Multiple Linear Regression Using Python

Problem of Multicollinearity
Normality of Errors

DATA SCIENCE
INSTITUTE

Problem of Multicollinearity

Multicollinearity exists if there is strong linear relationship among the independent variables

Multicollinearity has two serious consequences:

1. Highly Unstable Model Parameters

As standard errors of their estimates are inflated

2. Model Fails to Accurately Predict for Out of Sample Data

Therefore, it is important to check for Multicollinearity in regression analysis

Detecting Multicollinearity Through VIF

VIF (Variance Inflation Factor) Method:

Dependent Variable : Y

Independent variables : X1, X2, X3, X4

Dependent Variable	Independent Variables	R^2	$1 - R^2$ = Tolerance	VIF = 1/(Tolerance)
X1	X2, X3, X4			
X2	X1, X3, X4			
X3	X1, X2, X4			
X4	X1, X2, X3			

Any VIF > 5, indicates presence of Multicollinearity

Detecting Multicollinearity in Python

```
#Importing the Data, Fitting Linear Model
import pandas as pd
perindex=pd.read_csv("Performance Index.csv")

import statsmodels.formula.api as smf
jpimodel=smf.ols('jpi~aptitude+tol+technical+general',data=perindex).fit()
```

```
#Variance Inflation Factor
from patsy import dmatrices
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Break data into left and right hand side; y and X
y, X = dmatrices('jpi ~ aptitude + tol +  technical +general',
data=perindex, return_type="dataframe")
```

- **patsy** is a library that helps in converting data frames into design matrices.
- **dmatrices** Construct two design matrices using specified formula.By convention, the first matrix is the "y" data, and the second is the "x" data.
- **var**) calculate vif

* We use the same dataset “Performance Index” which was used in previous ppt

Detecting Multicollinearity in Python

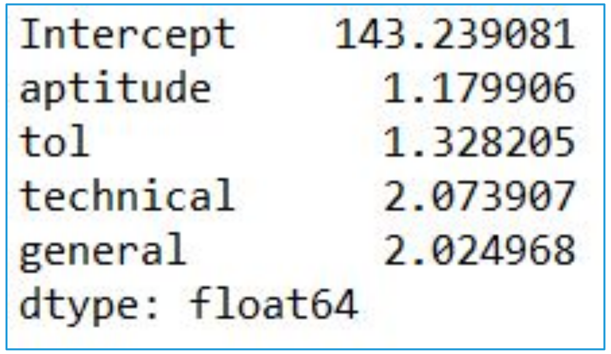
```
# Calculating VIF & getting vif with their corresponding variable
# name
```

```
vif = pd.Series([variance_inflation_factor(X.values, i)for i in
range(X.shape[1])],index=X.columns)

vif
```

variance_inflation_factor() calculates VIFs.

```
# Output
```



```
Intercept    143.239081
aptitude       1.179906
tol            1.328205
technical      2.073907
general        2.024968
dtype: float64
```

Interpretation :
All VIFs are less than 5, Multicollinearity is not present.

Multicollinearity – Remedial Measures

The problem of Multicollinearity can be solved by different approaches:

- Drop one of the independent variables, which is explained by others
- Use Principal Component Regression in case of severe Multicollinearity
- Use Ridge Regression

* Dropping a variable may not be a good idea if many VIFs are large.
Principal Component Method will be discussed in detail under Data Reduction and Segmentation

