Linear Regression

By Dr Shaik A Qadeer

Professor MJCET

Simple linear regression(SLR)

- Regression works by establishing a relationship between variables in the data that represent characteristics—known as the *features*—of the thing being observed, and the variable we're trying to predict—known as the *label*.
- SLR is a statistical technique used for finding the existence of an association relationship between **a dependent variable** (response variable) **and an** independent variable (feature).

Multiple linear regression(MLR)

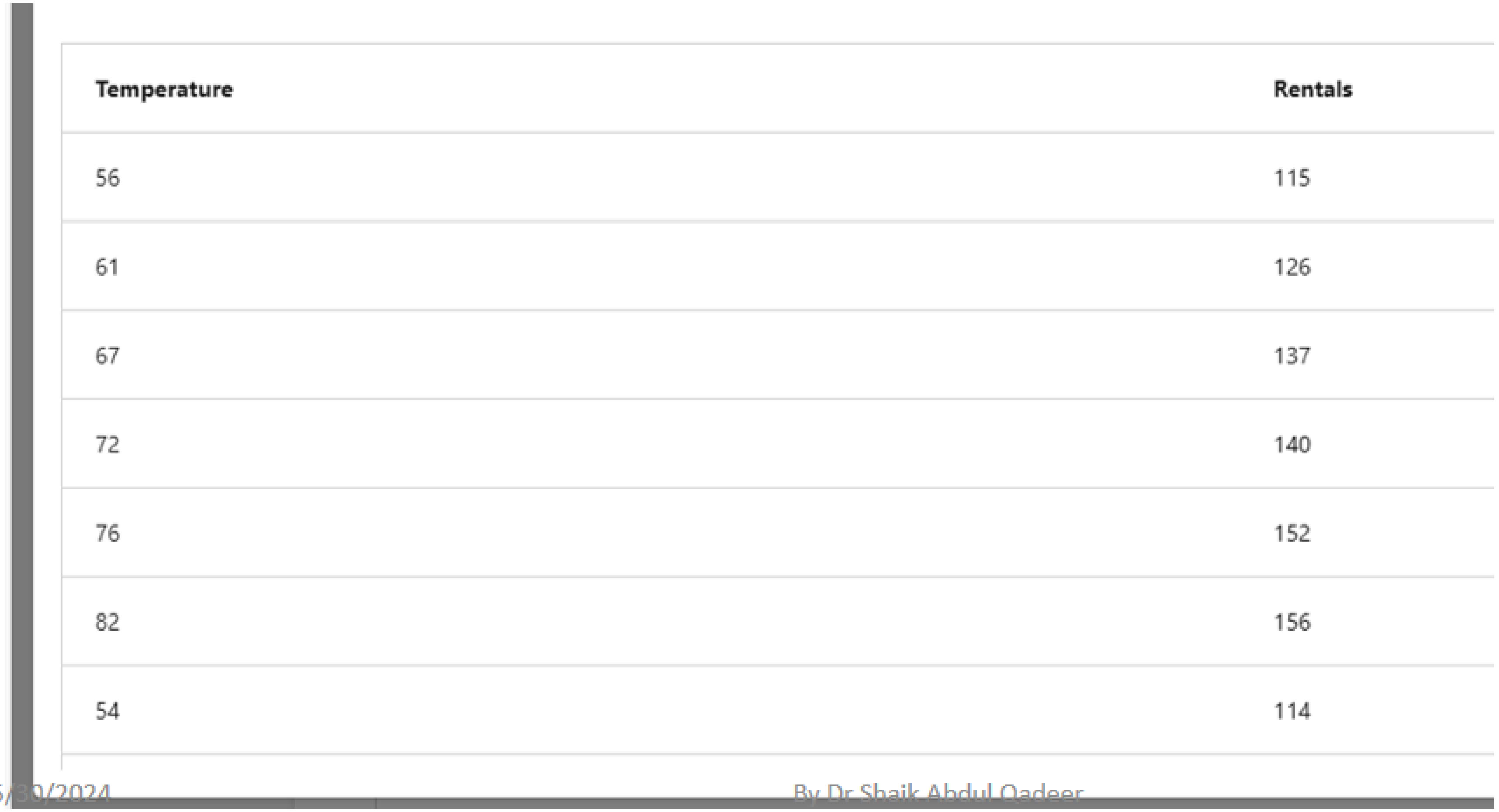
- SLR is a statistical technique used for finding the existence of an association relationship between **N dependent variables** (response variables) **and an** independent variable (feature).

