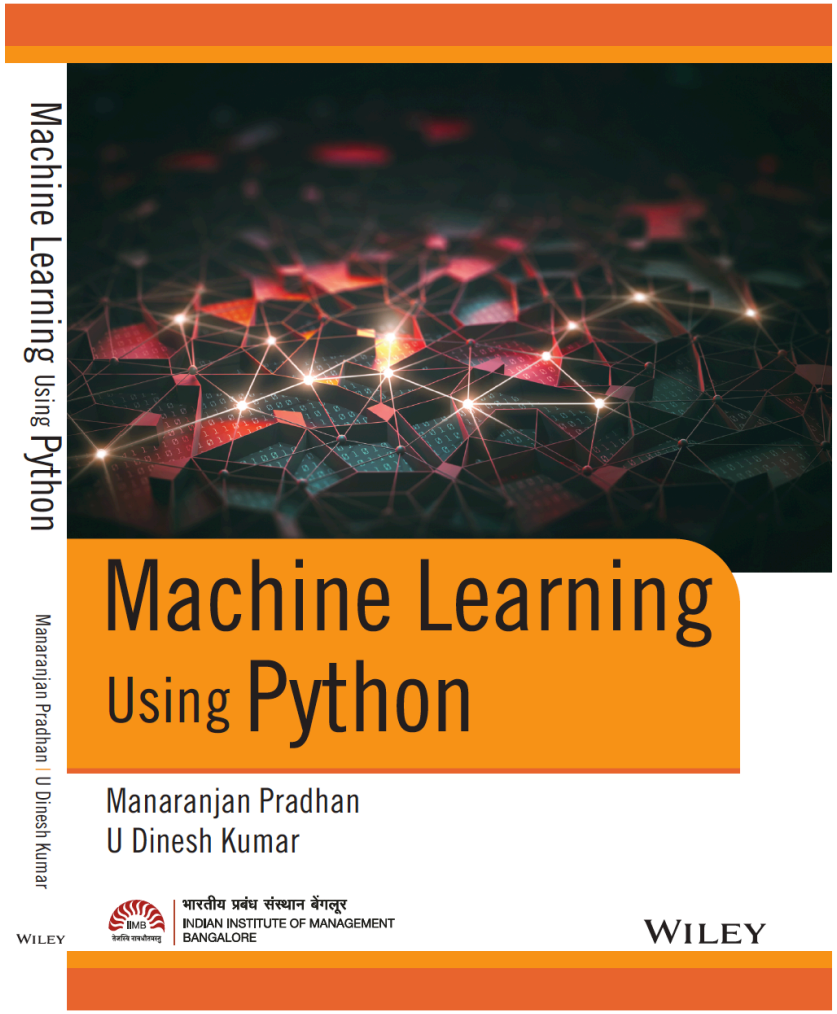

Chapter 04: Linear Regression

Prepared By: Purvi Tiwari

Learning Objectives

- Understanding the concepts of simple and multiple linear regression and its applications in predictive analytics.
- Building simple and multiple linear regression models using Python package *statsmodel.*
- Learning to incorporate categorical variables in regression model.
- Learning regression model diagnostics and performing residual analysis, identification of outliers, and influential observations.

Simple Linear Regression

- A statistical technique used for finding the existence of an association relationship between a dependent variable (response variable) and an independent variable (feature).
- It requires the knowledge of both the response variable and the feature variables in the training dataset.
- It can establish that change in the value of the response variable is associated with change in the value of feature X.
- It cannot establish casual relationship between two variables.

Simple Linear Regression(Cntd.)

- Few examples of simple and multiple linear regression problems:
  1. A hospital in interested in finding how the total cost of a patient for a treatment varies with the body weight of the patient.
  2. Insurance companies would like to understand the association between healthcare costs and ageing.
  3. An organization may be interested in finding the relationship between revenue generated from a product and features such as the price, money spent on promotion, competitor's price, and promotion expenses.
  4. Restaurants would like to know the relationship between the customer waiting time after placing the order and the revenue.

Steps in Building Regression Model

- Collect/ Extract Data
- Pre-Process the Data
  1. Data imputation techniques are used to deal with missing data.
  2. New variables (such as ratio/product of variables) can be derived.
  3. Categorical data is pre-processed using dummy variables.
- Dividing Data into Training and Validation Datasets
  1. The proportion of training dataset is usually between 70-80% of the data
  2. The remaining data is treated as validation data.
  3. Subsets may be created using random/ stratified sampling methods

Steps in Building Regression Model (Cntd.)

- Perform Descriptive Analytics
  1. It helps in understanding the variability in the model.
  2. Identifying outliers in the data.
  3. Describing the functional relationship between the outcome variable and features.
- Build the Model
  1. Model is built using the training dataset.
  2. Method of Ordinary Least Squares (OLS) is used to estimate the regression parameters.
- Perform Model Diagnostics
  1. Model is validated for all model assumptions
  2. If the model assumptions are violated, then the modeler must use remedial measure.

Steps in Building Regression Model (Cntd.)

- Validate the Model and Measure Model Accuracy
  1. Checking over-fitting
  2. May be cross-validated using multiple training and test datasets.
- Decide on Model Deployment
  1. Develop a deployment strategy
  2. Build actionable items and business rules that can be used by the organization.

Building Simple Linear Regression Model

- Functional form of Simple Linear Regression (SLR) for a dataset with n observations (X, Y), where i=1,2,…,n

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

- The error term $\varepsilon_i = Y_i - \beta_0 - \beta_1 X_i$

Building Simple Linear Regression Model (Cntd.)

- The regression parameters are estimated by minimizing the sum of squared errors (SSE)

$$\mathrm{SSE} = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)$$

- This method is known as method of ordinary least square (OLS).

Building Simple Linear Regression Model (Cntd.)

- The estimated parameter values are given by

$$\widehat{\beta_1} = \sum_{i=1}^{n} \frac{(X_i - \overline{X})\,(Y_i - \overline{Y})}{(X_i - \overline{X})^2}$$

$$\widehat{\beta_0} = \overline{Y} - \beta_1 \overline{X}$$

Where and are the estimated values of the regression parameters and respectively.

Assumptions of the Linear Regression Model

1. The errors or residuals are assumed to follow a normal distribution with expected value of error E()=0.
2. The variance of error, VAR(), is constant for various values of independent variable X. This is known as homoscedasticity. When the variance is not constant, it is called heteroscedasticity.
3. The error and independent variable are uncorrelated.
4. The functional relationship between the outcome variable and feature is correctly defined.

Properties of Simple Linear Regression

- The mean value of Y for given X ,

$$E(Y_i \mid X) = \widehat{\beta}_0 + \widehat{\beta}_1 X$$

- Y follows a normal distribution with mean

$$\widehat{\beta}_0 + \widehat{\beta}_1 X$$

- and variance

$$\mathrm{VAR}(\varepsilon_i)$$

Example

- File name: *MBA Salary.csv* contains the salary of 50 graduating MBA students of a Business School in 2016 and their corresponding percentage marks in grade 10. Develop an SLR model to understand and predict salary based on the percentage of marks in Grade 10.

Example (Cntd.)

- Steps for building a regression model using Python:

1. Import *pandas* and *numpy* libraries.
2. Use *read_csv* to load the dataset into DataFrame.
3. Identify the feture(s) (X) and outcome (Y) variable in the DataFrame for building the model.
4. Split the dataset into training and validation sets using *train_test_split().*
5. Import *statsmodel* library and fit the model using *OLS()* method.
6. Print model summary and conduct model diagnostics.

Example (Cntd.)

- Importing *pandas* and *numpy* libraries.

```
import pandas as pd
import numpy as np
## Setting pandas print option to print decimal values upto
   4 decimal places
np.set_printoptions(precision=4, linewidth=100)
```

Example (Cntd.)

- Loading the dataset.

```
mba_salary_df = pd.read_csv( 'MBA Salary.csv' )
mba_salary_df.head( 10 )
```

	S. No.	Percentage in Grade 10	Salary
0	1	62.00	270000
1	2	76.33	200000
2	3	72.00	240000
3	4	60.00	250000
4	5	61.00	180000

	S. No.	Percentage in Grade 10	Salary
5	6	55.00	300000
6	7	70.00	260000
7	8	68.00	235000
8	9	82.80	425000
9	10	59.00	240000

Example (Cntd.)

- Printing more information about the dataset

```
mba_salary_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 3 columns):
S. No.                    50 non-null int64
Percentage in Grade 10    50 non-null float64
Salary                    50 non-null int64
dtypes: float64(1), int64(2)
memory usage: 1.2 KB
```

Example (Cntd.)