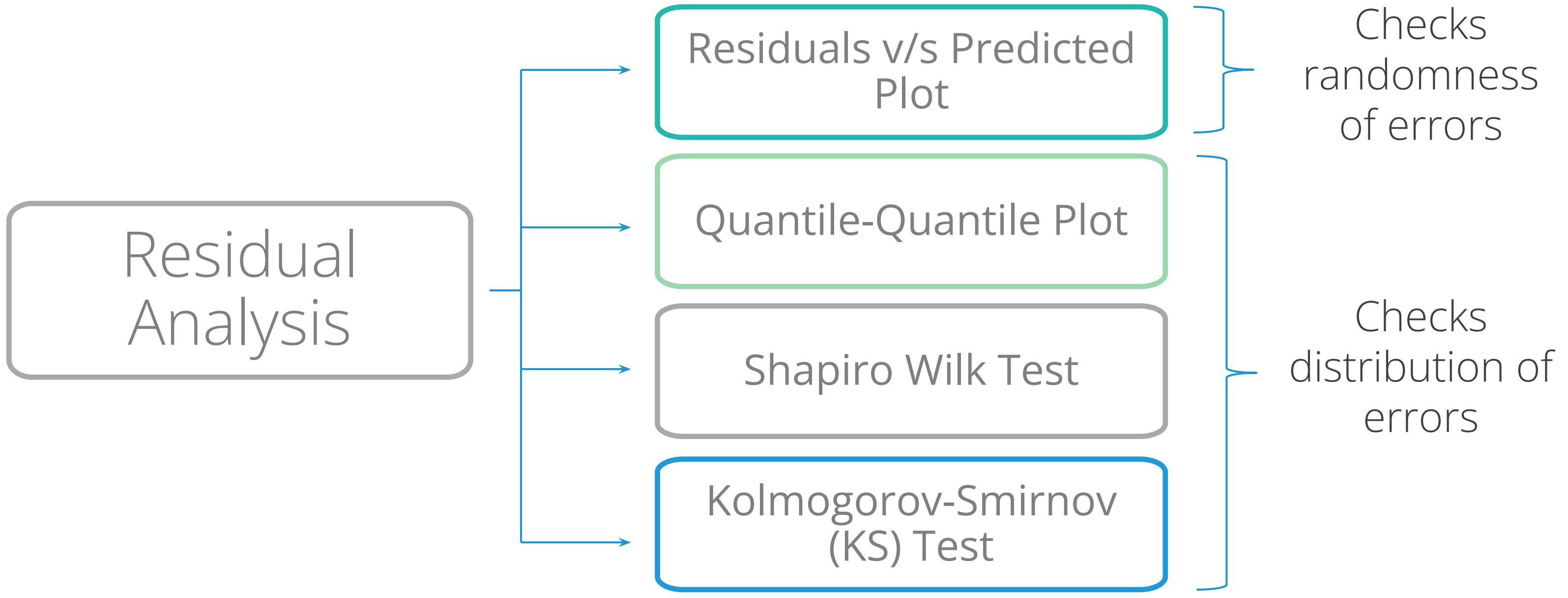
Normality of Errors

- The errors in Multiple Linear Regression are assumed to follow Normal Distribution.

- If Normality of Errors is not true then statistical tests and associated P values based on F and t distribution are not reliable.

Residual Analysis

Observed Value – Predicted value = Residual


Residual Analysis
Residuals v/s Predicted Plot
Quantile-Quantile Plot
Shapiro Wilk Test
Kolmogorov-Smirnov (KS) Test
Checks randomness of errors
Checks distribution of errors

DATA SCIENCE INSTITUTE

Residual Analysis for Performance Index Data

Continuing with the "**Performance Index** " data,

- **Model** job performance index (**jpi**) based on aptitude score (**aptitude**), test of language (**tol**), technical knowledge (**technical**) and general information (**general**)
- Get fitted values and residuals.
- Analyse the distribution of residuals

Residual v/s Predicted Plot in Python

```
#Importing the Data, Fitting Linear Model and Calculating Fitted
Values and Residuals
```

```
import pandas as pd
perindex= pd.read_csv("Performance Index.csv")

import statsmodels.formula.api as smf
jpimodel = smf.ols('jpi ~ tol + aptitude + technical +general',
data=perindex).fit()

perindex = perindex.assign(pred=pd.Series(jpimodel.fittedvalues))
perindex = perindex.assign(res=pd.Series(jpimodel.resid))
```

- **ols()** fits a linear regression.
- **fittedvalues()** and **resid()** fetch fitted values and residuals

