Data Science

Linear Regression

CS202 11.21.22

Linear Regression

Regression Evaluation Metrics

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)

Linear Regression

Mean Absolute Error

In statistics **mean absolute error** (**MAE**) is a measure errors between paired observations expressing the same phenomenon.

Examples of *Y* versus *X* include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement.

MAE is calculated as the **sum of absolute errors** divided by the sample size.

Linear Regression

Mean Absolute Error

Mean Absolute Error is calculated via the following formula:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

It is an arithmetic average of the absolute errors. Where y_i is the prediction and $\hat{y}_i$ is the actual value. We use the absolute value of all the errors so that negative values are taken to account appropriately.

Linear Regression

Mean Squared Error

Another way to calculate error is via calculating MSE.

In statistics, the **mean squared error** (**MSE**) of an estimator measures the average of the squares of the errors.

In other words, the average squared difference between the estimated values and the actual value.

In machine learning MSE may refer to the *empirical* risk (the average loss on an observed data set), as an estimate of the true MSE (the true risk: the average loss on the actual population distribution).

Linear Regression

Mean Squared Error

The MSE tells you how close your regression line is to a set of points. It does this by taking the distances of those points to the regression line. Those errors are squared to remove any negative signs.

The MSE also gives more weight to larger differences. It calculates the average of these errors squared. The lower the MSE, the better the forecast of your regression line.

Linear Regression

MAE vs. MSE

The Mean Squared Error (MSE) is measured via the following formula:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where notation for the formula is as before (same as MAE).

Here, we can see that the higher errors take weigh more in the metric than lower ones just by the nature of the power function.

Linear Regression

MSE vs. RMSE

A backlash in MSE is the fact that the unit of the metric is also squared, so if the model tries to predict price in US$, the MSE will yield a number with unit $(US\$)^2$ which does not make sense.

Therefore, Root mean squared error (**RMSE**) is used then to return the MSE error to the original unit by taking the square root of it, while maintaining the property of penalizing higher errors.

Linear Regression

Root Mean Squared Error

The **root-mean-square error** (**RMSE**) is a frequently used measure of the differences between values (sample or population values) predicted by a model and the values observed.

The RMSE serves to aggregate the magnitudes of the errors in predictions for various data points into a single measure of predictive power. RMSE is a measure of accuracy, to compare forecasting errors of different models for a particular dataset and not between datasets, as it is scale-dependent.

Linear Regression

Root Squared Mean Error

RMSE is always non-negative, and a value of 0 (almost never achieved in practice) would indicate a perfect fit to the data.

A lower RMSE is better than a higher one.

RMSE is the square root of the average of squared errors (the MSE). The effect of each error on RMSE is proportional to the size of the squared error; thus larger errors have a disproportionately large effect on RMSE. Consequently, RMSE is sensitive to outliers (as is MSE).

Linear Regression

Root Mean Squared Error

The Root Mean Squared Error (RMSE) is given by:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Where the notation is as in previous slides (MAE and MSE).

Linear Regression

Regression Evaluation Metrics

Ok. So how do we do this using sklearn?

Easy! We first import metrics from sklearn:

```
from sklearn import metrics
```

Then we input our predictions vs. our testing variables. Remember last time, we had a “y_test” when we were performing a train_test_split. (We will look closer during demo):

```
metrics.mean_absolute_error(y_test, predictions) #will give MAE
metrics.mean_squared_error(y_test, predictions) #will give MSE
np.sqrt(metrics.mean_squared_error(y_test, predictions) #will give RMSE
```

All you need to do now is print them out.

Linear Regression

K-Folds Cross Validation

Generalization is a frequent term that is used in conversations about Machine Learning. It refers to how able the model can adapt to new, unseen data and how it works effectively using various inputs. It is understandable to say that if new unseen data is inputted into a model, if this unseen data has similar characteristics to the training data, it will perform well.

Generalizing things is easy to us humans, however it can be challenging to Machine Learning models. This is where Cross-Validation comes into the picture.

Linear Regression

K-Folds Cross Validation

You may see cross-validation also being referred to as rotation estimation and/or out-of-sample testing. The overall aim of Cross-Validation is to use it as a tool to evaluate machine learning models, by training a number of models on different subsets of the input data.

Cross-validation can be used to detect overfitting in a model which infers that the model is not effectively generalizing patterns and similarities in the new inputted data.