- Creating Feature Set (X) and Outcome Variable (Y)

```
import statsmodels.api as sm
X = sm.add_constant( mba_salary_df['Percentage in Grade 10'] )
X.head(5)
```

	Const	Percentage in Grade 10
0	1.0	62.00
1	1.0	76.33
2	1.0	72.00
3	1.0	60.00
4	1.0	61.00

```
Y = mba_salary_df['Salary']
```

Example (Cntd.)

- Splitting the Dataset into Training and Validation Sets
- *train_test_split()* function from *skelearn.model_selection* module provides the ability to split the dataset randomly into training and validation datasets.
- *train_test_split()* method returns four variable as below
  1. *train_X* contains X features of the training set
  2. *train_Y* contains the values of response variable for the training set
  3. *train_X* contains X features of the test set
  4. *train_Y* contains the values of response variable for the test set.

Example (Cntd.)

```
from sklearn.model_selection import train_test_split
```

```
train_X, test_X, train_y, test_y = train_test_split( X,
                                                     Y,
                                                train_size = 0.8,
                                                random_state = 100 )
```

- *train_size = 0.8* implies 80% of the data is used for training the model and the remaining 20% is used for validating the model

Example (Cntd.)

- Fitting the Model

```
mba_salary_lm = sm.OLS( train_y, train_X ).fit()
```

- Printing estimated parameters and interpreting them

```
print( mba_salary_lm.params )
```

```
Const                      30587.285652
Percentage in Grade 10      3560.587383
dtype: float64
```

Example (Cntd.)

- The estimated (predicted) model can be written as

```
MBA Salary = 30587.285 + 3560.587 * (Percentage in Grade 10)
```

- The equation can be interpreted as follows: For every 1% increase in Grade 10, the salary of the MBA students will increase by 3560.587

Example (Cntd.)

- Complete code for building regression model

```
# Importing all required libraries for building the regression model
import pandas as pd import numpy as np
import statsmodels.api as sm
from sklearn.model_selection import train_test_split

# Load the dataset into dataframe
mba_salary_df = pd.read_csv( 'MBA Salary.csv' )

# Add constant term of 1 to the dataset
X = sm.add_constant( mba_salary_df['Percentage in Grade 10'] )

# Split dataset into train and test set into 80:20 respectively
train_X, test_X, train_y, test_y = train_test_split( X,
                                                     Y,
                                                train_size = 0.8,
                                                random_state = 100 )

# Fit the regression model
mba_salary_lm = sm.OLS( train_y, train_X ).fit()

# Print the model parameters
print( mba_salary_lm.params )
```

Model Diagnostics

- The following measures are used to validate the simple linear regression models

1. Co-efficient of determination (R-squared)
2. Hypothesis test for the regression coefficient
3. Analysis of variance for overall model validity (important for multiple linear regression)
4. Residual analysis to validate the regression model assumptions
5. Outlier analysis, since the presence of outliers can significantly impact the regression parameters.

Model Diagnostics (Cntd.)

1. Co-efficient of Determination (R-Squared)
   - Measures the percentage of variation in Y explained by the model.

$$\underbrace{Y_i}_{\text{Variation in } Y} = \underbrace{\beta_0 + \beta_1 X_i}_{\substack{\text{Variation in } Y \text{ explained} \\ \text{by the model}}} + \underbrace{\varepsilon_i}_{\substack{\text{Variation in } Y \text{ not explained} \\ \text{by the model}}}$$

   - Mathematically

$$\underbrace{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}_{SST} = \underbrace{\sum_{i=1}^{n}(\widehat{Y}_i - \overline{Y})^2}_{SSR} + \underbrace{\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2}_{SSE}$$

Model Diagnostics (Cntd.)

- SST – sum of squares of total variation
- SSR – sum of squares of explained variation due to the model
- SSE – sum of squares of unexplained variation (error)
- Co-efficient of determination is given by

$$R\text{-Squared} = \frac{\text{SSR}}{\text{SST}} = 1 - \frac{\text{SSE}}{\text{SST}}$$

Model Diagnostics (Cntd.)

The co-efficient of determination (R-squared) has the following properties:

- The value of R-squared lies between 0 and 1.
- Mathematically, R-squared (R^2) is square of correlation coefficient , where r is the Pearson correlation co-efficient.
- Higher R-squared indicates better fit; However, one should be careful about the spurious relationship.

Model Diagnostics (Cntd.)

2. Hypothesis Test for Regression Co-efficient

- The regression coefficient (β_1) captures the existence of a linear relationship between the outcome variable and the feature.
- If β_1=0, there is no statistically significant linear relationship between the two variables.
- Sampling distribution of β_1 is a t-distribution (Kutner et al., 2013; U Dinesh Kumar, 2017)
- The null and alternative hypotheses are

Model Diagnostics (Cntd.)

- The corresponding test statistic is given by

$$H_0: \beta_1 = 0$$
$$H_A: \beta_1 \neq 0$$

- t-statistic is given by

$$t_{\alpha/2,n-2} = \frac{\widehat{\beta}_1}{S_e(\widehat{\beta}_1)}$$

- The standard error of estimate of the regression co-efficient is given by

$$S_e(\widehat{\beta}_1) = \frac{S_e}{\sqrt{(X_i - \bar{X})^2}}$$

- Where S is the standard error of the estimated value of Y is given by

$$S_e = \sqrt{\frac{(Y_i - \widehat{Y}_i)}{n-2}}$$

Model Diagnostics (Cntd.)

3. Analysis of Variance (ANOVA) in Regression Analysis

- In case of multiple linear regression model with k-features, ANOVA is used to check the overall validity of the regression model
- The null and alternate hypotheses are given by

$$H_0: \beta_1 = \beta_2 = \ldots = \beta_k = 0$$

H_A: Not all regression coefficients are zero

- The corresponding F-statistic is given by

$$F = \frac{\text{MSR}}{\text{MSE}} = \frac{\text{SSR}/k}{\text{SSE}/(n-k-1)}$$

- Where MSR and MSE are mean squared regression and mean squared error respectively.

Model Diagnostics (Cntd.)

4. Regression Model Summary using Python

```
mba_salary_lm.summary2()
```

TABLE 4.2 Model summary: Simple linear regression model

Model:	OLS	Adj. R-squared:	0.190
Dependent Variable:	Salary	AIC:	1008.8680
Date:	2018-04-08 07:27	BIC:	1012.2458
No. Observations:	40	Log-Likelihood:	–502.43
Df Model:	1	F-statistic:	10.16
Df Residuals:	38	Prob (F-statistic):	0.00287
R-squared:	0.211	Scale:	5.0121e+09

Model Diagnostics (Cntd.)

	Coef.	Std.Err.	*t*	$P > \|t\|$	[0.025	0.975]
const	30587.2857	71869.4497	0.4256	0.6728	−114904.8089	176079.3802
Percentage in Grade 10	3560.5874	1116.9258	3.1878	0.0029	1299.4892	5821.6855

Omnibus:	2.048	Durbin-Watson:	2.611
Prob(Omnibus):	0.359	Jarque-Bera (JB):	1.724
Skew:	0.369	Prob(JB):	0.422
Kurtosis:	2.300	Condition No.:	413

Model Diagnostics (Cntd.)

- Inference form the summary output
  1. The model R-squared value is 0.211, that is, the model explains 21.1% of the variation in salary.
  2. The p-value for the t-test is 0.0029 which indicates that there is a statistically significant relationship (at significance value =0.05) between the feature, percentage in grade 10, and salary.
  3. The probability value of F-statistic of the model is 0.0029 which indicates that the overall model is statistically significant.
- Note - In a SLR, the p-value for t-test and F-test will be the same since the null hypothesis is the same ($F = t^2$)

Model Diagnostics (Cntd.)

5. Residual Analysis

- Residuals or errors are the difference between the actual value of the outcome variable and the predicted value.
- Residual analysis is performed to check the following
  - The residuals are normally distributed.
  - Variance of residual is constant (homoscedasticity).
  - The functional form of regression is correctly specified.
  - There are no outliers.

Model Diagnostics (Cntd.)

5.1 *check for normal distribution of residual*

- P-P plot is used to check the distribution of the residuals matches with that of a normal distribution.

```
import matplotlib.pyplot as plt
import seaborn as sn
%matplotlib inline
```

```
mba_salary_resid = mba_salary_lm.resid
probplot = sm.ProbPlot(mba_salary_resid)
plt.figure( figsize = (8, 6))
probplot.ppplot( line='45' )
plt.title( "Fig 4.1 - Normal P-P Plot of Regression Standardized
           Residuals" )
plt.show()
```

Model Diagnostics (Cntd.)

