MEAN SQUARE ERROR & R2 SCORE CLEARLY EXPLAINED

Today we're going to introduce some terms that are important to machine learning:

- Variance
- r2 score
- Mean square error

We illustrate these concepts using scikit-learn.

(This article is part of our scikit-learn Guide. Use the right-hand menu to navigate.)

Why these terms are important

You need to understand these metrics in order to determine whether regression models are accurate or misleading. Following a flawed model is a bad idea, so it is important that you can quantify how accurate your model is. Understanding that is not so simple.

These first metrics are just a few of them. Other concepts, like **bias** and **overtraining** models, also yield misleading results and incorrect predictions.

(Learn more in Bias and Variance in Machine Learning.)

To provide examples, let's use the code from our last blog post, and add additional logic. We'll also introduce some randomness in the dependent variable (**y**) so that there is some error in our predictions. (Recall that, in the last blog post we made the independent **y** and dependent variables x perfectly correlate to illustrate the basics of how to do linear regression with scikit-learn.)

What is variance?

In terms of linear regression, **variance** is a measure of how far observed values differ from the average of predicted values, i.e., their difference from the **predicted value mean**. The goal is to have a value that is low. What **low** means is quantified by the **r2 score** (explained below).

In the code below, this is **np.var(err)**, where **err** is an array of the differences between observed and predicted values and **np.var()** is the numpy array variance function.

What is r2 score?

The **r2 score** varies between 0 and 100%. It is closely related to the **MSE** (see below), but not the same. Wikipedia defines **r2** as

" ...the proportion of the variance in the dependent variable that is predictable from the independent variable(s)."

Another definition is "(total variance explained by model) / total variance." So if it is 100%, the two variables are perfectly correlated, i.e., with no variance at all. A low value would show a low level of correlation, meaning a regression model that is not valid, but not in all cases.

Reading the code below, we do this calculation in three steps to make it easier to understand. **g** is the sum of the differences between the observed values and the predicted ones. **(ytest - preds) **2**. **y** is each observed value **y** minus the average of observed values **np.mean(ytest)**. And then the results are printed thus:

```
print ("total sum of squares", y)
print ("ẗotal sum of residuals ", g)
print ("r2 calculated", 1 - (g / y))
```

Our goal here is to explain. We can of course let scikit-learn to this with the r2_score() method:

```
print("R2 score : %.2f" % r2_score(ytest,preds))
```

What is mean square error (MSE)?

Mean square error (MSE) is the average of the square of the errors. The larger the number the larger the error. **Error** in this case means the difference between the observed values y1, y2, y3, ... and the predicted ones pred(y1), pred(y2), pred(y3), ... We square each difference (pred(yn) - yn)) ** 2 so that negative and positive values do not cancel each other out.

The complete code

So here is the complete code:

```
import matplotlib.pyplot as plt
from sklearn import linear_model
import numpy as np
from sklearn.metrics import mean_squared_error, r2_score

reg = linear_model.LinearRegression()
ar = np.array(,,], ,,]])
y = ar
x = ar
reg.fit(x,y)
print('Coefficients: n', reg.coef_)
xTest = np.array(,,])
ytest =  np.array(,,])

preds = reg.predict(xTest)
print("R2 score : %.2f" % r2_score(ytest,preds))
print("Mean squared error: %.2f" % mean_squared_error(ytest,preds))

er = []
g = 0
for i in range(len(ytest)):
    print( "actual=", ytest, " observed=", preds)
    x = (ytest - preds) **2
    er.append(x)
    g = g + x
x = 0
for i in range(len(er)):
   x = x + er

print ("MSE", x / len(er))

v = np.var(er)
print ("variance", v)

print ("average of errors ", np.mean(er))

m = np.mean(ytest)
print ("average of observed values", m)

y = 0
for i in range(len(ytest)):
    y = y + ((ytest - m) ** 2)
```

```
print ("total sum of squares", y)
print ("ẗotal sum of residuals ", g)
print ("r2 calculated", 1 - (g / y))
```

Results in:

```
Coefficients:
 ]
R2 score : 0.62
Mean squared error: 2.34
actual=  observed=
actual=  observed=
actual=  observed=
MSE
variance 1.2881398892129619
average of errors 2.3402861111111117
average of observed values 10.5
total sum of squares
ẗotal sum of residuals
r2 calculated
```

You can see by looking at the data **np.array(,,], ,,]])** that every dependent variable is roughly twice the independent variable. That is confirmed as the calculated coefficient **reg.coef_** is 2.015.

There is no correct value for **MSE**. Simply put, the lower the value the better and 0 means the model is perfect. Since there is no correct answer, the MSE's basic value is in selecting one prediction model over another.

Similarly, there is also no correct answer as to what **R2** should be. 100% means perfect correlation. Yet, there are models with a low R2 that are still good models.

Our take away message here is that you cannot look at these metrics in isolation in sizing up your model. You have to look at other metrics as well, plus understand the underlying math. We will get into all of this in subsequent blog posts.

Additional Resources

Extending R-squared beyond ordinary least-squares linear regression from pcdjohnson