Steps in building Linear 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

SLR in more detail with example

- Suppose we use a single feature—average daily temperature—to predict the bicycle rentals label.



Temperature	Rentals
56	115
61	126
67	137
72	140
76	152
82	156
54	114

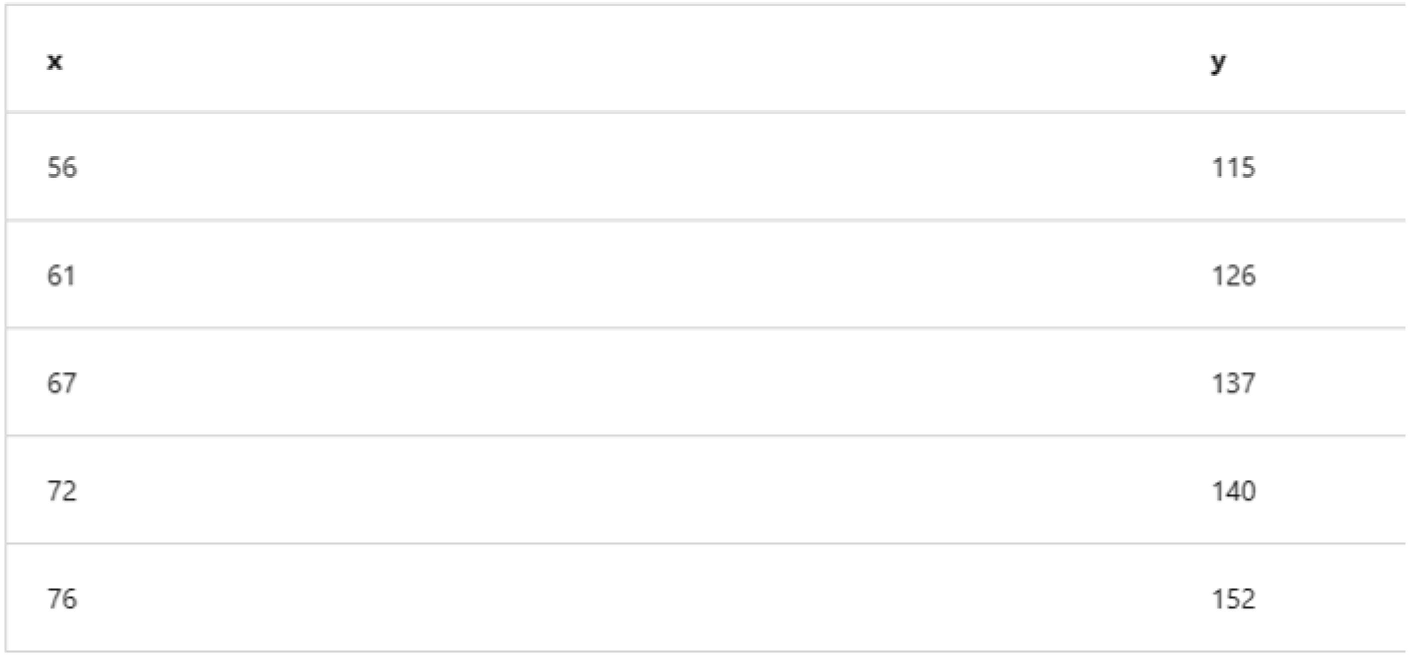
SLR in more detail with example..

- Let 5 random samples from this and training of model, then model behaves like this: **f(x) = y(take x and predict y)**

x	y
56	115
61	126
67	137
72	140
76	152

SLR in more detail with example..