5.1 *check for normal distribution of residual*

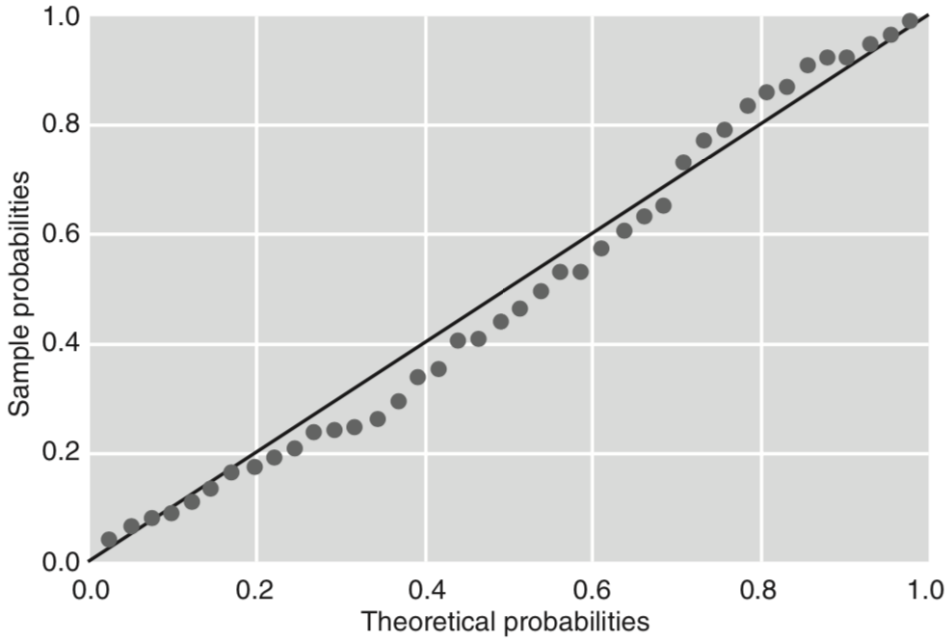

FIGURE 4.1 Normal P-P plot of regression standardized residuals.

- The diagonal line is the cumulative distribution of a normal distribution.
- The dotted line represents the cumulative distribution of the residuals.
- Hence, the residuals follow an approximate normal distribution.

Model Diagnostics (Cntd.)

5.2 *test for homoscedasticity*

- Homoscedasticity can be observed by drawing residual plot.
  - If there is heteroscedasticity, then a funnel type shape in the residual plot can be expected.

```
def get_standardized_values( vals ):
    return (vals - vals.mean())/vals.std()
```

```
plt.scatter( get_standardized_values( mba_salary_lm.fittedvalues ),
           get_standardized_values( mba_salary_resid ))
plt.title( "Fig 4.2 - Residual Plot: MBA Salary Prediction" );
plt.xlabel( "Standardized predicted values" )
plt.ylabel( "Standardized Residuals" );
```

Model Diagnostics (Cntd.)

5.2 *test for homoscedasticity*

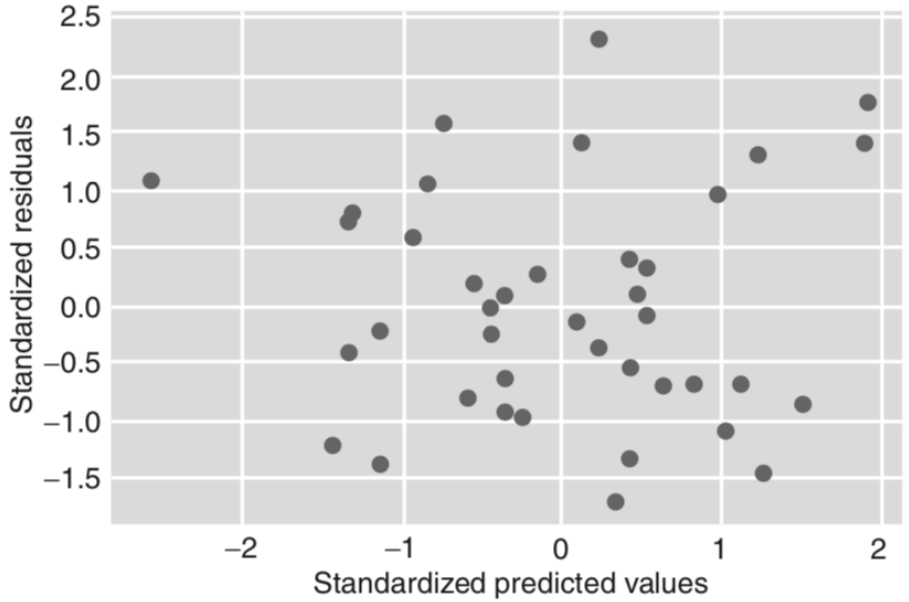

FIGURE 4.2 Residual plot: MBA salary prediction.

- The residuals have constant variance (homoscedasticity).

Model Diagnostics (Cntd.)

6. Outlier Analysis

- outliers are observations whose values show a large deviation from the mean value.
- The following distance measures are useful in identifying influential observations:
  1. Z-Score
  2. Mahalanobis Distance
  3. Cook's distance
  4. Leverage Values

Model Diagnostics (Cntd.)

6.1 Z-Score

- the standardized distance of an observation from its mean value
- For the predicted value of the dependent variable Y, the Z-score is given by

$$Z = \frac{Y_i - \bar{Y}}{\sigma_Y}$$

- Where Y_i is the predicted value of Y for i^{th} observation, $\bar{Y}$ is the mean value of Y, σ_Y is the variance of Y.
- Any observation with a Z-score of more than 3 may be flagged as an outlier.

Model Diagnostics (Cntd.)

6.1 Z-Score

```
from scipy.stats import zscore
```

```
mba_salary_df['z_score_salary'] = zscore( mba_salary_df.Salary )
```

```
mba_salary_df[ (mba_salary_df.z_score_salary > 3.0) | (mba_salary_
df.z_score_salary < -3.0) ]
```

S. No.	Percentage in Grade 10	Salary	z_score_salary

- So, there are no observations that are outliers as per the Z-Score.

Model Diagnostics (Cntd.)

6.2 Cook's Distance

- It measures how much the predicted value of the dependent variable changes for all the observations in the sample, when a particular observation is excluded from the sample for the estimation of regression parameters.
- Cook's distance value of more than 1 indicates highly influential observation.

```
import numpy as np

mba_influence = mba_salary_lm.get_influence()
(c, p) = mba_influence.cooks_distance

plt.stem(np.arange( len( train_X) ),
         np.round( c, 3 ),
         markerfmt="," );
plt.title( "Figure 4.3 - Cooks distance for all observations in MBA
           Salaray data set" );
plt.xlabel("Row index")
plt.ylabel("Cooks Distance");
```

Model Diagnostics (Cntd.)

6.2 Cook's Distance

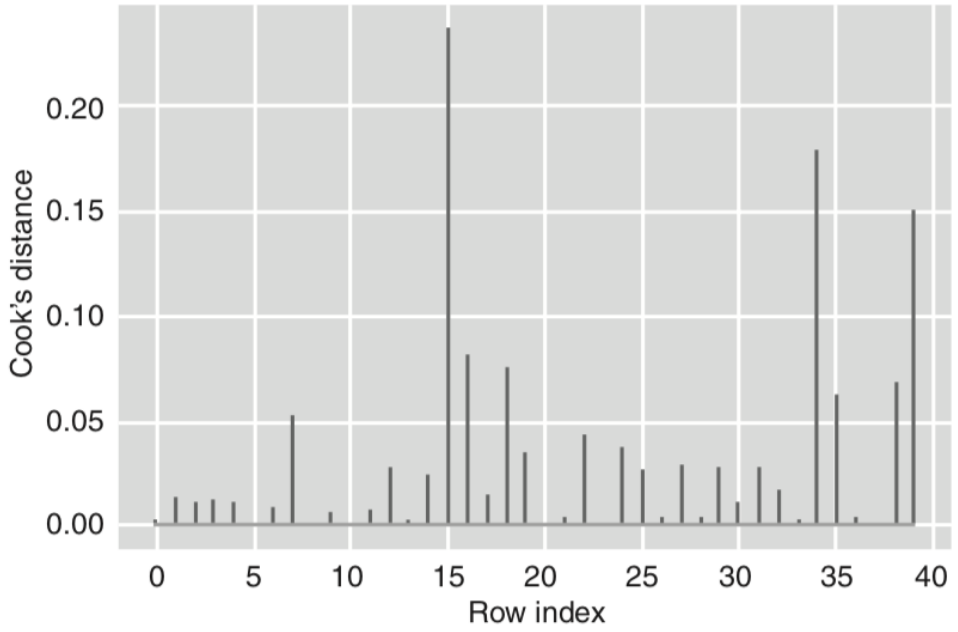

FIGURE 4.3 Cook's distance for all observations in MBA salary dataset.

- None of the observations' Cook's distance exceed 1 and hence none of them are outliers.

Model Diagnostics (Cntd.)

