = 0.001

\* std error of Radio = 0.008

\* std error of Newspaper = 0.006

std error => high number(>0.5) if there is multicollinearity among the independent varibles.

But here, the std error are small numbers thus indicating there is no multicollinearity among the independent variables

Example2:

**Dataset 2 => Salary Dataset with age and YOE**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Import Dataset | df\_salary = pd.read\_csv("Salary\_Data.csv")  df\_salary.head()   |  | **YearsExperience** | **Age** | **Salary** | | --- | --- | --- | --- | | 0 | 1.1 | 21.0 | 39343 | | 1 | 1.3 | 21.5 | 46205 | | 2 | 1.5 | 21.7 | 37731 | | 3 | 2.0 | 22.0 | 43525 | | 4 | 2.2 | 22.2 | 39891 | |
|  | The independent features:  **'Years of Experience'** and the **'Age'**. | X = df\_salary[['YearsExperience', 'Age']]  y = df\_salary[['Salary']]  X.head() |
|  | We have to predict **the dependent variable 'Salary'** based on these two independent features | y.head()   |  | **Salary** | | --- | --- | | 0 | 39343 | | 1 | 46205 | | 2 | 37731 | | 3 | 43525 | | 4 | 39891 | |
|  | Fitting the OLS(Ordinary Least Squared) model, similar to the previous dataset | X = sm.add\_constant(X)  model = sm.OLS(y, X).fit()  model.summary()   |  |  |  |  | | --- | --- | --- | --- | | OLS Regression Results | | | | | Dep. Variable: | Salary | R-squared: | 0.960 | | Model: | OLS | Adj. R-squared: | 0.957 | | Method: | Least Squares | F-statistic: | 323.9 | | Date: | Tue, 16 Feb 2021 | Prob (F-statistic): | 1.35e-19 | | Time: | 16:29:57 | Log-Likelihood: | -300.35 | | No. Observations: | 30 | AIC: | 606.7 | | Df Residuals: | 27 | BIC: | 610.9 | | Df Model: | 2 |  |  | | Covariance Type: | nonrobust |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | coef | std err | t | P>|t| | [0.025 | 0.975] | | const | -6661.9872 | 2.28e+04 | -0.292 | 0.773 | -5.35e+04 | 4.02e+04 | | YearsExperience | 6153.3533 | 2337.092 | 2.633 | 0.014 | 1358.037 | 1.09e+04 | | Age | 1836.0136 | 1285.034 | 1.429 | 0.165 | -800.659 | 4472.686 |  |  |  |  |  | | --- | --- | --- | --- | | Omnibus: | 2.695 | Durbin-Watson: | 1.711 | | Prob(Omnibus): | 0.260 | Jarque-Bera (JB): | 1.975 | | Skew: | 0.456 | Prob(JB): | 0.372 | | Kurtosis: | 2.135 | Cond. No. | 626. | |
|  | Confirming the multicollinearity between Age and YearsOfExperience by plotting the correlation table  With the help of this Correlation Table / Matrix we can imply that age and yearsofexperience have 98% correlation (very highly correlated).  This implies that taking one of these features will be more than enough to predict the salary. | X.iloc[:, 1:].corr()    Now the Question is which of the input features (YearsOfExperience and Age) to keep and which one to drop for the final prediction of salary  ### Remedy for this Multicollinearity problem:  \* Solution 1 : Dont do anything, keep things as it is and don't care about multicollinearity and take all the input features to create the model  \* Solution 2 : Check the P values for Age and YearsOfExperience. P value of Age > P value of YearsOfExperience. Thus drop the 'Age' feature. This will not have much effect on the model as the correlation is about 98%. Thus the whole model can be trained just by considering the feature 'YearsOfExperience' |

In this scenario, According to the summary,

\* B0 = coeff of const = -6661.9872

\* B1 = coeff of YearsofExperience = 6153.3533

\* [this coeff value means that if we change the YearsOfExperience(i.e. input feature) by 1 unit, the change in salary(i.e. output) will be 6153.3533]

\* B2 = coeff of Age = 1836.0136

\* [thus if we change the Age(i.e. input feature) by 1 unit(1 year), the change in salary(i.e. output) will be 6153.3533]

\* R-squared value = 0.960 => very close to 1 => the model has fitted very well

\* P value of const = 0.773

\* P value of YearsOfExperience = 0.014

\* P value of Age = 0.165

=> for Age => the P-value is >0.05 => Age and YearsOfExperience may have some kind of correlation

\* std error of const = 0.308

\* std error of YearsOfExperience = 2337.092

\* std error of Age = 1285.034

Here we can see the std errors of both YearsOfExperience and Age are very very high, thus indicating that there is a huge Multicollinearity among them