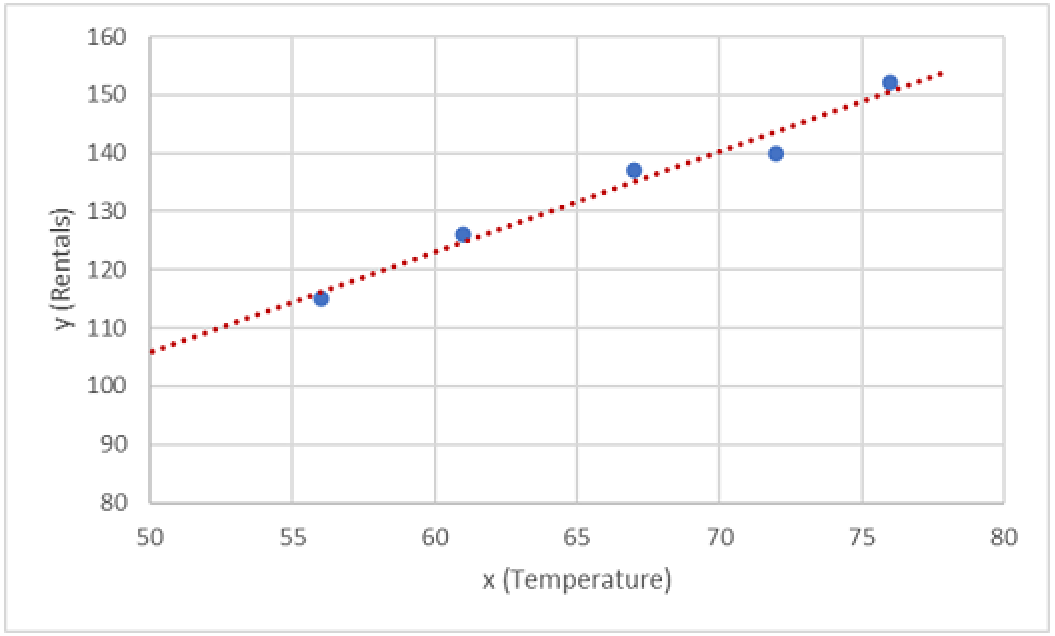
- Let 5 random samples from this and training of model, then model behaves like this: **f(x) = y(take x and predict y)**



x	y
56	115
61	126
67	137
72	140
76	152

SLR in more detail with example..

- Let's start by plotting the training values for ***x*** and ***y*** on a chart:



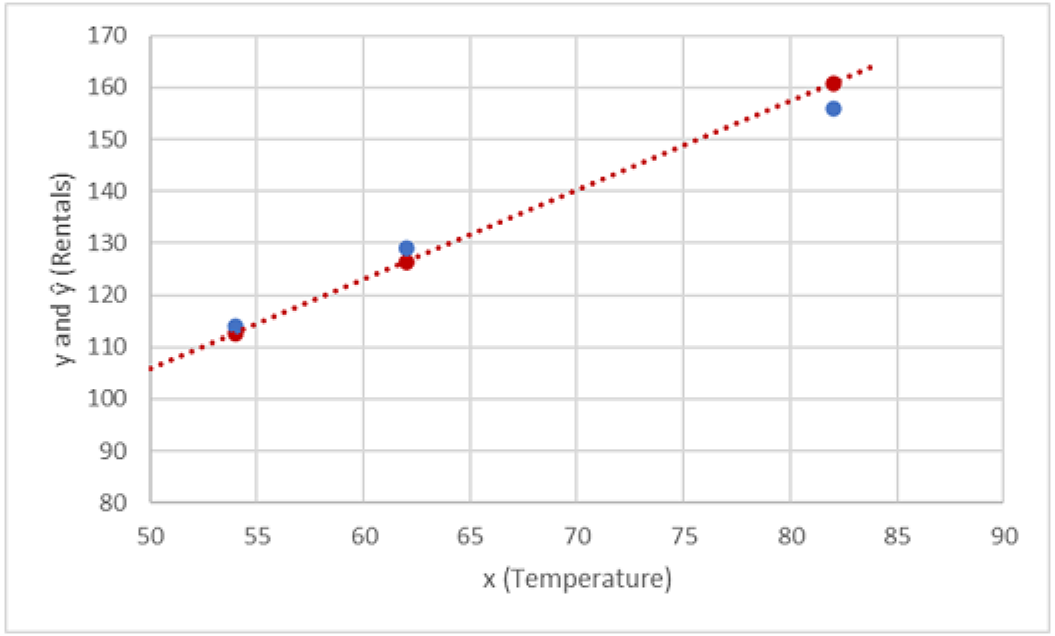
SLR in more detail with example..

- Evaluate this model by applying remaining 3 samples and calculate error:

x	y	ŷ
82	156	159.4
54	114	111.8
62	129	125.4

SLR in more detail with example..

- Lets see and predicted and actual outcome looks like:



SLR in more detail with example..

- There are various ways we can measure the variance between the predicted and actual values, and we can use these metrics to evaluate how well the model predicts. One of the metric is MSE

y	$\hat{y}$	$y - \hat{y}$	$(y - \hat{y})^2$
156	159.4	-3.4	11.56
114	111.8	2.2	4.84
129	125.4	3.6	12.96
	Sum	$\sum$	29.36
	Mean	$\bar{x}$	**9.79**

Therefore, the loss for our model based on the MSE metric is 9.79

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.. Steps:

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 Scikit-Learn library and fit the model using **LinearRegression** estimator.
6. Print model summary and conduct model diagnostics.