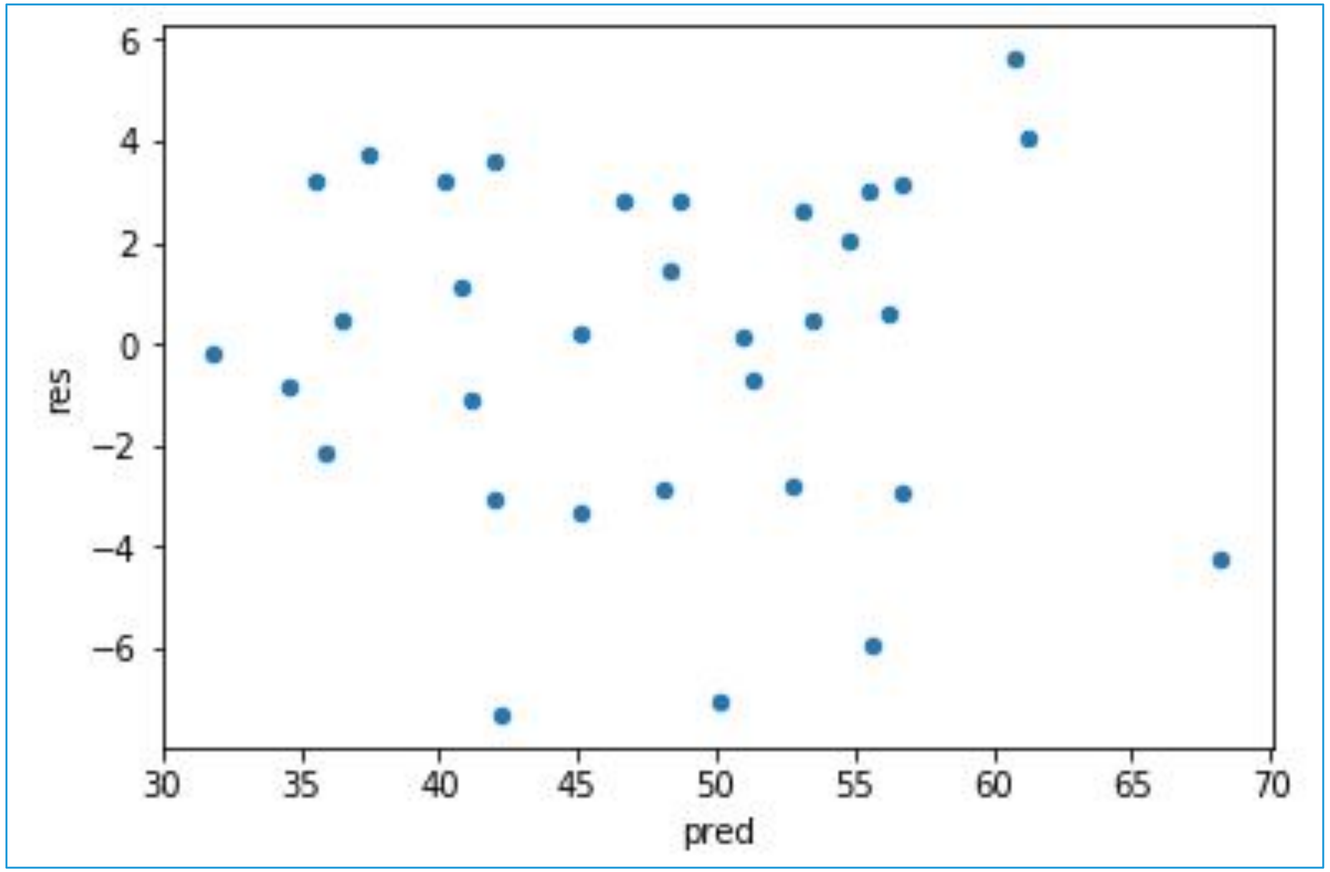
```
#Residuals v/s Predicted Plot
```

```
perindex.plot.scatter(x='pred', y='res')
```

.plot.scatter() is used to obtain scatter plot of predicted values against residuals.

Residual v/s Predicted Plot in Python

```
# Output
```



Interpretation:

- Residuals in our model are randomly distributed which indicates presence of Homoscedasticity

DATA SCIENCE INSTITUTE

QQ Plot

- The Quantile-Quantile (QQ) Plot is a powerful graphical tool for assessing normality.
- Quantiles are calculated using sample data and plotted against expected quantiles under Normal distribution.