6.3 Leverage Values

- It measures the influence of that observation on the overall fit of the regression function and is related to Mahalanobis distance.
- Leverage value of more than 3(k+1)/n is treated as highly influential observation, where k is the number of features in the model and n is the sample size.

```
from statsmodels.graphics.regressionplots import influence_plot

fig, ax = plt.subplots( figsize=(8,6) )
influence_plot( mba_salary_lm, ax = ax )
plt.title("Figure 4.4 - Leverage Value Vs Residuals")
plt.show();
```

Model Diagnostics (Cntd.)

6.3 Leverage Values

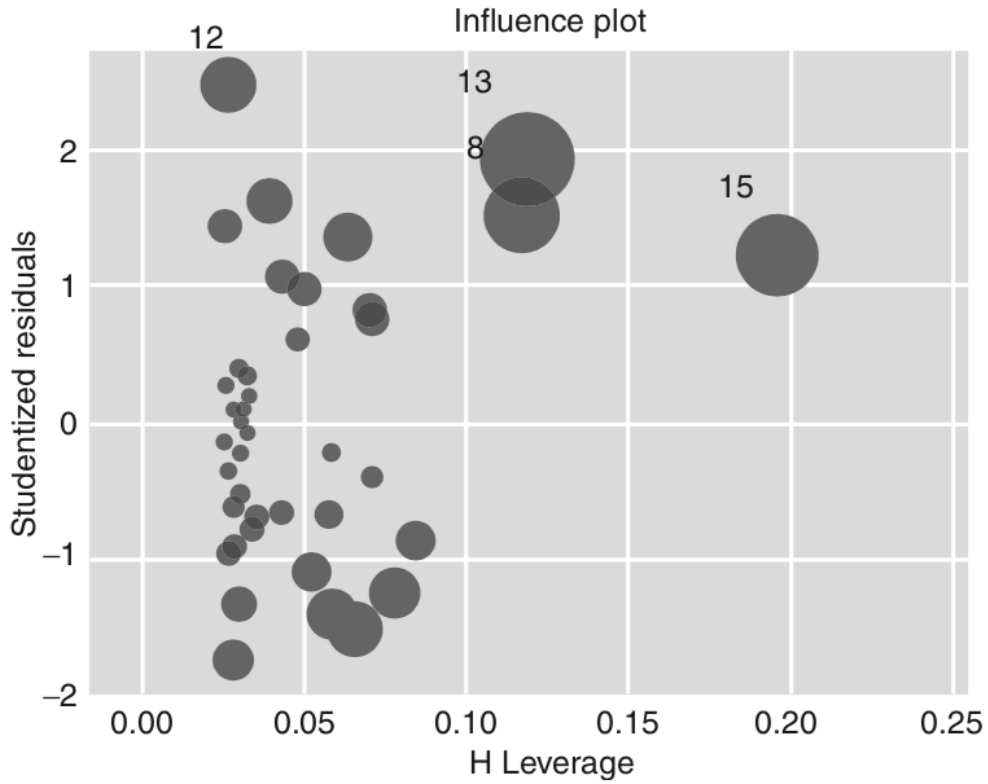

FIGURE 4.4 Leverage value versus residuals.

- The size of the circle is proportional to the product of residual and leverage value.
- The larger the circle, the larger is the residual and hence influence of the observation.

Model Diagnostics (Cntd.)

7. Making Prediction and Measuring Accuracy

7.1 Predicting using the validation set

```
pred_y = mba_salary_lm.predict( test_X )
```

7.2 Finding R-squared and RMSE

```
from sklearn.metrics import r2_score, mean_squared_error
```

```
np.abs(r2_score(test_y, pred_y))
```

```
0.15664584974230378
```

So, the model only explains 15.6% of the variance in the validation set.

```
import numpy
```

```
np.sqrt(mean_squared_error(test_y, pred_y))
```

```
73458.043483468937
```

Model Diagnostics (Cntd.)

7. Making Prediction and Measuring Accuracy

7.3 Calculating prediction intervals

```
from statsmodels.sandbox.regression.predstd import wls_prediction_std

# Predict the y values
pred_y = mba_salary_lm.predict( test_X )
```

```
# Predict the low and high interval values for y
_, pred_y_low, pred_y_high = wls_prediction_std( mba_salary_lm,
                    test_X,
                    alpha = 0.1)

# Store all the values in a dataframe
pred_y_df = pd.DataFrame( { 'grade_10_perc': test_X['Percentage in
Grade 10'],
                            'pred_y': pred_y,
                            'pred_y_left': pred_y_low,
                            'pred_y_right': pred_y_high } )
```

Model Diagnostics (Cntd.)

7. Making Prediction and Measuring Accuracy

7.3 Calculating prediction intervals

```
pred_y_df[0:10]
```

	grade_10_perc	pred_y	pred_y_left	pred_y_right
6	70.0	279828.402452	158379.832044	401276.972860
36	68.0	272707.227686	151576.715020	393837.740352
37	52.0	215737.829560	92950.942395	338524.716726
28	58.0	237101.353858	115806.869618	358395.838097
43	74.5	295851.045675	173266.083342	418436.008008
49	60.8	247070.998530	126117.560983	368024.436076
5	55.0	226419.591709	104507.444388	348331.739030
33	78.0	308313.101515	184450.060488	432176.142542
20	63.0	254904.290772	134057.999258	375750.582286
42	74.4	295494.986937	172941.528691	418048.445182

Multiple Linear Regression (MLR)

- Supervised learning algorithm
- Used for finding the existence of an association relationship between a dependent variable (response variable) and several independent variables (features).
- The functional form of MLR is given by

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_k X_{ki} + \varepsilon_i$$

- The regression coefficients are called partial regression coefficients
- The relationship between a feature and the response variable is calculated after removing the effect of all the other features in the model.

Multiple Linear Regression (MLR) (Cntd.)

- The assumptions of MLR model
  1. The regression model is linear in regression parameters.
  2. The residuals follow a normal distribution and the expected value of the residuals is zero.
  3. In time series data, residuals are assumed to be uncorrelated.
  4. It follows homoscedasticity, i.e. the variance of the residuals is constant for all values of X.
  5. There is no high correlation between independent variables in the model (called multi-collinearity). As it can lead to destabilization of the model and can result in an incorrect estimation of the regression parameters.

Multiple Linear Regression (MLR) (Cntd.)

- Predicting the SOLD PRICE (Auction Price) of players in Indian Premier League (IPL)

TABLE 4.3 Metadata of IPL dataset

Data Code	Description
AGE	Age of the player at the time of auction classified into three categories. Category 1 (L25) means the player is less than 25 years old, category 2 means that the age is between 25 and 35 years (B25– 35) and category 3 means that the age is more than 35 (A35).
RUNS-S	Number of runs scored by a player.
RUNS-C	Number of runs conceded by a player.
HS	Highest score by a batsman in IPL.
AVE-B	Average runs scored by a batsman in IPL.
AVE-BL	Bowling average (number of runs conceded/number of wickets taken) in IPL.
SR-B	Batting strike rate (ratio of the number of runs scored to the number of balls faced) in IPL.
SR-BL	Bowling strike rate (ratio of the number of balls bowled to the number of wickets taken) in IPL.
SIXERS	Number of six runs scored by a player in IPL.
WKTS	Number of wickets taken by a player in IPL.

Multiple Linear Regression (MLR) (Cntd.)

ECON	Economy rate of a bowler (number of runs conceded by the bowler per over) in IPL.
CAPTAINCY EXP	Captained either a T20 team or a national team.
ODI-SR-B	Batting strike rate in One-Day Internationals.
ODI-SR-BL	Bowling strike rate in One-Day Internationals.
ODI-RUNS-S	Runs scored in One-Day Internationals.
ODI-WKTS	Wickets taken in One-Day Internationals.
T-RUNS-S	Runs scored in Test matches.
T-WKTS	Wickets taken in Test matches.
PLAYER-SKILL	Player's primary skill (batsman, bowler, or allrounder).
COUNTRY	Country of origin of the player (AUS: Australia; IND: India; PAK: Pakistan; SA: South Africa; SL: Sri Lanka; NZ: New Zealand; WI: West Indies; OTH: Other countries).
YEAR-A	Year of Auction in IPL.
IPL TEAM	Team(s) for which the player had played in the IPL (CSK: Chennai Super Kings; DC: Deccan Chargers; DD: Delhi Dare-devils; KXI: Kings XI Punjab; KKR: Kolkata Knight Riders; MI: Mumbai Indians; PWI: Pune Warriors India; RR: Rajasthan Royals; RCB: Royal Challengers Bangalore). A + sign is used to indicate that the player has played for more than one team. For example, CSK+ would mean that the player has played for CSK as well as for one or more other teams.

Multiple Linear Regression (MLR) (Cntd.)