Example.. Steps:

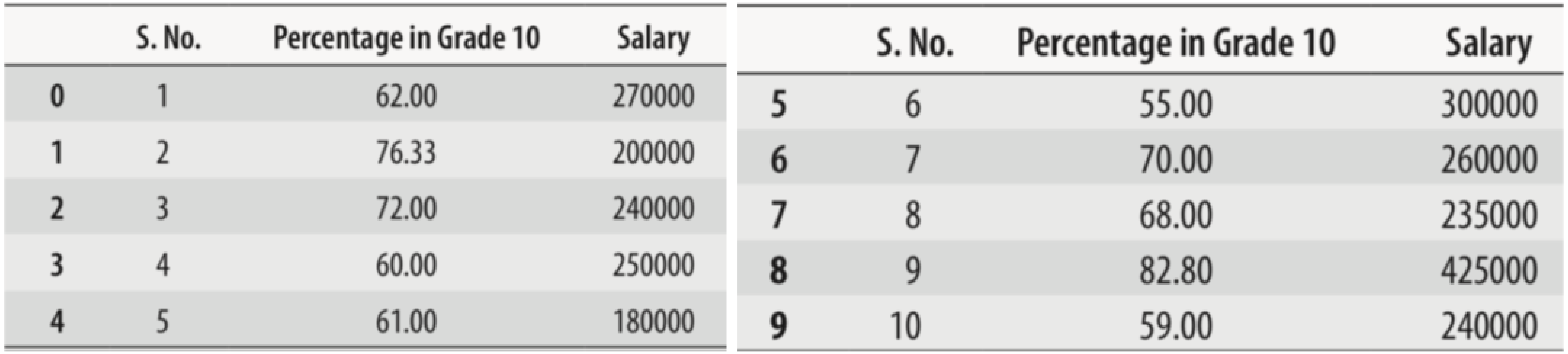
1. Import *pandas* and *numpy* libraries.
2. Use *read_csv* to load the dataset into DataFrame.
3. Identify the feture(s) (X) and outcome (Y) variable in the DataFrame for building the model.
4. Split the dataset into training and validation sets using *train_test_split().*
5. Import Scikit-Learn library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

Example.. 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)
```

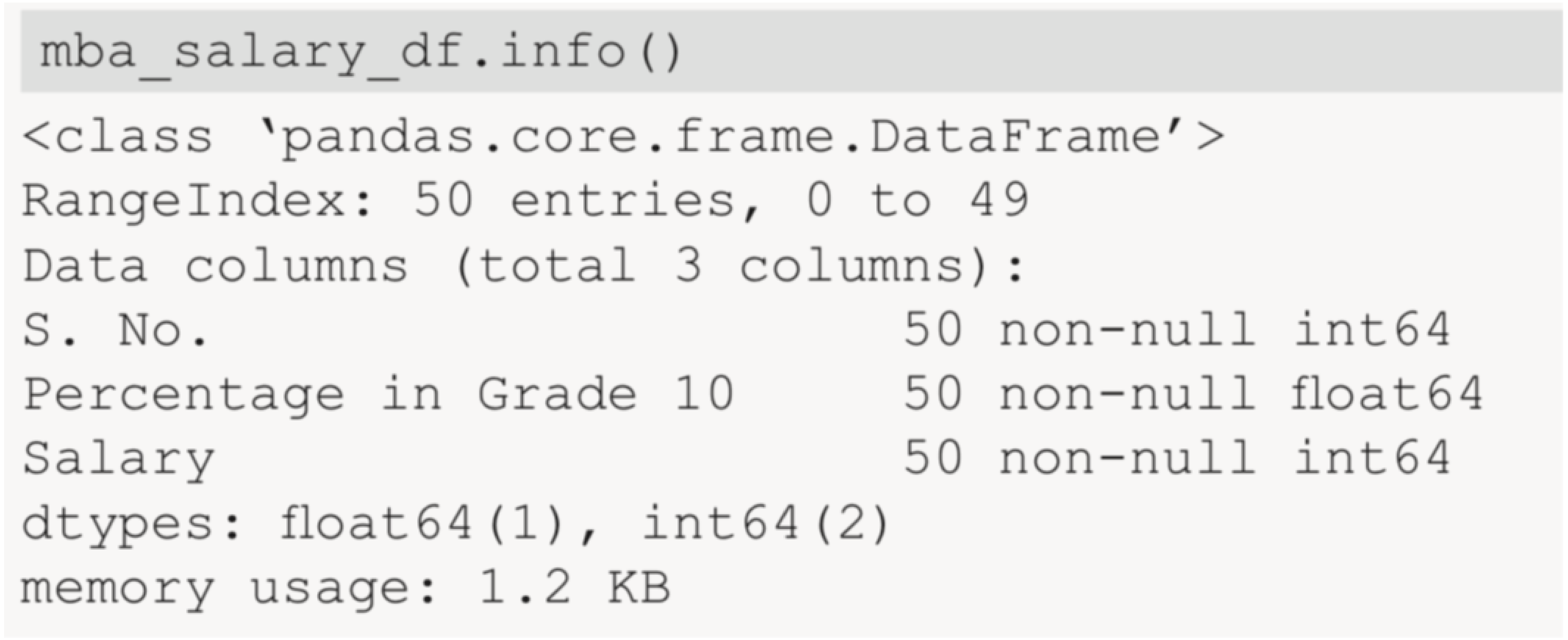
Example.. Loading data set:

```
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
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.. Printing the detail information of DF:



```
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.. Creating Feature X and label 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..

- 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.

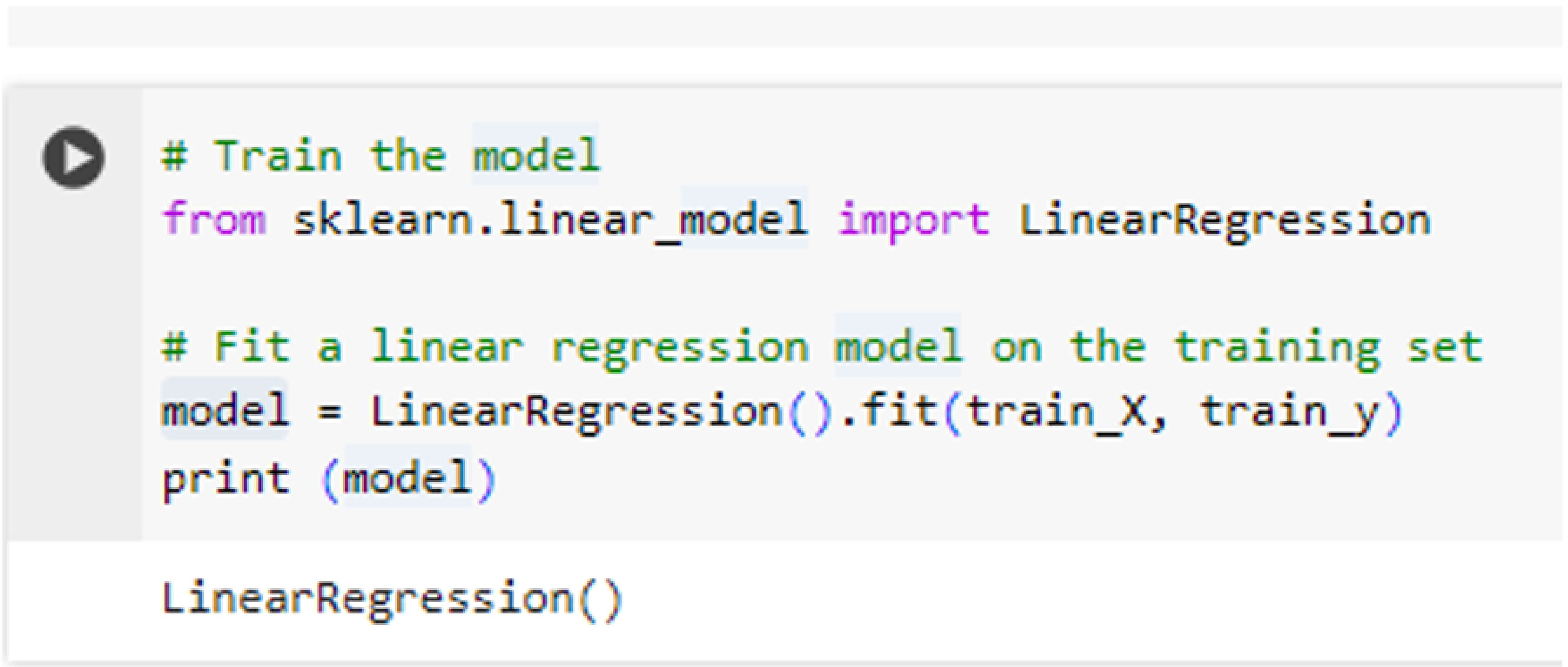
Splitting of data

```
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

Building the model



```
# Train the model
from sklearn.linear_model import LinearRegression

# Fit a linear regression model on the training set
model = LinearRegression().fit(train_X, train_y)
print (model)
```

```
LinearRegression()
```