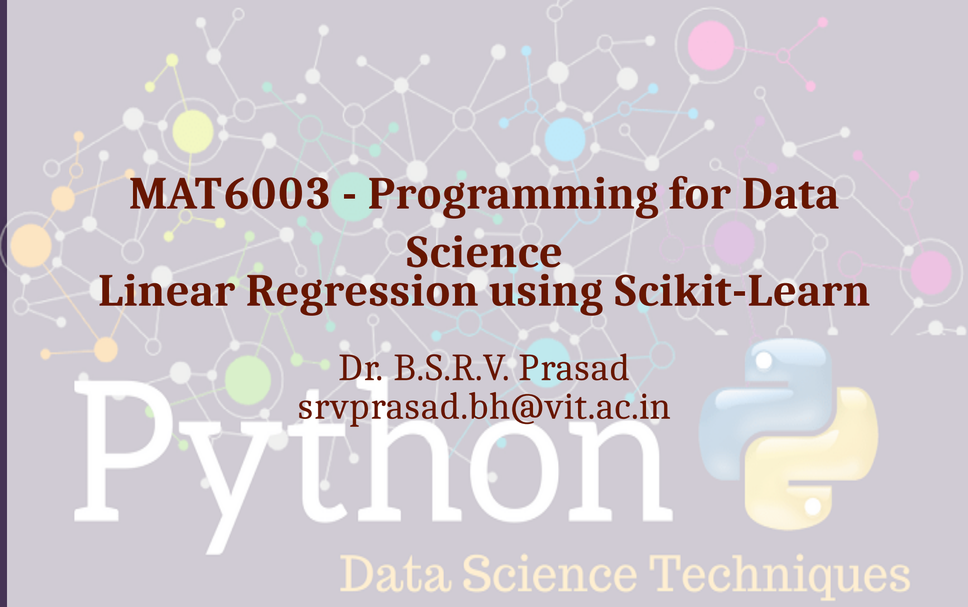
MAT6003 - Programming for Data Science
Linear Regression using Scikit-Learn

Dr. B.S.R.V. Prasad
srvprasad.bh@vit.ac.in



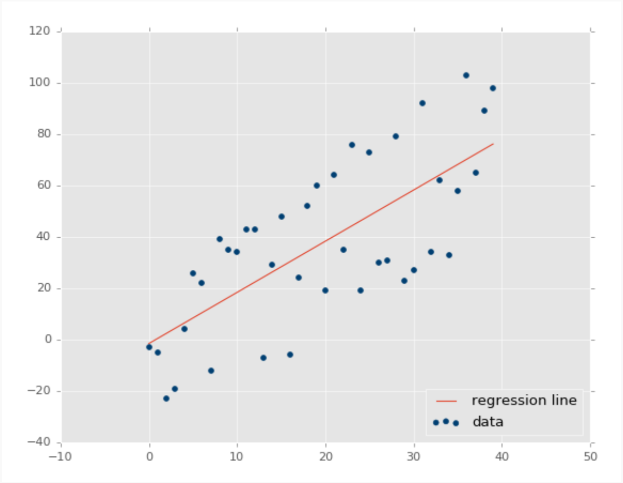
Linear Regression using Scikit-Learn

- In this tutorial, we study the most fundamental machine learning algorithms i.e., linear regression using Scikit-learn machine learning library.
- We discuss the Python implementation of both simple linear regression and multiple linear regression.

Linear Regression Theory - Review

- Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x) using a linear relationship between x and y.
- If we plot the independent variable (x) on the $x-$axis and dependent variable (y) on the $y-$axis, linear regression gives us a straight line that best fits the data points, as shown in the figure below.