- Developing MLR model using Python
  - Loading the dataset and displaying first five records

```
ipl_auction_df = pd.read_csv( 'IPL IMB381IPL2013.csv' )
```

```
ipl_auction_df.iloc[0:5, 0:10]
```

	Sl. NO.	PLAYER NAME	AGE	COUNTRY	TEAM	PLAYING ROLE	T-RUNS	T-WKTS	ODI-RUNS-S	ODI-SR-B
0	1	Abdulla, YA	2	SA	KXIP	Allrounder	0	0	0	0.00
1	2	Abdur Razzak	2	BAN	RCB	Bowler	214	18	657	71.41
2	3	Agarkar, AB	2	IND	KKR	Bowler	571	58	1269	80.62
3	4	Ashwin, R	1	IND	CSK	Bowler	284	31	241	84.56
4	5	Badrinath, S	2	IND	CSK	Batsman	63	0	79	45.93

Multiple Linear Regression (MLR) (Cntd.)

- Creating the list of features

```
X_features = ipl_auction_df.columns
```

```
X_features = ['AGE', 'COUNTRY', 'PLAYING ROLE',
              'T-RUNS', 'T-WKTS', 'ODI-RUNS-S', 'ODI-SR-B',
              'ODI-WKTS', 'ODI-SR-BL', 'CAPTAINCY EXP', 'RUNS-S',
              'HS', 'AVE', 'SR-B', 'SIXERS', 'RUNS-C', 'WKTS',
              'AVE-BL', 'ECON', 'SR-BL']
```

- Categorical variables cannot directly be included in the regression model, and they must be encoded using dummy variables before incorporating in the model building.

Multiple Linear Regression (MLR) (Cntd.)

- Encoding Categorical Features
  1. If a categorical variable has n categories then we will need n-1 dummy variables.

```
ipl_auction_df['PLAYING ROLE'].unique()
```

```
array(['Allrounder', 'Bowler', 'Batsman', 'W. Keeper'], dtype=object)
```

```
pd.get_dummies(ipl_auction_df['PLAYING ROLE'])[0:5]
```

	Allrounder	Batsman	Bowler	W. Keeper
0	1.0	0.0	0.0	0.0
1	0.0	0.0	1.0	0.0
2	0.0	0.0	1.0	0.0
3	0.0	0.0	1.0	0.0
4	0.0	1.0	0.0	0.0

Multiple Linear Regression (MLR) (Cntd.)

- Creating dummy variables for all categorical variables in the dataset

```
categorical_features = ['AGE', 'COUNTRY', 'PLAYING ROLE',
                        'CAPTAINCY EXP']
```

```
ipl_auction_encoded_df = pd.get_dummies(ipl_auction_df[X_features],
                         columns = categorical_features,
                         drop_first = True)
```

```
ipl_auction_encoded_df.columns
```

```
Index(['T-RUNS', 'T-WKTS', 'ODI-RUNS-S', 'ODI-SR-B', 'ODI-WKTS',
       'ODI-SR-BL', 'RUNS-S', 'HS', 'AVE', 'SR-B', 'SIXERS',
       'RUNS-C', 'WKTS', 'AVE-BL', 'ECON', 'SR-BL', 'AGE_2',
       'AGE_3', 'COUNTRY_BAN', 'COUNTRY_ENG', 'COUNTRY_IND',
       'COUNTRY_NZ', 'COUNTRY_PAK', 'COUNTRY_SA', 'COUNTRY_SL',
       'COUNTRY_WI', 'COUNTRY_ZIM', 'PLAYING ROLE_Batsman',
       'PLAYING ROLE_Bowler', 'PLAYING ROLE_W. Keeper',
       'CAPTAINCY EXP_1'], dtype='object')
```

```
X_features = ipl_auction_encoded_df.columns
```

Multiple Linear Regression (MLR) (Cntd.)

- Splitting the Dataset into Train and Validation Sets

```
X = sm.add_constant( ipl_auction_encoded_df )
Y = ipl_auction_df['SOLD PRICE']

train_X, test_X, train_y, test_y = train_test_split(X ,
                                                    Y,
                                                train_size = 0.8,
                                                random_state = 42 )
```

Multiple Linear Regression (MLR) (Cntd.)

- Building the Model on the Training Dataset

```
ipl_model_1 = sm.OLS(train_y, train_X).fit()
ipl_model_1.summary2()
```

TABLE 4.4 Model summary for *ipl_model_1*

Model:	OLS	Adj. R-squared:	0.362
Dependent Variable:	SOLD PRICE	AIC:	2965.2841
Date:	2018-04-08 07:27	BIC:	3049.9046
No. Observations:	104	Log-Likelihood:	–1450.6
Df Model:	31	F-statistic:	2.883
Df Residuals:	72	Prob (F-statistic):	0.000114
R-squared:	0.554	Scale:	1.1034e+11

Multiple Linear Regression (MLR) (Cntd.)

	Coef.	Std.Err.	t	P > |t|	[0.025	0.975]
const	375827.1991	228849.9306	1.6422	0.1049	–80376.7996	832031.1978
T-RUNS	–53.7890	32.7172	–1.6441	0.1045	–119.0096	11.4316
T-WKTS	–132.5967	609.7525	–0.2175	0.8285	–1348.1162	1082.9228
ODI-RUNS-S	57.9600	31.5071	1.8396	0.0700	–4.8482	120.7681
ODI-SR-B	–524.1450	1576.6368	–0.3324	0.7405	–3667.1130	2618.8231
ODI-WKTS	815.3944	832.3883	0.9796	0.3306	–843.9413	2474.7301
ODI-SR-BL	–773.3092	1536.3334	–0.5033	0.6163	–3835.9338	2289.3154
RUNS-S	114.7205	173.3088	0.6619	0.5101	–230.7643	460.2054
HS	–5516.3354	2586.3277	–2.1329	0.0363	–10672.0855	–360.5853
AVE	21560.2760	7774.2419	2.7733	0.0071	6062.6080	37057.9439

	Coef.	Std.Err.	t	P > |t|	[0.025	0.975]
SR-B	–1324.7218	1373.1303	–0.9647	0.3379	–4062.0071	1412.5635
SIXERS	4264.1001	4089.6000	1.0427	0.3006	–3888.3685	12416.5687
RUNS-C	69.8250	297.6697	0.2346	0.8152	–523.5687	663.2187
WKTS	3075.2422	7262.4452	0.4234	0.6732	–11402.1778	17552.6622
AVE-BL	5182.9335	10230.1581	0.5066	0.6140	–15210.5140	25576.3810
ECON	–6820.7781	13109.3693	–0.5203	0.6045	–32953.8282	19312.2721
SR-BL	–7658.8094	14041.8735	–0.5454	0.5871	–35650.7726	20333.1539
AGE_2	–230767.6463	114117.2005	–2.0222	0.0469	–458256.1279	–3279.1648
AGE_3	–216827.0808	152246.6232	–1.4242	0.1587	–520325.1772	86671.0155
COUNTRY_BAN	–122103.5196	438719.2796	–0.2783	0.7816	–996674.4194	752467.3801
COUNTRY_ENG	672410.7654	238386.2220	2.8207	0.0062	197196.5172	1147625.0135
COUNTRY_IND	155306.4011	126316.3449	1.2295	0.2229	–96500.6302	407113.4325
COUNTRY_NZ	194218.9120	173491.9293	1.1195	0.2667	–151630.9280	540068.7521
COUNTRY_PAK	75921.7670	193463.5545	0.3924	0.6959	–309740.7804	461584.3143
COUNTRY_SA	64283.3894	144587.6773	0.4446	0.6579	–223946.8775	352513.6563
COUNTRY_SL	17360.1530	176333.7497	0.0985	0.9218	–334154.7526	368875.0586
COUNTRY_WI	10607.7792	230686.7892	0.0460	0.9635	–449257.9303	470473.4887
COUNTRY_ZIM	–145494.4793	401505.2815	–0.3624	0.7181	–945880.6296	654891.6710
PLAYING ROLE_Batsman	75724.7643	150250.0240	0.5040	0.6158	–223793.1844	375242.7130
PLAYING ROLE_Bowler	15395.8752	126308.1272	0.1219	0.9033	–236394.7744	267186.5249
PLAYING ROLE_W. Keeper	–71358.6280	213585.7444	–0.3341	0.7393	–497134.0278	354416.7718
CAPTAINCY EXP_1	164113.3972	123430.6353	1.3296	0.1878	–81941.0772	410167.8716

Multiple Linear Regression (MLR) (Cntd.)

- As per the p-value (<0.05), only the features HS, AGE_2, AVE and COUNTRY_ENG have come out significant
- None of the other features are influencing SOLD PRICE
- This is not very intuitive and could be a result of multi-collinearity effect of the variables.

Omnibus:	0.891	Durbin-Watson:	2.244
Prob(Omnibus):	0.640	Jarque-Bera (JB):	0.638
Skew:	0.190	Prob(JB):	0.727
Kurtosis:	3.059	Condition No.:	84116

Multiple Linear Regression (MLR) (Cntd.)