High Correlation between Sample Quantiles and Theoretical Quantiles → Normality

- If the data are truly sampled from a Gaussian (Normal) distribution, the QQ plot will be linear.

QQ Plot in Python

```
#QQ Plot
```

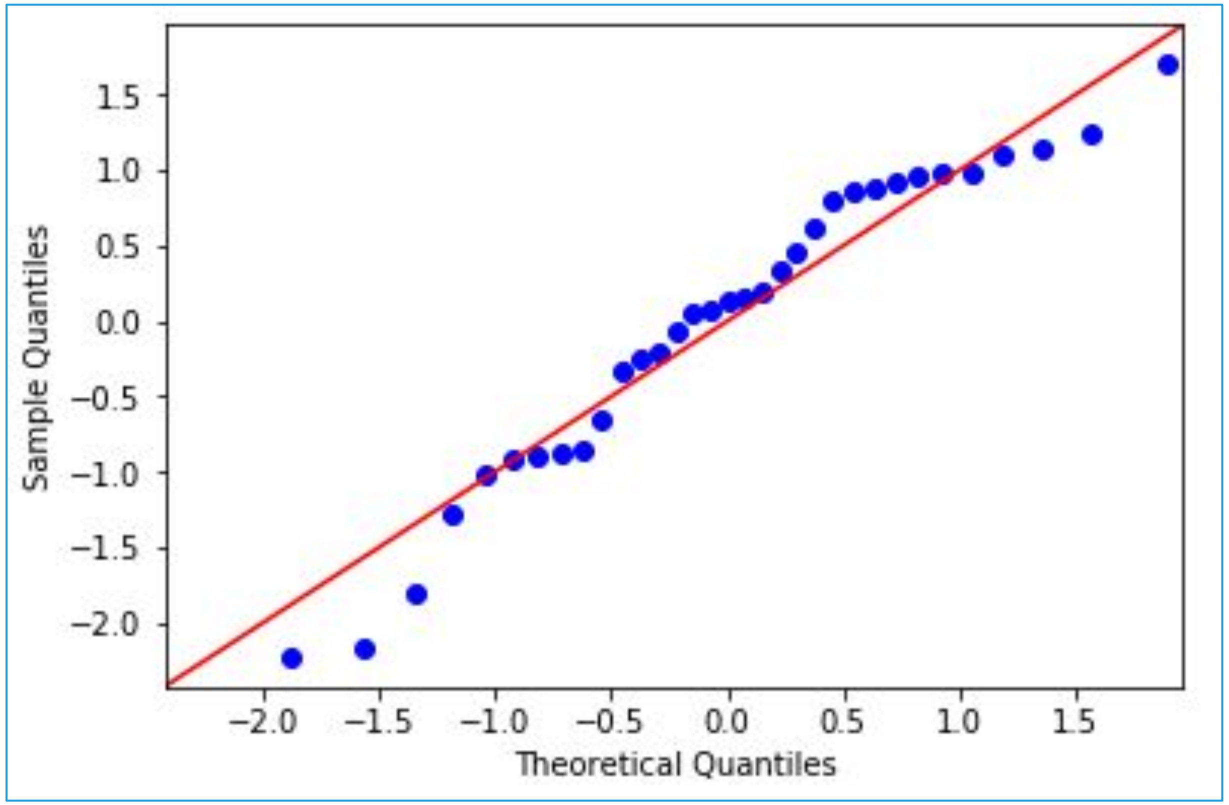
```
import statsmodels.api as sm
fig = sm.graphics.qqplot(perindex.res, line='45', fit=True)
```

- **qqplot()** produces a plot with theoretical quantiles on x axis against the sample quantiles on y axis. Column for which normality is being tested is specified in the first argument.
- **line=** is an argument that adds reference line to the qqplot. Here it adds a 45-degree line
- **fit=True** indicates, parameters are fit using the distribution's fit() method.

DATA SCIENCE INSTITUTE

QQ Plot in Python

```
# Output
```



Interpretation:

- Most of these points are close to the line except few values indicating no serious deviation from Normality.

