MEAN SQUARED ERROR

Calculating the MSE in Python and Scikit-Learn

datagy.io

The mean squared error is a common way to measure the prediction accuracy of a model. **In this tutorial, you'll learn how to calculate the mean squared error in Python**. You'll start off by learning what the mean squared error represents. Then you'll learn how to do this using Scikit-Learn (sklean), Numpy, as well as from scratch.

Table of Contents

What is the Mean Squared Error

The **mean squared error measures the average of the squares of the errors**. What this means, is that it returns the average of the sums of the square of each difference between the estimated value and the true value.

The MSE is always positive, though it can be 0 if the predictions are completely accurate. It incorporates the variance of the estimator (how widely spread the estimates are) and its bias (how different the estimated values are from their true values).

The formula looks like below:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

The formula for the mean squared error (MSE)

Now that you have an understanding of how to calculate the MSE, let's take a look at how it can be calculated using Python.

Interpreting the Mean Squared Error

The mean squared error is always 0 or positive. When a MSE is larger, this is an indication that the linear regression model doesn't accurately predict the model.

An important piece to note is that the MSE is sensitive to outliers. This is because it calculates the average of every data point's error. Because of this, a larger error on outliers will amplify the MSE.

There is no "target" value for the MSE. The MSE can, however, be a good indicator of how well a model fits your data. It can also give you an indicator of choosing one model over another.

Loading a Sample Pandas DataFrame

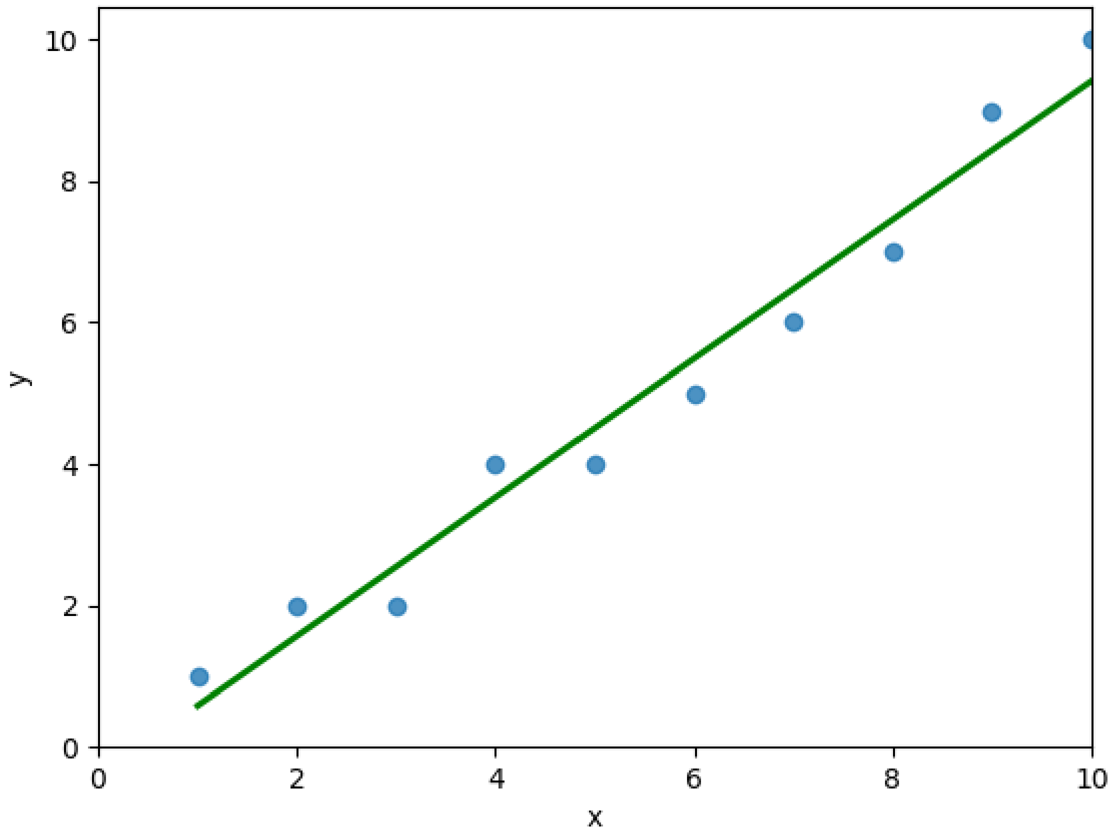
Let's start off by loading a sample Pandas DataFrame. If you want to follow along with this tutorial line-by-line, simply copy the code below and paste it into your favorite code editor.

```
# Importing a sample Pandas DataFrame
import pandas as pd

df = pd.DataFrame.from_dict({
    'x': [1,2,3,4,5,6,7,8,9,10],
    'y': [1,2,2,4,4,5,6,7,9,10]})

print(df.head())
#    x  y
# 0  1  1
# 1  2  2
# 2  3  2
# 3   4   4
# 4   5   4
```