Linear Regression

K-Folds Cross Validation

In order to perform cross-validation, the following steps are typically taken:

1. Split the dataset into training data and test data
2. The parameters will undergo a Cross-Validation test to see which are the best parameters to select.
3. These parameters will then be implemented into the model for retraining
4. Final evaluation will occur and this will depend if the cycle has to go again, depending on the accuracy and the level of generalization that the model performs.

Linear Regression

K-Folds Cross Validation

K-fold Cross-Validation is when the dataset is split into a K number of folds and is used to evaluate the model's ability when given new data.

K refers to the number of groups the data sample is split into.

For example, if you see that the k-value is 5, we can call this a 5-fold cross-validation.

Each fold is used as a testing set at one point in the process.

Linear Regression

K-Fold Cross Validation Process

- Choose your k-value
- Split the dataset into the number of k folds.
- Start off with using your k-1 fold as the test dataset and the remaining folds as the training dataset
- Train the model on the training dataset and validate it on the test dataset
- Save the validation score
- Repeat steps 3 – 5, but changing the value of your k test dataset. So we chose k-1 as our test dataset for the first round, we then move onto k-2 as the test dataset for the next round.
- By the end of it you would have validated the model on every fold that you have.
- Average the results that were produced in step 5 to summarize the skill of the model.

Linear Regression

K-Fold Cross Validation

Why use a k-fold cross validation?

We have a lot of readily available data that can explain a lot of things and help us identify hidden patterns. However, if we only have a small dataset, splitting it into a training and test dataset at a 80:20 ratio respectively doesn’t seem to do much for us.

However, when using K-fold cross validation, all parts of the data will be able to be used as part of the testing data. This way, all of our data from our small dataset can be used for both training and testing, allowing us to better evaluate the performance of our model.

Linear Regression
K-Fold Cross Validation

Secondly, we have more available metrics to help evaluate:

Let’s refer back to the example of splitting your dataset into training and testing as a 80:20 ratio or a train-test split - this provides us with only one result to refer to in our evaluation of the model. We are not sure about the accuracy of this result, if it’s due to chance, the level of bias, or if it actually performed well.

However, when using k-fold cross validation, we have more models that will be producing more results. For example, if we chose our k value at 10, we would have 10 results to use in our evaluation of the model's performance.

If we were using accuracy as our measurement; having 10 different accuracy results where all of the data was used in the test phase is always going to be better and more reliable than using one accuracy result that was produced by a train-test split - where all the data wasn’t used in the test phase.

You would have more trust and confidence in your model's performance if the accuracy outputs were 94.0, 92.8, 93.0, 97.0, and 94.5 in a 5-fold cross validation than a 93.0 accuracy in a train-test split. This proves to us that our algorithm is generalizing and actively learning and is providing consistent reliable outputs.

Linear Regression

K-Fold Cross Validation

Thirdly, we get better performance using dependent data:

In a random train-test split, we assume that the data inputs are independent. Let's further expand on this. Let's say we are using a random train-test split on a speech recognition dataset for British English speakers who reside from London. There are 5 speakers, with 250 recording each. Once the model performs a random train-test split, both the training and testing dataset will learn from the same speaker, which will be saying the same dialogue.

Yes this does improve the accuracy and boost the performance of the algorithms, however, what about when a new speaker is introduced to the model? How does the model perform then?

Using K-fold cross validation will allow you to train 5 different models, where in each model you are using one of the speakers for the testing dataset and the remaining for the training dataset. This way, not only can we evaluate the performance of our model, but our model will be able to perform better on new speakers and can be deployed in production to produce similar performance for other tasks.

Linear Regression

K-Fold Cross Validation

What is the desirable k?

Choosing the right k-value is important as it can affect your level of accuracy, variance, bias and cause you to misinterpreted the overall performance of your model.

The simplest way to choose your k value is by equating it to your ‘n’ value, also known as leave-one-out cross-validation. n represents the size of the dataset which allows for each test sample the opportunity to be used in the test data or hold out data.

However, through various experimentations from Data Scientists, Machine Learning Engineers, and Researchers they have found that choosing a k-value of 10 has proven to provide a low bias and a modest variance.

Linear Regression

K-Folds Cross Validation

Cross-Validation is an important tool that every Data Scientist should be using or very proficient in at least. It allows you to make better use of all your data as well as providing Data Scientists, Machine Learning Engineers and Researchers with a better understanding of the performance of the algorithm.

Having confidence in your model is important to future deployment and trust that the model will be effective and perform well.

Linear Regression

K-Folds Cross Validation

Alright! So how do we do this using sklearn?

This is a little more tricky to do: so we will look at the code from our demo portion of the class to see exactly how this is performed.

Ok! Lets get to it!