Applying Machine Learning

Applying machine learning to problems can be a difficult task because of all the different models that are offered. Discover how to evaluate and select machine learning models and apply machine learning to a problem.

Table of Contents

Bias and Variance

[Topic title: Bias and Variance. The presenter is Jamie Campbell.] In this video, I'll describe the two main types of error in machine learning models and the trade-off between them. There are two main types of error in machine learning models and there's a trade-off depending on which one you use.

The first type is bias, which is error caused by making assumptions. It's mathematically defined by taking the difference of the expected average value estimated and the actual value. The expected estimated value is the average of values if a model is trained multiple times. For example, fitting a linear regression line 100 times or getting yi for each and taking the average. High-bias models require many assumptions. For example, linear regression mode and Naive Bayes. And these models can miss certain relationships in the data.

The second type is variance, which is error caused by fluctuations in the data, for example noise. High variance can cause a model to be prone to overfitting – so excessive noise. And training the same model on different data sets will yield different results when the variance is high. Here are the trade-offs between bias and variance. First, in machine learning, every model has some bias and some variance. Decreasing variance increases the bias. And decreasing bias increases the variance. Some models are more prone to different types of error. Changing model parameters can help mitigate one type of error but it can't solve both. And so hybrid models can be used to mitigate both types of error.

Here is a comparison of bias and variance in supervised learning models. High-bias models use linear regression with regularization, the k nearest number model, with high k, a shallow or pruned decision tree, and neural networks with not a lot of hidden units. In high variance models, linear regression is

utilized along with k nearest neighbor with a low k value. They represent a deep decision tree and neural networks with many hidden units.

Cross-validation

[Topic title: Cross-validation. The presenter is Jamie Campbell.] In this video, I'll describe how cross-validation is used. *[Training, Testing, and Validation Sets.]* When using training, testing, and validation sets in neural networks, using the training sets to test a model leads to overfitting. A statistical anomaly that occurs when data noise obfuscates any relationships the data might normally indicate. A model can simply remember all inputs to make perfect predictions. But it won't be able to predict different examples correctly. In such scenarios, a test set should be extracted for experimenting. The model can still be overfitted to learn how to be good at predicting only the test set. And an extract validation set is used to mitigate overfitting in a test set. *[Cross-validation.]* So here is the major issue. Splitting a dataset into three can make it too small for proper training. Cross-validation, on the other hand, can split a dataset k times. Train each dataset, calculate the accuracy for each dataset, and average the metrics over each. This method requires a lot more computations than regular splitting. But it returns much stronger evaluation metrics.

Cross-validation in Python

[Topic title: Cross-validation in Python. The presenter is Jamie Campbell.] In this video, I'll discuss how to perform cross-validation in Python to obtain strong evaluation scores. *[The scikit learn page is open in the scikit-learn web site. This page includes various sections such as 3.1. Cross-validation: evaluating estimator performance and 3.1.1. Computing cross-validated metrics.]* When you're using training, testing, and validation sets in neural networks, using the training sets to test a model leads to overfitting, which is a statistical anomaly that occurs when data noise obfuscates any relationship that the data might normally indicate. A model can simply remember all inputs to make perfect predictions, but it won't be able to predict different examples correctly. And, in such scenarios, a test set should be extracted for experimenting. The model can still be overfitted to learn how to be good at predicting only the test set and an extract validation set is used to mitigate overfitting in a test set. So here is the issue.

Splitting a dataset into three can make it too small for proper training. Cross-validation, on the other hand, can split a dataset k times, train each dataset, calculate the accuracy for each dataset, and average the metrics over each. This method requires a lot more computations than regular splitting, but it returns much stronger evaluation metrics. Now *[He highlights the URL http://scikit-learn.org/stable/modules/cross_validation.html.]* this tutorial provides a good introduction to cross-validation with Python using the train_test_split helper function. It loads Iris data and fits it to a linear support vector machine sampling a training set and holding on to 40% of the data for testing. And it does much more. There are several code samples here and some useful information on how cross-validation is implemented. *[He points at the various sections on the page.]* So I recommend you check this out and try some of these examples so you can better understand cross-validation and how it can be implemented.