Liner Regression Theory - Review



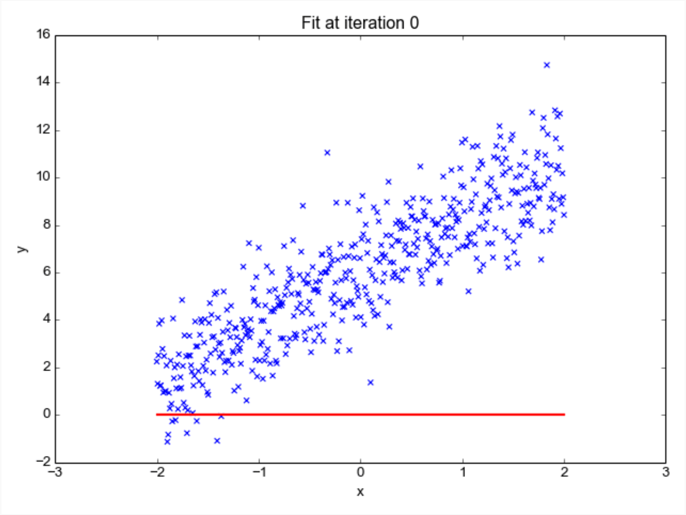
Linear Regression Theory - Review

The equation of the above line is :

$$y = mx + b$$

- *Here, b is the intercept and m is the slope of the line.*
- *Basically, the linear regression algorithm gives us the most optimal value for the intercept and the slope (in two dimensions).*
- *The y and x variables remain the same, since they are the data features and cannot be changed.*
- *The values that we can control are the intercept(b) and slope(m).*
- *There can be multiple straight lines depending upon the values of intercept and slope.*
- *Basically what the linear regression algorithm does is it fits multiple lines on the data points and returns the line that results in the least error.*

Liner Regression Theory - Review



Linear Regression Theory - Review

- This same concept can be extended to cases where there are more than two variables and is called multiple linear regression.
  - For instance, consider a scenario where you have to predict the price of the house based upon its area, number of bedrooms, the average income of the people in the area, the age of the house, and so on.
- In this case, the dependent variable (target variable) is dependent upon several independent variables. A regression model involving multiple variables can be represented as:

$$y = b_0 + m_1 b_1 + m_2 b_2 + m_3 b_3 + \ldots + m_n b_n$$

Linear Regression Theory - Review

- The above equation is called **hyperplane**.
- Linear regression in two dimensions is a straight line; in three dimensions it is a plane; and in more than three dimensions, a hyperplane.

Simple Linear Regression using Scikit-Learn

For discussing the simple linear regression using Scikit-learn, we use the dataset on weather reports from the period of World war II to compare with missions in the bombing operations dataset. This is a publicly available dataset and can be downloaded from **here**

The dataset contains information on weather conditions recorded on each day at various weather stations around the world. Information includes precipitation, snowfall, temperatures, wind speed and whether the day included thunderstorms or other poor weather conditions.

Our task is to predict the maximum temperature taking input feature as the minimum temperature.

Simple Linear Regression using Scikit-Learn

Import the required packages

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as seabornInstance
from sklearn.model_selection import
   train_test_split
from sklearn.linear_model import
   LinearRegression
from sklearn import metrics
```

Simple Linear Regression using Scikit-Learn

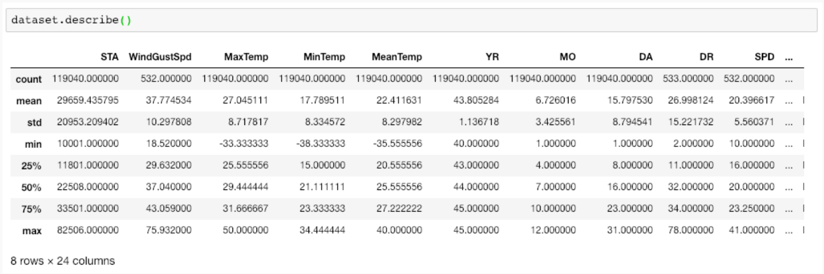
We now import the CSV dataset into a DataFrame using pandas:

```python
dataset = pd.read_csv('Weather.csv')
```

We now explore the dataset attributes and statistical description

```python
print('Dataset size is: ',dataset.shape)
print(dataset.describe())
```

Simple Linear Regression using Scikit-Lea



```
dataset.describe()
```

	STA	WindGustSpd	MaxTemp	MinTemp	MeanTemp	YR	MO	DA	DR	SPD	...
count	119040.000000	532.000000	119040.000000	119040.000000	119040.000000	119040.000000	119040.000000	119040.000000	533.000000	532.000000	...
mean	29659.435795	37.774534	27.045111	17.789511	22.411631	43.805284	6.726016	15.797530	26.998124	20.396617	...
std	20953.209402	10.297808	8.717817	8.334572	8.297982	1.136718	3.425561	8.794541	15.221732	5.560371	...
min	10001.000000	18.520000	-33.333333	-38.333333	-35.555556	40.000000	1.000000	1.000000	2.000000	10.000000	...
25%	11801.000000	29.632000	25.555556	15.000000	20.555556	43.000000	4.000000	8.000000	11.000000	16.000000	...
50%	22508.000000	37.040000	29.444444	21.111111	25.555556	44.000000	7.000000	16.000000	32.000000	20.000000	...
75%	33501.000000	43.059000	31.666667	23.333333	27.222222	45.000000	10.000000	23.000000	34.000000	23.250000	...
max	82506.000000	75.932000	50.000000	34.444444	40.000000	45.000000	12.000000	31.000000	78.000000	41.000000	...

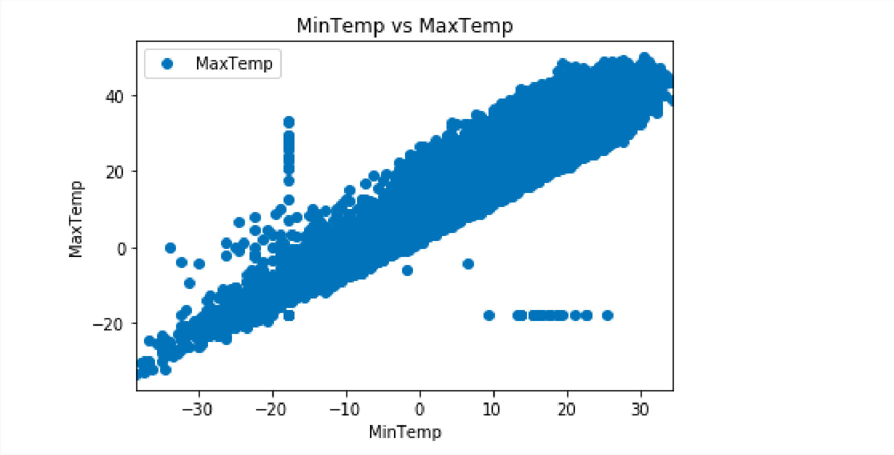
8 rows × 24 columns

Simple Linear Regression using Scikit-Lea

Before proceeding to linear regression, first let's plot our data points related to MinTemp and MaxTemp on a 2—D graph to view our dataset and see if we can manually find any relationship between the data:

```python
dataset.plot(x='MinTemp', y='MaxTemp',
   style='o', figsize=(15,10))
plt.title('MinTemp vs MaxTemp')
plt.xlabel('MinTemp')
plt.ylabel('MaxTemp')
plt.show()
```

Simple Linear Regression using Scikit-Learn

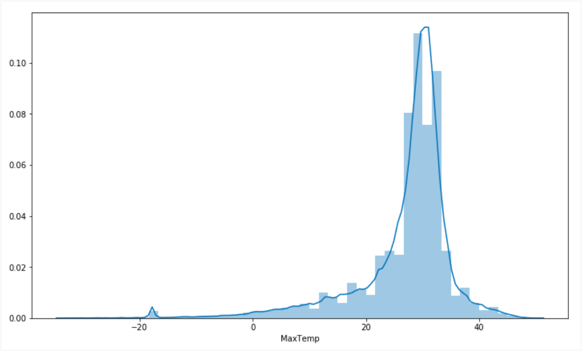


Simple Linear Regression using Scikit-Learn

We now plot Average Maximum Temperature along with univariate distribution of observations. For this we will use `seaborn.distplot()`