You can see that the editor has loaded a DataFrame containing values for variables `x` and `y`. We can plot this data out, including the line of best fit using Seaborn’s `.regplot()` function:

```
# Plotting a line of best fit
import seaborn as sns
import matplotlib.pyplot as plt
sns.regplot(data=df, x='x', y='y', ci=None)
plt.ylim(bottom=0)
plt.xlim(left=0)
plt.show()
```

This returns the following visualization:



Plotting a line of best fit to help visualize mean squared error in Python

The mean squared error calculates the average of the sum of the squared differences between a data point and the line of best fit. By virtue of this, the lower a mean sqared error, the more better the line represents the relationship.

We can calculate this line of best using Scikit-Learn. You can learn about this in [this in-depth tutorial on linear regression in sklearn](). The code below predicts values for each `x` value using the linear model:

```
# Calculating prediction y values in sklearn
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(df[['x']], df['y'])
y_2 = model.predict(df[['x']])
df['y_predicted'] = y_2
print(df.head())

# Returns:
#    x  y  y_predicted
# 0  1  1     0.581818
# 1  2  2     1.563636
# 2  3  2     2.545455
# 3  4  4     3.527273
# 4  5  4     4.509091
```

Calculating the Mean Squared Error with Scikit-Learn

The simplest way to calculate a mean squared error is to use Scikit-Learn (sklearn). The `metrics` module comes with a function, `mean_squared_error()` which allows you to pass in true and predicted values.

Let’s see how to calculate the MSE with sklearn:

```
# Calculating the MSE with sklearn
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(df['y'], df['y_predicted'])
print(mse)

# Returns: 0.24727272727272714
```

This approach works very well when you’re already importing Scikit-Learn. That said, the function works easily on a Pandas DataFrame, as shown above.

In the next section, you’ll learn how to calculate the MSE with Numpy using a custom function.

Calculating the Mean Squared Error from Scratch using Numpy

Numpy itself doesn’t come with a function to calculate the mean squared error, but you can easily define a custom function to do this. We can make use of the `subtract()` function to subtract arrays element-wise.

```
# Definiting a custom function to calculate the MSE
import numpy as np

def mse(actual, predicted):
    actual = np.array(actual)
    predicted = np.array(predicted)
    differences = np.subtract(actual, predicted)
    squared_differences = np.square(differences)
    return squared_differences.mean()

print(mse(df['y'], df['y_predicted']))

# Returns: 0.24727272727272714
```

The code above is a bit verbose, but it shows how the function operates. We can cut down the code significantly, as shown below:

```
# A shorter version of the code above
import numpy as np

def mse(actual, predicted):
    return np.square(np.subtract(np.array(actual), np.array(predicted))).mean()

print(mse(df['y'], df['y_predicted']))

# Returns: 0.24727272727272714
```

Conclusion

In this tutorial, you learned what the mean squared error is and how it can be calculated using Python. First, you learned how to use Scikit-Learn's `mean_squared_error()` function and then you built a custom function using Numpy.

The MSE is an important metric to use in evaluating the performance of your machine learning models. While Scikit-Learn abstracts the way in which the metric is calculated, understanding how it can be implemented from scratch can be a helpful tool.

Additional Resources

To learn more about related topics, check out the tutorials below:

- Pandas Variance: Calculating Variance of a Pandas Dataframe Column
- Calculate the Pearson Correlation Coefficient in Python
- How to Calculate a Z-Score in Python (4 Ways)
- Official Documentation from Scikit-Learn