Multi-Collinearity and Handling Multi-Collinearity

- Impact of multi-collinearity
  1. The standard error of estimate is inflated. It results in an underestimation of t-statistic value.
  2. A statistically significant explanatory variable may be labelled as statistically insignificant due to the large p-value.
  3. The sign of the regression coefficient may be different.
  4. Adding/ removing a variable or even an observation may result in large variation in regression coefficient estimates.

Multiple Linear Regression (MLR) (Cntd.)

Multi-Collinearity and Handling Multi-Collinearity

- Variance Inflation Factor (VIF)
  1. A measure used for identifying the existence of multi-collinearity.
  2. Let R be the R-squared value of this model. Then the VIF, which is a measure of multi-collinearity, is given by

$$\mathrm{VIF} = \frac{1}{1 - R_{12}^2}$$

  3. $\sqrt{VIF}$ is the value by which t-statistic value is deflated.
  4. VIF value of greater than 4 requires further investigation to assess the impact of multi-collinearity.

Multiple Linear Regression (MLR) (Cntd.)

Multi-Collinearity and Handling Multi-Collinearity

- Calculating VIF for the features

```
from statsmodels.stats.outliers_influence import variance_inflation_
factor

def get_vif_factors( X ):
        X_matrix = X.as_matrix()
        vif = [ variance_inflation_factor( X_matrix, i ) for i in range
              ( X_matrix.sh ape[1] ) ]
        vif_factors = pd.DataFrame()
        vif_factors['column'] = X.columns
        vif_factors['VIF'] = vif

        return vif_factors
```

Multiple Linear Regression (MLR) (Cntd.)

Multi-Collinearity and Handling Multi-Collinearity

```
vif_factors = get_vif_factors( X[X_features] )
vif_factors
```

	Column	VIF
1	T-WKTS	7.679284
2	ODI-RUNS-S	16.426209
3	ODI-SR-B	13.829376
4	ODI-WKTS	9.951800
5	ODI-SR-BL	4.426818
6	RUNS-S	16.135407
7	HS	22.781017
8	AVE	25.226566
9	SR-B	21.576204
10	SIXERS	9.547268
11	RUNS-C	38.229691
12	WKTS	33.366067
13	AVE-BL	100.198105
14	ECON	7.650140
15	SR-BL	103.723846

	Column	VIF
16	AGE_2	6.996226
17	AGE_3	3.855003
18	COUNTRY_BAN	1.469017
19	COUNTRY_ENG	1.391524
20	COUNTRY_IND	4.568898
21	COUNTRY_NZ	1.497856
22	COUNTRY_PAK	1.796355
23	COUNTRY_SA	1.886555
24	COUNTRY_SL	1.984902
25	COUNTRY_WI	1.531847
26	COUNTRY_ZIM	1.312168
27	PLAYING ROLE_Batsman	4.843136
28	PLAYING ROLE_Bowler	3.795864
29	PLAYING ROLE_W. Keeper	3.132044
30	CAPTAINCY EXP_1	4.245128

Multiple Linear Regression (MLR) (Cntd.)

Multi-Collinearity and Handling Multi-Collinearity

- Checking correlation of columns with large VIFs

```
columns_with_large_vif = vif_factors[vif_factors.vif > 4].column
```

```
plt.figure( figsize = (12,10) )
sn.heatmap( X[columns_with_large_vif].corr(), annot = True );
plt.title("Figure 4.5 - Heatmap depicting correlation between
features");
```

Multiple Linear Regression (MLR) (Cntd.)

Multi-Collinearity and Handling Multi-Collinearity

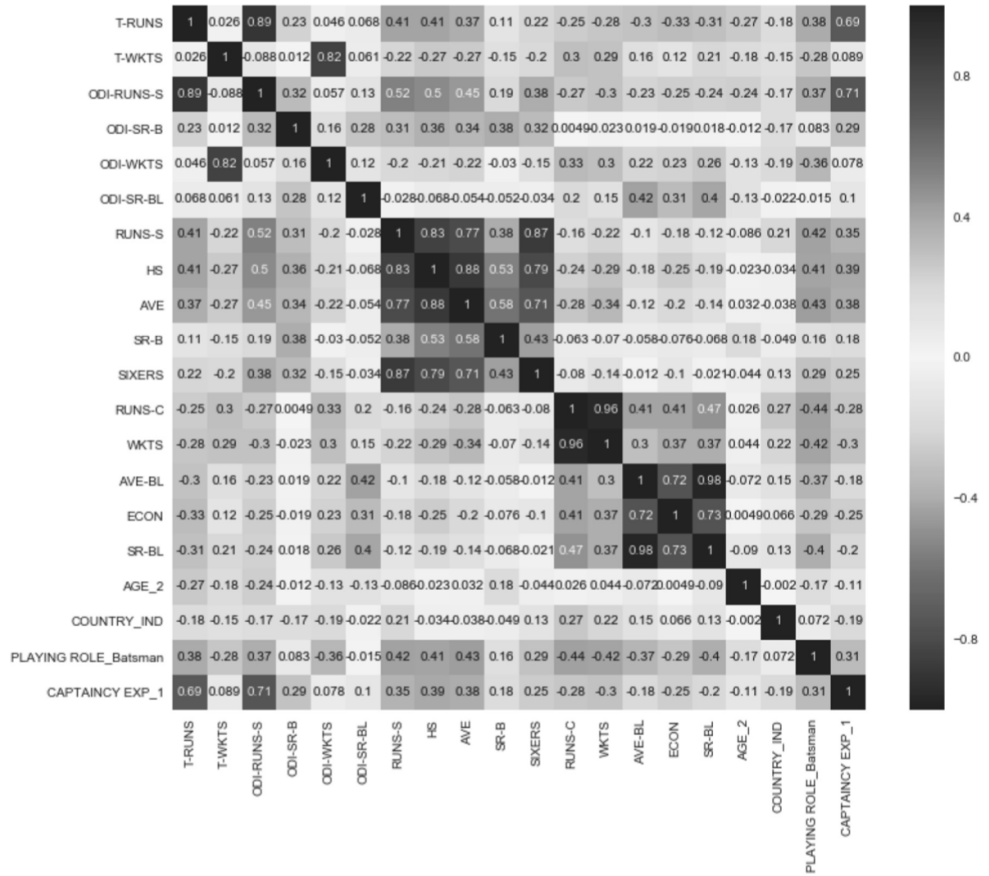

FIGURE 4.5 Heatmap depicting the correlation between features.

1. T-RUNS and ODI-RUNS-S are highly correlated
2. ODI-WKTS and T-WKTS are highly correlated
3. Batsman features like RUNS-S, HS, AVE, SIXERS are highly correlated.
4. Bowler's features like AVE-BL, ECON and SR-BL are highly correlated.

Note – To avoid multi-collinearity, we can keep only one column from each group of highly correlated variables and remove the others.

Multiple Linear Regression (MLR) (Cntd.)

Multi-Collinearity and Handling Multi-Collinearity

```
columns_to_be_removed = ['T-RUNS', 'T-WKTS', 'RUNS-S', 'HS', 'AVE',
                         'RUNS-C', 'SR-B', 'AVE-BL', 'ECON',
                         'ODI-SR-B', 'ODI-RUNS-S', 'AGE_2', 'SR-BL' ]
```

```
X_new_features = list( set(X_features) - set(columns_to_be_removed))
```

```
get_vif_factors( X[X_new_features] )
```

Multiple Linear Regression (MLR) (Cntd.)

Multi-Collinearity and Handling Multi-Collinearity

	Column	VIF
0	AGE_3	1.779861
1	ODI-SR-BL	2.822148
2	COUNTRY_IND	3.144668
3	COUNTRY_ENG	1.131869
4	COUNTRY_NZ	1.173418
5	COUNTRY_PAK	1.334773
6	COUNTRY_WI	1.194093
7	COUNTRY_SL	1.519752
8	COUNTRY_ZIM	1.205305
9	CAPTAINCY EXP_1	2.458745
10	PLAYING ROLE_W. Keeper	1.900941
11	PLAYING ROLE_Bowler	3.060168
12	SIXERS	2.397409
13	COUNTRY_BAN	1.094293
14	COUNTRY_SA	1.416657
15	PLAYING ROLE_Batsman	2.680207
16	ODI-WKTS	2.742889
17	WKTS	2.883101

Multiple Linear Regression (MLR) (Cntd.)