```
plt.figure(figsize=(15,10))
plt.tight_layout()
seabornInstance.distplot(dataset['MaxTemp'
    ])
```

Simple Linear Regression using Scikit-Learn



Observe that Average Maximum temperature which is in between 25 and 35.

Simple Linear Regression using Scikit-Learn

- Our next step is to divide the data into "**attributes**" and "**labels**".
- **Attributes** are the independent variables while **labels** are dependent variables whose values are to be predicted.
- In our dataset, we only have two columns.
- We want to predict the MaxTemp depending upon the MinTemp recorded.
- Therefore our attribute set will consist of the "MinTemp" column which is stored in the X variable, and the label will be the "MaxTemp" column which is stored in y variable.

Simple Linear Regression using Scikit-Learn

```
X = dataset['MinTemp'].values.reshape
   (-1,1)
y = dataset['MaxTemp'].values.reshape
   (-1,1)

print('Attributes size is: ',X.shape)
print('Labels size is: ',y.shape)
```

Simple Linear Regression using Scikit-Lea

- Next, we split 80% of the data to the training set while 20% of the data to test set using `tranin_test_split()` function of `sklearn.model_selection`.
- The test_size variable is where we actually specify the proportion of the test set.

Simple Linear Regression using Scikit-Lea

```python
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.2,
    random_state=0)

print('Training Attributes size is: ',
    X_train.shape)
print('Testing Attributes size is: ',
    X_test.shape)
print('Training Labels size is: ',y_train.
    shape)
print('Testing Lables size is: ' ,y_test.
    shape)
```

Simple Linear Regression using Scikit-Lea

- After splitting the data into training and testing sets, finally, the time is to train our algorithm.
- For that, we need to import `LinearRegression` class, instantiate it, and call the `fit()` method along with our training data.

```
regressor = LinearRegression()
regressor.fit(X_train, y_train) #training
    the algorithm
```

Simple Linear Regression using Scikit-Lea

- The linear regression model basically finds the best value for the intercept and slope, which results in a line that best fits the data.
- To see the value of the intercept and slope calculated by the linear regression algorithm for our dataset, execute the following code.

```
print('Intercept is: ',regressor.intercept_) #To
    retrieve the intercept:
print('Coefficient is: ',regressor.coef_) #For
   retrieving the slope:
```

Result is:

```
10.66185201
0.92033997
```

This means that for every one unit of change in Min temperature, the change in the Max temperature is about 0.92%.

Simple Linear Regression using Scikit-Learn

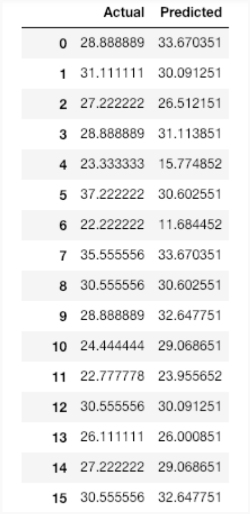
- Now that we have trained our algorithm, it’s time to make some predictions.
- To do so, we will use our test data and see how accurately our algorithm predicts the percentage score.
- To make predictions on the test data, execute the following code.

```
y_pred = regressor.predict(X_test)
```

- Now compare the actual output values for ‘X_test‘ with the predicted values.