Metrics for Binary Classification Models

[Topic title: Metrics for Binary Classification Models. The presenter is Jamie Campbell.] In this video, I'll discuss different metrics that can be used to evaluate binary classification models. There are different metrics that can be used to evaluate binary classification models. *[Precision and Recall.]* The confusion matrix is a visual method used to represent what kind of errors a model is making. For example, in a binary classification, you can determine the possible outcomes of a simple yes or no answer with a table like this. You begin with a prediction made by a machine learning algorithm. The possible answer being either yes or no. And map that choice against the correct answer, again, yes or no. You're left with four possible outcomes – true positive, false negative, false positive, or true negative. Precision refers to the ratio of correct answers. So, in a binary example, we would determine precision by dividing true positives by true positives plus false positives. And recall refers to the ratio of true examples that are correctly identified. So we define recall by dividing true positives by true positives plus false negatives. *[F1 Score.]*

An F1 score is a metric that considers both precision and recall equally using this formula. *[The formula is, F1 is equal to twice the result of precision multiplied by recall divided by precision plus recall.]* F1 equals 1 is the best score and F1 equals 0 is the worst score. *[Alternate F-measures.]* There are alternate F measures. F1 can be generalized to weight precision and recall differently using this formula. *[The formula is F beta is equal to one plus beta square multiplied by the result of precision multiplied by recall divided by beta square multiplied by the precision plus recall.]* Beta < 1 means precision is more important and beta > 1 means recall is more important. *[Matthews Correlation Coefficient.]* The Matthews Correlation Coefficient is a balanced coefficient that returns a number between -1 and 1. 1 is a perfect prediction. 0 is a completely random prediction. And -1 is a complete inverse prediction. Balanced refers to the fact that it can still be used when class sizes have a large difference. *[Cohen Kappa Score.]* And the Cohen Kappa Score is a score that compares the model to a random selection with the expectation that accuracy will be attained using this formula. *[The formula is k is equal to the p0 - pe divided by 1 - pe.]* P0 is the model's accuracy when tested. Pe is the expected accuracy if random choices were made. And the score is interpreted using the following scale – 0.8-1 is considered a strong score. 0.2-0.6 is considered fair or moderate. 0- 0.2 is a slight score. And anything less than 0 isn't any better than random classification.

Metrics for Non-binary Classification Models

[Topic title: Metrics for Non-binary Classification Models. The presenter is Jamie Campbell.] In this video, I'll describe different metrics that can be used to evaluate non-binary classification models. Different metrics can be used to evaluate non-binary classification models. *[Multiclass Metrics.]* Multiclass metrics use more than two possible target classes. Binary metrics can be used for each class, for example, Class 1 versus everything else or Class 2 versus everything else and so on. A multiclass F1 score takes the weighted average of the F1 score of each class. *[Multilabel and Ranked Metrics.]* In multilabel metrics, each input has one or more target classes. Ranked labels are a returned set of values which are ranked. And they can be split like multiclass metrics for each label at each position, for example, Class 1 at position 1 versus everything else. Class 1 at position 2 versus everything else and so on. *[Average Metrics.]* With average metrics, precision, recall, and other metrics are calculated across all possible classes and label sets. Metrics are averaged using different methods. The macro average takes the average over all classes. The weighted average weights classes by how common they are when averaging a metric. And the micro average sums all metrics' dividends and divisors and uses that ratio.

Classification Metrics in Python

[Topic title: Classification Metrics in Python. The presenter is Jamie Campbell.] In this video, I'll discuss how you can use Python to calculate common valuation metrics. There are different metrics that can be used to evaluate binary classification models. *[The Classification metrics section is open in the scikit-learn web site.]* The confusion matrix is a visual method used to represent what kind of errors a model is making. For example, in binary classifications, you can determine the possible outcomes of a simple yes or no answer using a table. Precision refers to the ratio of correct answers and recall refers to the ratio of true examples that are truly identified. AN F1 score is a metric that considers both precision and recall equally and their alternate F measures. There is also the math use correlation coefficient. A balanced coefficient returns a number between -1 and 1.