Shapiro Wilk Test

Objective	To **correlate**, sample ordered values with expected Normal scores in order **to test normality of the sample**

Null Hypothesis (H_0): Sample is drawn from Normal Population

Alternate Hypothesis (H_1): Not H_0

Test Statistic	
Decision Criteria	Reject the null hypothesis if p-value < 0.05

DATA SCIENCE INSTITUTE

Shapiro Wilk Test in Python

```
# Shapiro Wilk Test
import scipy as sp
sp.stats.shapiro(perindex.res)
```

shapiro() from scipy package, returns correlation coefficient w and p-value.

```
# Output
(0.9498621821403503, 0.1318102478981018)
```

Interpretation:
p-value>0.05, Do not reject H_0. Normality can be assumed.

DATA SCIENCE INSTITUTE

Case Study - Modelling Resale Price of Cars

Background

- A car garage has old cars for resale. They keep records for different models of cars and their specifications.

Objective

- To predict the resale price based on the information available about the engine size, horse power, weight and years of use of the cars

Available Information

- Records -26
- Independent Variables: engine size, horse power, weight and years
- Dependent Variable: resale price

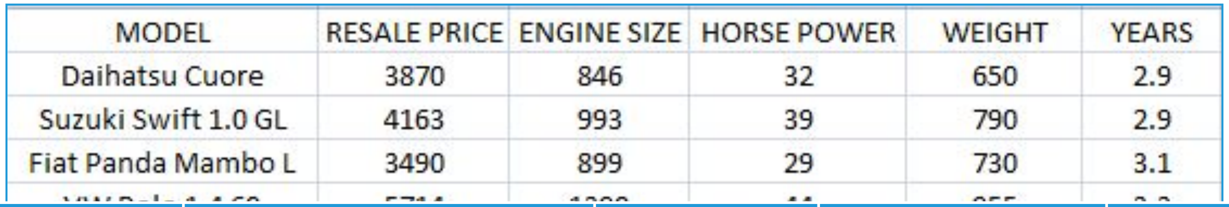
DATA SCIENCE INSTITUTE

Data Snapshot

Car price data

Dependent variable

Independent variables



MODEL	RESALE PRICE	ENGINE SIZE	HORSE POWER	WEIGHT	YEARS
Daihatsu Cuore	3870	846	32	650	2.9
Suzuki Swift 1.0 GL	4163	993	39	790	2.9
Fiat Panda Mambo L	3490	899	29	730	3.1

Observatio

Columns	Description	Type	Measurement	Possible values
MODEL	Model of the car	character	-	-
RESALE PRICE	Resale price	numeric	Euro	positive values
ENGINE SIZE	Size of the engine	numeric	cc	positive values
HORSE POWER	Power of the engine	numeric	kW	positive values
WEIGHT	Weight of the car	numeric	kg	positive values
YEARS	Number of years in use	numeric	-	positive values

DATA SCIENCE INSTITUTE

Correlation Matrix

```
# Importing the Data
```

```
ridgedata=pd.read_csv(”car price data.csv")
```

```
# Graphical representation of data
# Install and load package “seaborn”
```

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(ridgedata[['MODEL', 'RESALE PRICE', 'ENGINE SIZE', 'HORSE
POWER','YEARS']]);plt.title('Fig.No. 2: ScatterPlot Matrix')
```

pairplot() in the package seaborn is used to plot the scatter plot matrix

Scatter Plot Matrix