Multi-Collinearity and Handling Multi-Collinearity

	Column	VIF
0	AGE_3	1.779861
1	ODI-SR-BL	2.822148
2	COUNTRY_IND	3.144668
3	COUNTRY_ENG	1.131869
4	COUNTRY_NZ	1.173418
5	COUNTRY_PAK	1.334773
6	COUNTRY_WI	1.194093
7	COUNTRY_SL	1.519752
8	COUNTRY_ZIM	1.205305
9	CAPTAINCY EXP_1	2.458745
10	PLAYING ROLE_W. Keeper	1.900941
11	PLAYING ROLE_Bowler	3.060168
12	SIXERS	2.397409
13	COUNTRY_BAN	1.094293
14	COUNTRY_SA	1.416657
15	PLAYING ROLE_Batsman	2.680207
16	ODI-WKTS	2.742889
17	WKTS	2.883101

Multiple Linear Regression (MLR) (Cntd.)

Building a new model after removing Multi-Collinearity

```
train_X = train_X[X_new_features]

ipl_model_2 = sm.OLS(train_y, train_X).fit()
ipl_model_2.summary2()
```

TABLE 4.5 Model summary for *ipl_model_2*

Model:	OLS	Adj. R-squared:	0.728
Dependent Variable:	SOLD PRICE	AIC:	2965.1080
Date:	2018-04-08 07:27	BIC:	3012.7070
No. Observations:	104	Log-Likelihood:	−1464.6
Df Model:	18	F-statistic:	16.49
Df Residuals:	86	Prob (F-statistic):	1.13e-20
R-squared:	0.775	Scale:	1.2071e+11

Multiple Linear Regression (MLR) (Cntd.)

Building a new model after removing Multi-Collinearity

	Coef.	Std.Err.	t	P > |t|	[0.025	0.975]
COUNTRY_IND	282829.8091	96188.0292	2.9404	0.0042	91614.3356	474045.2827
COUNTRY_BAN	–108758.6040	369274.1916	–0.2945	0.7691	–842851.4010	625334.1930
AGE_3	–8950.6659	98041.9325	–0.0913	0.9275	–203851.5772	185950.2453
COUNTRY_PAK	122810.2480	159600.8063	0.7695	0.4437	–194465.6541	440086.1502
COUNTRY_WI	–22234.9315	213050.5847	–0.1044	0.9171	–445765.4766	401295.6135
ODI-WKTS	772.4088	470.6354	1.6412	0.1044	–163.1834	1708.0009
COUNTRY_SA	108735.9086	115092.9596	0.9448	0.3474	–120061.3227	337533.1399
COUNTRY_ENG	682934.7166	216150.8279	3.1595	0.0022	253241.0920	1112628.3411
CAPTAINCY EXP_1	208376.6957	98128.0284	2.1235	0.0366	13304.6315	403448.7600
WKTS	2431.8988	2105.3524	1.1551	0.2512	–1753.4033	6617.2008
SIXERS	7862.1259	2086.6101	3.7679	0.0003	3714.0824	12010.1694
PLAYING ROLE_W. Keeper	–55121.9240	169922.5271	–0.3244	0.7464	–392916.7280	282672.8801
COUNTRY_ZIM	–67977.6781	390859.9289	–0.1739	0.8623	–844981.5006	709026.1444
PLAYING ROLE_Bowler	–18315.4968	106035.9664	–0.1727	0.8633	–229108.0215	192477.0279
COUNTRY_SL	55912.3398	142277.1829	0.3930	0.6953	–226925.3388	338750.0184
COUNTRY_NZ	142968.8843	151841.7382	0.9416	0.3491	–158882.5009	444820.2695
PLAYING ROLE_Batsman	121382.0570	106685.0356	1.1378	0.2584	–90700.7746	333464.8886
ODI-SR-BL	909.0021	1267.4969	0.7172	0.4752	–1610.6983	3428.7026

Omnibus:	8.635	Durbin-Watson:	2.252
Prob(Omnibus):	0.013	Jarque-Bera (JB):	8.345
Skew:	0.623	Prob(JB):	0.015
Kurtosis:	3.609	Condition No.:	1492

Multiple Linear Regression (MLR) (Cntd.)

Building a new model after removing Multi-Collinearity

- In Table 4.5, base don the –values, only the variables COUNTRY_IND, COUNTRY_ENG, SIXERS, CAPTAINCYEXP_1 have come out statistically significant.
- Features that decide the SOLD PRICE are
  1. Whether the players belong to India or England.
  2. How many sixes has the player hit in previous versions of the IPL? How many wickets have been taken by the player in ODIs?
  3. Whether the player has any previous captaincy experience or not.

Multiple Linear Regression (MLR) (Cntd.)

Building a new model after removing Multi-Collinearity

- Create a new list of significant variables and build a new model

```
significant_vars = ['COUNTRY_IND', 'COUNTRY_ENG', 'SIXERS',
'CAPTAINCY EXP_1']

train_X = train_X[significant_vars]

ipl_model_3 = sm.OLS(train_y, train_X).fit()
ipl_model_3.summary2()
```

TABLE 4.6 Model summary for *ipl_model_3*

Model:	OLS	Adj. R-squared:	0.704
Dependent Variable:	SOLD PRICE	AIC:	2961.8089
Date:	2018-04-08 07:27	BIC:	2972.3864
No. Observations:	104	Log-Likelihood:	–1476.9
Df Model:	4	F-statistic:	62.77
Df Residuals:	100	Prob (F-statistic):	1.97e-26
R-squared:	0.715	Scale:	1.3164e+11

Multiple Linear Regression (MLR) (Cntd.)

Building a new model after removing Multi-Collinearity

- Create a new list of significant variables and build a new model

	Coef.	Std.Err.	*t*	*P* > |*t*|	[0.025	0.975]
COUNTRY_IND	387890.2538	63007.1511	6.1563	0.0000	262885.8606	512894.6471
COUNTRY_ENG	731833.6386	214164.4988	3.4172	0.0009	306937.3727	1156729.9045
SIXERS	8637.8344	1675.1313	5.1565	0.0000	5314.4216	11961.2472
CAPTAINCY EXP_1	359725.2741	74930.3460	4.8008	0.0000	211065.6018	508384.9463

Omnibus:	1.130	Durbin-Watson:	2.238
Prob(Omnibus):	0.568	Jarque-Bera (JB):	0.874
Skew:	0.223	Prob(JB):	0.646
Kurtosis:	3.046	Condition No.:	165

Multiple Linear Regression (MLR) (Cntd.)

Building a new model after removing Multi-Collinearity

- The following inference can be derived from the latest model ipl_model_3
  1. All the variables are statistically significant, as p-value is less than 0.05.
  2. The overall model is significant as the p-value for the F-statistic is also less than 0.05%.
  3. The model can explain 71.5% of the variance in SOLD PRICE as the R-squared value is 0.715 and the adjusted R-squared value is 0.704. Adjusted R-squared is a measure that is calculated after normalizing SSE and SST with the corresponding degrees of freedom.

Multiple Linear Regression (MLR) (Cntd.)

- Residual Analysis in Multiple Linear Regression
  1. Test for normality of residuals (P-P plot)

```
def draw_pp_plot( model, title ):
    probplot = sm.ProbPlot( model.resid );
    plt.figure( figsize = (8, 6) );
    probplot.ppplot( line='45' );
    plt.title( title );
    plt.show();
```

```
draw_pp_plot(ipl_model_3,
             "Figure 4.6 - Normal P-P Plot of Regression Standardized
             Residuals");
```

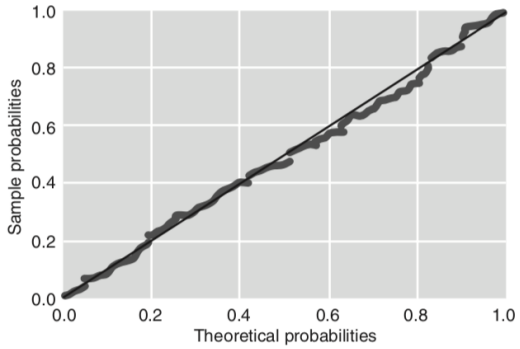

FIGURE 4.6 Normal P-P plot of regression standardized residuals for *ipl_model_3*.

Multiple Linear Regression (MLR) (Cntd.)