With 1 being a perfect prediction, 0 a completely random prediction, and -1 a complete inverse prediction. And the Cohen Kappa Score is a score that compares the model to a random selection with the expectation that accuracy will be attained. Now, on the scikit-learn site, you can find out more about classification metrics and the sklearn.metrics modules includes functions that can be used to measure classification performance. Some work only with binary classification. Others work with multi-class and multi-label instances. Some are used for ranking and some can be used with binary and multi-class problems, but not multi-class. Now clicking on any of these metrics will take you to a page where you can find out more about the syntax used *[He clicks the accuracy_score link and the sklearn.metrics.accuracy_score web page opens.]* as well as examples.

So let's take a look at this one. This is accuracy score and here is some information about the metric and a brief code example, which I'll demonstrate now. *[He opens the Python REPL (ptpython) editor window.]* So we start by importing numpy as np. And then from sklearn.metrics import accuracy_score. Okay. So we'll create y_pred= [0,1,2,3]. And, again, you should check out the example on the scikit-learn page to get more information about this. I'll create one called y_true. And this is actually what's being done on the scikit-learn page. [0,1,2,3]. Okay, and now we can use the accuracy_score using the two parameters y pred and y true. And there you go.

Metrics for Regression Models

[Topic title: Metrics for Regression Models. The presenter is Jamie Campbell.] In this video, I'll describe different metrics that can be used to evaluate regression models. There are different metrics that can be used to evaluate regression models. Mean squared error – whose formula is shown here – is often used as a cost function. *[Mean Squared Error. The formula for MSE is displayed on the screen. The formula is given at the end of transcript under the heading MSE.]]* It describes how good a model is at making predictions with 0 being the best score. And higher scores indicating increasingly error prone models. *[Other Measures of Error.]* Other measures of error include mean absolute error. For example, a non square mean square error as demonstrated by this formula. *[The formula is given at the end of transcript under the heading MAE.]*

MSE is typically preferred. It takes variance into account and it eliminates statistical bias. The mean square log error is good for the exponential growth model. The output becomes linear and here's the formula. *[The formula is given at the end of transcript under the heading MLSE.] [R2 Score.]* And an R2 score determines how well the model can describe the data. 1 is the best score and 0 represents a model that always returns average output from a training set. That is, it always returns an expected value. If the score is less than 0, that indicates it's worse than using the expected value.

MSE:

1/n∑ i=0 to m (f(xi) - yi)^2

MAE:

1/n∑ i=0 to m (f(xi) - yi)

MLSE:

1/n∑ i=0 to m (ln(f(xi) - yi))^2

Regression Metrics in Python

[Topic title: Regression Metrics in Python. The presenter is Jamie Campbell.] In this video, I'll discuss how you can use Python to calculate common evaluation methods. *[The Regression metrics section is open in the scikit-learn web site.]* There are different metrics that can be used to evaluate regression models. For example, mean squared error is often used as a cost function. It describes how good a model is at making predictions with 0 being the best score. And higher scores indicating increasingly error prone models. Other measures of error include mean absolute error, for example, a non square mean error. MSC is typically preferred. It takes variance into account and it eliminates statistical bias. The mean square log error is good for the exponential growth model. And an R2 score determines how well the model can describe the data. 1 is the best score and 0 represents a model that always returns average output from a training set. That is, it always returns an expected value. If the score is less than 0, that indicates it's worse than using the expected value. Now this scikit-learn site has some useful information on regression metrics using the sklearn.metrics module. You can click on several of the functions here to learn more about the metrics and their implementation, including code samples. For example, I'll click on r2_score and that takes us to a new page where we can find out more about the parameters that can be used. *[The sklearn.metrics.r2_score web page is open.]* And it also includes code samples that you can try out.