```
df = pd.DataFrame({'Actual': y_test.flatten
   (), 'Predicted': y_pred.flatten()})
df
```

Simple Linear Regression using Scikit-Lea



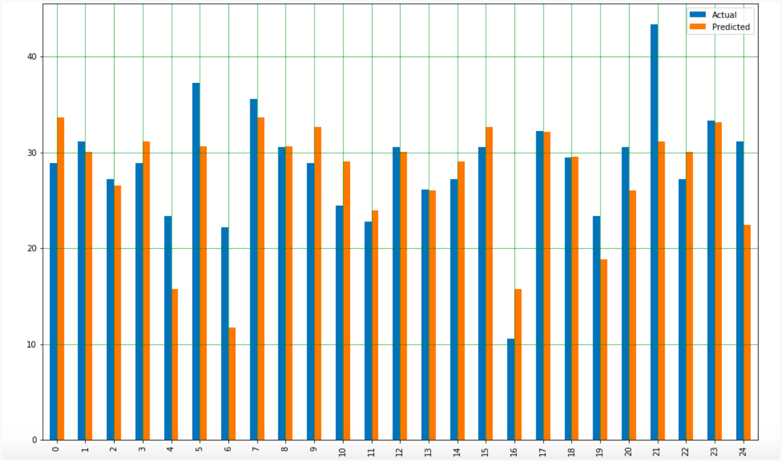
	Actual	Predicted
0	28.888889	33.670351
1	31.111111	30.091251
2	27.222222	26.512151
3	28.888889	31.113851
4	23.333333	15.774852
5	37.222222	30.602551
6	22.222222	11.684452
7	35.555556	33.670351
8	30.555556	30.602551
9	28.888889	32.647751
10	24.444444	29.068651
11	22.777778	23.955652
12	30.555556	30.091251
13	26.111111	26.000851
14	27.222222	29.068651
15	30.555556	32.647751

Simple Linear Regression using Scikit-Lea

- We can also visualize comparison result as a bar graph. Note: As the number of records is huge, for representation purpose we will consider the first 25 records only.

```
df1 = df.head(25)
df1.plot(kind='bar',figsize=(15,10))
plt.grid(which='major', linestyle='-',
   linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':',
   linewidth='0.5', color='black')
plt.show()
```

Simple Linear Regression using Scikit-Lea



Simple Linear Regression using Scikit-Learn

- We now plot the straight line with the test data.

```
plt.figure(figsize=(15,10))
plt.scatter(X_test, y_test,  color='gray')
plt.plot(X_test, y_pred, color='red',
   linewidth=2)
plt.show()
```

The straight line above shows that our algorithm is working correctly.

Simple Linear Regression using Scikit-Learn

- The final step in our modelling approach is to evaluate the performance of our algorithm.
- This is very important step to compare how well different algorithms perform on a particular dataset.

Simple Linear Regression using Scikit-Lea

- For regession algorithms, three evaluation metrics are commonly used:

1. **Mean Absolute Error (MAE)** is the mean of the absolute of the errors. It is calculated as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |Y_i - y_i|$$

2. **Mean Squared Error (MSE)** is the mean of the squared errors. It is calculated as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (Y_i - y_i)^2$$

3. **Root Mean Squared Error (RMSE)** is the square root of the mean of the squared errors. It is calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (Y_i - y_i)^2}$$

Simple Linear Regression using Scikit-Learn

- The above tree metrics can be computed using the pre-built functions in `sklearn.metrics` library.

```
print('Mean Absolute Error:', metrics.
   mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.
   mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(
   metrics.mean_squared_error(y_test,
   y_pred)))
```

Simple Linear Regression using Scikit-Lea

Output:

```
('Mean Absolute Error:', 3.19932917837853)
('Mean Squared Error:',
   17.631568097568447)
('Root Mean Squared Error:',
   4.198996082109204)
```

- Observe that the RMS error is 4.19, which is more than 10% of the mean value of the percentages of all the temperature i.e, 22.41.
- This means that our algorithm was not very accurate but can still make reasonably good predictions.