- Residual Analysis in Multiple Linear Regression
  1. Residual plot for homoscedasticity and model specification

```
def plot_resid_fitted(fitted, resid, title):
    plt.scatter( get_standardized_values( fitted ),
                 get_standardized_values( resid ) )
    plt.title(title)
    plt.xlabel("Standardized predicted values")
    plt.ylabel("Standardized residual values")
    plt.show()
```

```
plot_resid_fitted(ipl_model_3.fittedvalues,
                  ipl_model_3.resid,
                  "Figure 4.7 - Residual Plot")
```

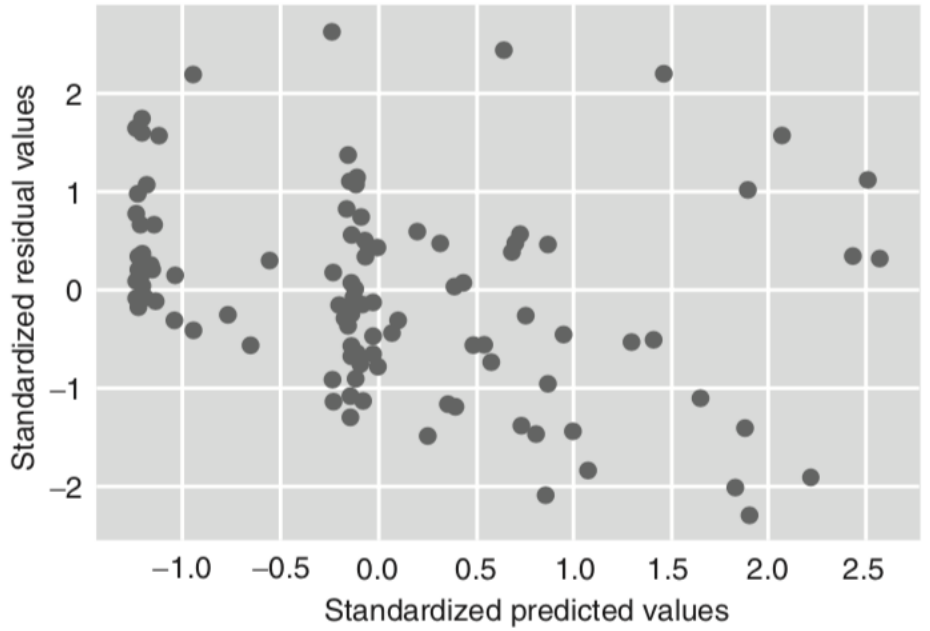

FIGURE 4.7 Residual plot.

Multiple Linear Regression (MLR) (Cntd.)

- Detecting Influencers

```
k = train_X.shape[1]
n = train_X.shape[0]
```

```
print("Number of variables:", k, " and number of observations:", n)
```

```
Number of variables: 4 and number of observations: 104
```

```
leverage_cutoff = 3*((k + 1)/n)
print( "Cutoff for leverage value:", round(leverage_cutoff, 3) )
```

```
Cutoff for leverage value: 0.144
```

- Observations with leverage value more than 0.144 are highly influential.

Multiple Linear Regression (MLR) (Cntd.)

- Detecting Influencers

```
from statsmodels.graphics.regressionplots import influence_plot
fig, ax = plt.subplots( figsize=(8,6) )
influence_plot(ipl_model_3, ax = ax )
plt.title( "Figure 4.8 - Leverage Value Vs Residuals" )
plt.show()
```

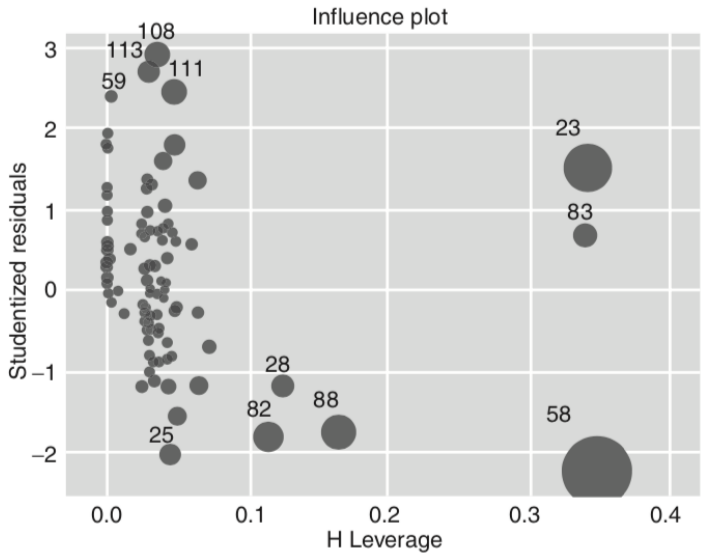

FIGURE 4.8 Leverage value versus residuals.

Multiple Linear Regression (MLR) (Cntd.)

- Transforming Response Variable
  - A process of deriving new dependent and/or independent variables to identify the correct functional form of the regression model.
  - Transformation in MLR is used to address the following issues:

1. Poor fit (low R-squared value)
2. Pattern in residual analysis indicating a potential non-linear relationship between the dependent and independent variables.
3. Residual do not follow a normal distribution.
4. Residuals are not homoscedastic.

Multiple Linear Regression (MLR) (Cntd.)

- Transforming Response Variable

```
train_y = np.sqrt(train_y)
```

```
ipl_model_4 = sm.OLS(train_y, train_X).fit()
ipl_model_4.summary2()
```

TABLE 4.7 Model summary for *ipl_model_4*

Model:	OLS	Adj. R-squared:	0.741
Dependent Variable:	SOLD PRICE	AIC:	1527.9999
Date:	2018-04-08 07:30	BIC:	1538.5775
No. Observations:	104	Log-Likelihood:	−760.00
Df Model:	4	F-statistic:	75.29
Df Residuals:	100	Prob (F-statistic):	2.63e-29
R-squared:	0.751	Scale:	1.3550e+05

Multiple Linear Regression (MLR) (Cntd.)

- Transforming Response Variable

	Coef.	Std.Err.	*t*	$P > \|t\|$	[0.025	0.975]
COUNTRY_IND	490.7089	63.9238	7.6765	0.0000	363.8860	617.5318
COUNTRY_ENG	563.0261	217.2801	2.5912	0.0110	131.9486	994.1036
SIXERS	8.5338	1.6995	5.0213	0.0000	5.1620	11.9055
CAPTAINCY EXP_1	417.7575	76.0204	5.4953	0.0000	266.9352	568.5799

Omnibus:	0.017	Durbin-Watson:	1.879
Prob(Omnibus):	0.992	Jarque-Bera (JB):	0.145
Skew:	0.005	Prob(JB):	0.930
Kurtosis:	2.817	Condition No.:	165

Multiple Linear Regression (MLR) (Cntd.)

- Transforming Response Variable

```
draw_pp_plot(ipl_model_4,
             "Figure 4.9 - Normal P-P Plot of Regression Standardized
             Residuals");
```

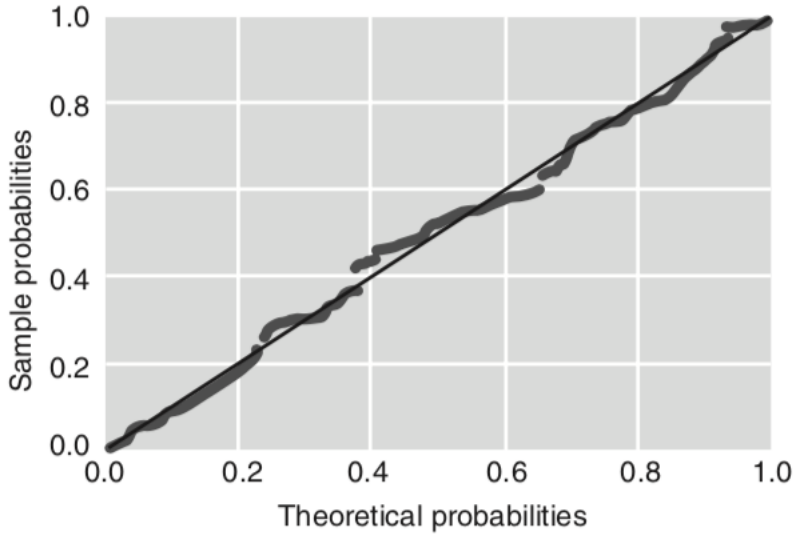

FIGURE 4.9 Normal P-P plot of regression standardized residuals.

Multiple Linear Regression (MLR) (Cntd.)

- Making Predictions on the Validation Set
  - As the model we have built predicts square root of the SOLD PRICE, we need to square the predicted values to get the actual SOLD PRICE of the players.

```
pred_y = np.power(ipl_model_4.predict(test_X[train_X.columns]),2)
```

  - Measuring RMSE

```
from sklearn import metrics
np.sqrt(metrics.mean_squared_error(pred_y, test_y))
```

```
496151.18122558104
```

  - Measuring R-squared value

```
np.round( metrics.r2_score(pred_y, test_y), 2 )
```

```
0.44
```

Multiple Linear Regression (MLR) (Cntd.)

- Auto-correlation between error terms
  1. If there is auto correlation, the standard error of estimate of the beta coefficients may be underestimated and that will result in over-estimation of the t-statistic value, in turn, will result in a low p-value.
  2. A variable which has no statistically significant relationship with the response variable may be accepted in the model due to the presence of auto-correlation.
  3. The presence of auto-correlation can be established using Durbin-Watson test.
  4. As a thumb rule, a Durbin-Watson statistic close 2 would imply the absence of autocorrelation.
  5. Auto-correlation is more relevant in the case of time-series data.

Thank You !