Introduction to AWS Machine Learning

[Topic title: Introduction to AWS Machine Learning. The presenter is Jamie Campbell.] In this video, I'll introduce you to AWS Machine Learning. *[What is AWS Machine Learning?]* Amazon Web Services Machine Learning is a cloud-based service that allows you to utilize machine learning. It's useful for beginners and developers alike and uses a simple and easy-to-use API. There's no need to understand the underlying algorithms behind machine learning, or even machine learning, for that matter. With AWS machine learning, you can get started right away creating machine learning models. And, once the models are ready, you can obtain predictions. *[AWS Machine Learning Key Concepts.]* There are a few AWS machine learning key concepts you should be familiar with. Data sources refers to the metadata that's associated with data input. Machine language models are the models you set up that generate predictions using patterns that are extracted from your input data. Evaluations determine the quality of machine language models.

Batch predictions create predictions for input data. And real-time predictions generate predictions for observations based on the data. *[Accessing AWS Machine Learning.]* There are several ways to access

AWS machine learning, you can use the web-based Amazon machine language console, the AWS command line interface, the Amazon machine language application programming interface, and there are SDKs for integrating AWS machine learning with software development. *[Regions and Pricing.]* As for regions and pricing, AWS offers two real-time prediction endpoint regions for machine learning. US East, out of North Virginia, and the EU out of Ireland. To avoid any confusion, regardless of where you're based, you can work with models from any region. As for pricing, if you're familiar with the AWS model already, you know that machine learning follows the pay per use models. So you only pay for what you use. Rates are hourly, and there's no commitment, initial cost, or minimum to get started with AWS machine learning.

Setting Up an AWS Environment for Machine Learning

[Topic title: Setting Up an AWS Environment for Machine Learning. The presenter is Jamie Campbell.] In this video, I will demonstrate how to set up an AWS environment and import the data source. *[The Setting Up Amazon Machine Learning page is open in the Amazon web site.]* Now Amazon Web Services machine learning is cloud based and it allows you to utilize machine learning. It's great for beginners and developers alike and it uses a pretty easy to use API. Now there's no need to understand the underlying algorithms behind machine learning. And, with AWS machine learning, you can get started right away creating machine learning models. Now, if you're familiar with the AWS model already, you know that machine learning follows the pay per use model. So you'll only pay for what you use. But you do have to have an AWS account. So here I've got a page in AWS called Setting Up Amazon Machine Learning. And the first thing you need to do if you don't have an account is click here to set up an account. *[He points to the http://aws.amazon.com link.]* It is free, there is no charge associated with it. However, you will have to give some sort of payment information – like a credit card – in order to use it because you may choose to use services later on that are billable.

The machine learning example I'm going to show you now doesn't have a cost associated with it. So, if you want to just try it out and you don't want to pay anything, at least at this point you can try it without having to pay anything. But you will have to provide that credit card information. Now, once you're set up, you can go to this page. This is the Amazon Machine Learning get started page and click this button. *[He clicks the Get started button. The Get started with Amazon Machine Learning page is open.]* Now you have two options here. You can use the Standard setup or the Dashboard, go right to the Machine Learning Dashboard. There's also a tutorial here provided by Amazon *[He points to the Amazon Machine Learning Tutorial link.]* and I do believe there's a cost associated with this. It's up to you, but you might want to try it out, it's pretty good. But let's just go ahead and use the Standard setup and click Launch. *[The Input data tabbed page is open under the Datasources page on the Amazon web site.]* Now there are several steps here beginning with inputting data. Now the thing is, you can have your own data. But if you don't have your own data, Amazon has provided you with some data in a CSV comma separated values format that you can use, banking data. And you can take a look at the file.

I'll go ahead and click it to download it. *[He clicks the banking.csv button. A pop-up box appears and he clicks the Save button. Then he clicks the Open folder button. The Downloads file explorer window is open.]* Just double click that. *[He double-clicks the banking notepad file and the same file is open.]* So there's your comma separated banking data. You can import that into Excel if you want to, to look at it as a spreadsheet, but there's a lot of data there that you can work with. *[He closes the Notepad file.]* So go ahead and close that. *[He closes the File Explorer window.]* Now, in order to use this, you have two choices here for data access. You provide an S3 location, a path on Amazon S3, or data

source name. The way we are going to do it with this data is to just copy this file. *[He highlights the ami-sample-data/banking.csv link and right-clicks on it and select copy option.]* You don't have to copy the beginning here, S3 colon slash slash, because already provided for us here. Go ahead and paste that in. *[He right-clicks on s3:// text box and paste the link.]* And once that is done I can go ahead and click verify. Now the validation is successful, and we expect that because the data was provided by Amazon. And at this point what we can do is go to the next step by clicking Continue.