```
# Output
```

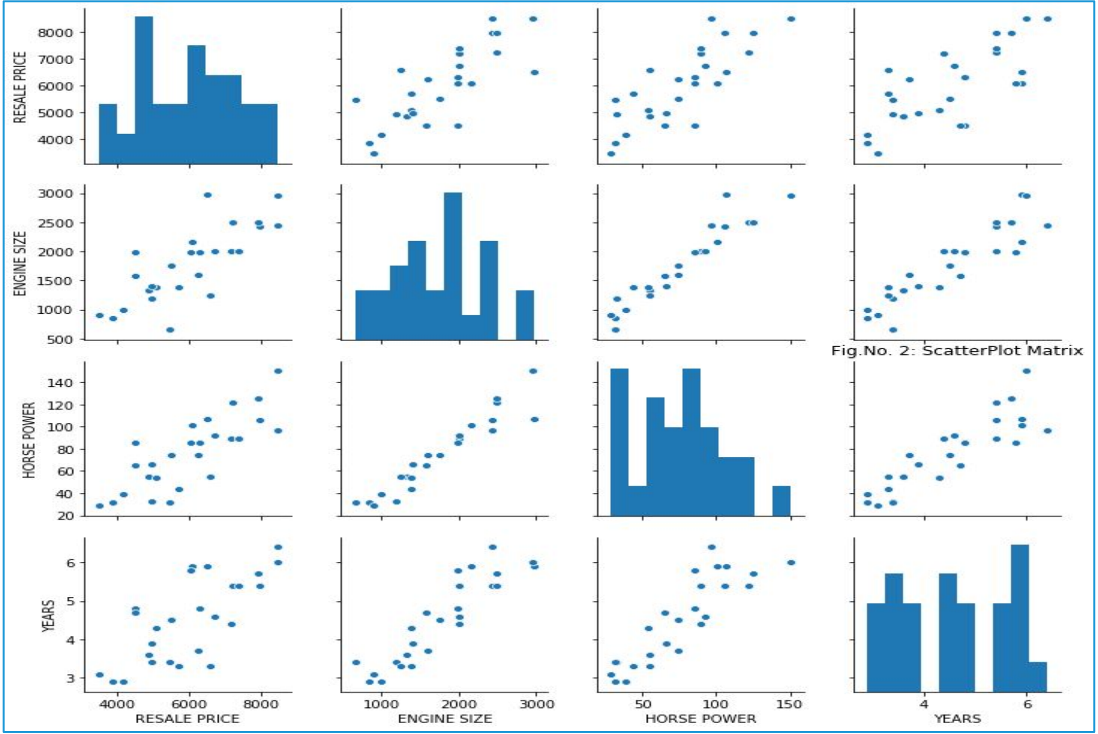


Fig.No. 2: ScatterPlot Matrix

Interpretation :

 The independent variables have high positive correlation among themselves .

Detecting Multicollinearity in Python

```
#Importing the Data, Fitting Linear Model
```

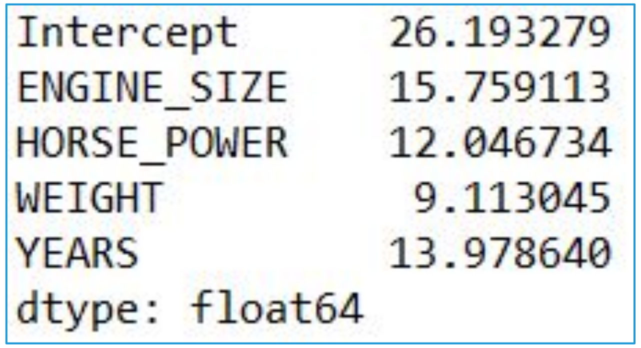
```
ridgedata.columns = [c.replace(' ', '_') for c in ridgedata.columns]
model = smf.ols('RESALE_PRICE~ENGINE_SIZE+ HORSE_POWER + WEIGHT + YEARS',
data = ridgedata).fit()
```

In pandas, the **column names cannot contain spaces** in between. Hence, before applying **ols()** remove spaces from column names wherever required.

```
#Variance Inflation Factor
```

```
y, X = dmatrices('RESALE_PRICE~ENGINE_SIZE+ HORSE_POWER + WEIGHT +
YEARS', data=ridgedata, return_type="dataframe")
vif = pd.Series([variance_inflation_factor(X.values, i)for i in
range(X.shape[1])],index=X.columns)
vif
```

```
# Output
```



```
Intercept        26.193279
ENGINE_SIZE      15.759113
HORSE_POWER      12.046734
WEIGHT            9.113045
YEARS            13.978640
dtype: float64
```

Interpretation:

VIF values for all the variables are greater than 5, hence we can conclude that there exist Multicollinearity between the independent variables.

DATA SCIENCE
INSTITUTE