Creating a Model in AWS

[Topic title: Creating a Model in AWS. The presenter is Jamie Campbell.] In this video, I'll demonstrate how to create a model in Amazon Web Services machine learning. *[The Schema tabbed page is open under the Datasources page in the Amazon web site. The information is displayed in the tabular form. It includes various column headers such as Name, Data type, and Sample field value.]* So the first step in the process to creating a model is inputting the data. I've chosen the banking comma-separated values file that's provided by Amazon because it's a good sample file that we can use. And once you've chosen that and verified that data, the second step is choosing your schema. Now I'm going to step though this pretty quickly simply because I don't want to mess with any of this. Everything's already set up in the data file. We know it's good and solid because it came right from Amazon. But if you want to – you can review the various categories here that are in the file, change the data type, and select your schema items if you want to. Notice that down here we've got 1 to 10 of 21. So if I click there, we can see, there's the one we actually want and that's y. It's a binary value. So this is a binary schema we're going to work with.

I'll go ahead and click Continue. *[The Target tabbed page is open.]* And we've got a message here. You've selected the binary attribute named y as the target. Yeah, we're using logistic regression to train a binary classification model. So we're in the third step now, Target. And we're just getting some feedback here that we're using y. So we'll use logistic regression for training. Now at this point we can choose the target. And you've got a number of different categories you can choose here. *[He points at the Name column header.]* Again, I'm going to leave that blank and just click on Continue. *[The Row ID tabbed page is open.]* Now we have the option in Step 4 of a row identifier and that just helps understand how prediction rows correspond to observation rows from the input data. So if you want to, you can choose to select Yes. But once again, we're working with the data from Amazon. I don't want to mess with this too much because, but that's okay. *[He clicks the Review button and the Review tabbed page is open.]* We're not going to change anything here. We'll just go to step five, which is reviewing. So we review everything from the input data, the schema. We've got one binary attribute, for example. We've got ten numeric attributes. Ten categorical attributes. Our target is binary classifications, so we're just using a yes, no binary choice. No row identifier and there we go. So we can go ahead and click on Continue.

Now we go to ML model settings. *[He opens this tabbed page.]* You can use this to customize if you want to. I'll go ahead and just go down here. What we can do at this point is choose the default recommended or custom where we can modify things. We'll use the default again, we're working with the Amazon sample data and we should be good. So I'll go ahead and click Review. *[He clicks this button.]* Okay, everything's completed. Now, at this point, we can go ahead and click Create machine language model. *[He clicks this button.]* And we got some feedback that this was successfully created. Now if I go to ML models right here, *[At the top of the page, he clicks the ML models and the information is displayed in the tabular form. It includes various column headers such as Name, ID, and Status.]* okay. Well, see, there's our banking.csv. It's got an ID number and a status of pending. So

right now, it's being processed. And once that's finished processing, we'll get change in status telling us that it's ready to work with. We'll get a change in status. And it'll tell us that it's ready to work with.

Training a Model and Making Predictions in AWS

[Topic title: Training a Model and Making Predictions in AWS. The presenter is Jamie Campbell.] In this video, I'll discuss models in training in AWS. *[The AWS services page is open in the Amazon web site.]* Okay, so I've got a data set that I've actually trained. And I'm at the AWS services console and I'm going to click on Machine Learning. Now this will take me to the Machine Learning console. And you can see that there are several items here generated from Banking.csv, a comma separated values document. And this is just a test document provided by Amazon. Three of them are actual data sources. We have our machine language model and evaluation right there. *[He points to the ML model: Banking.csv and the Evaluation: ML model: Banking.csv object names.]* And there's some useful information here. For instance, the status of these is all completed. They've all been processed. And we even have the completion time for these various items. So, for example, I can click on Banking.csv and just get some information on Banking.csv. *[The Datasources page opens in the content pane.]* And even use this data source to create, train a machine language model, evaluate a machine language model, or generate batch predictions. So I can do that from here. *[He clicks the Use this datasource to drop-down button and points to the various options.]*

Now if we go back, *[He switches to the Amazon Machine Learning page.]* here is our machine language model, Banking.csv. *[He clicks this option and the ML models page opens in the content pane.]* And we can see that it's been completed, it's a binary classification. We get some information on the data source, the training, the evaluations. One has been created, we can perform another if we want to. And then predictions, we can view these in CloudWatch, the metrics that is. I'll go ahead and click that. *[He clicks the View in CloudWatch link and the Untitled graph page opens in the content pane.]* Okay, and from here we would actually pick our metrics and generate the CloudWatch graph. Let's go back here. *[He switches to the Amazon Machine Learning page.]* Sorry, this one right here. *[He switches to the ML models page.]* We can generate batch predictions, try real-time predictions, and enable real-time predictions by creating an endpoint. I'll go ahead and generate batch predictions here, see what happens, okay? *[He clicks the Generate batch predictions button and the Data for batch prediction tabbed page opens.]* So there's our batch predictions. Now if I go back here *[He switches to the Amazon Machine Learning page.]* and click on the evaluation in the dashboard. *[He clicks the Evaluation: ML model: Banking.csv object name and the Evaluation Summary subpage opens in the content pane.]* Okay, now we get our evaluation summary. This tells us what happened with the data and the idea behind this and gives us an idea of the performance metrics. So, in this case, we actually got a really good score. Extremely good, actually. And, of course, that's because we're using the data that was provided to us by Amazon.

But this gives us an idea of the overall score, the baseline, and the difference. And we can adjust the score threshold here if we want to mess around with this a bit. *[He clicks the Adjust score threshold button and the ML model performance subpage opens in the content pane. A graph is displayed with the # of Records on the y-axis and Score on the x-axis and there is a slider line in the graph.]* So if I wanted to, say, put it here *[He points to the graph displayed on the screen.]* – we can get an idea of which are correct and which are errors. And here we can see that as we move it, *[He moves the slider line.]* we've got an idea of the true positives and the true negatives and the false positives and the false negatives. So as I move it more toward predicted most likely to be 1, we see that the errors actually go up. But if I go back toward predicted most likely to be 0, you can see the errors go up there as well. So it really depends on, this middle point is at where it originally was, is actually the sweet spot. And

if I move it up here, you can see the number of errors goes up, the number of correct go down, and then the opposite is true when we go back to the center. And then if I go to the other threshold, then we see the same thing occurring.

So we can actually play around with that a bit, which is nice. *[He switches to the Datasources page.]* I can also, where'd I put that, I think it was right there. No. Now go, no, that's not it. Now go back and let's explore performance. *[He switches to the ML model performance subpage.]* So I click on Explorer Performance and we get the same thing. So we can actually mess with that as well. So really, it has essentially a similar effect...modifying, let me say that again. So basically, clicking on either of those allows us to play around with the score threshold and see what the results are for our machine learning model. So there's plenty to do here with Amazon Machine Learning. *[He switches to the Amazon Machine Learning page.]* And the really nice thing about it is that you can get this all set up fairly quickly and easily, use some existing sample data if you want to, and there's no cost to it until you actually start to do more with it. For example, there's a tutorial that you can take for setting it up and I believe there's a cost associated with that. But most of the front end of this, getting set up with Amazon Web Services Machine Learning is free. And it offers you some really powerful processing opportunities when you're training models and making predictions.

Exercise: Evaluate and Select Models

Exercise

[Topic title: Exercise: Describe Bias and Variance. The presenter is Jamie Campbell.] Now that you've learned about applying machine learning, it's time to put some of that knowledge to work. In this exercise, you'll describe bias and variance. You'll describe bias, describe variance, and describe the tradeoffs between bias and variance. At this point, you can pause this video and perform the exercise. Once you're finished, resume the video and see how I would do it.

Solution

So let's tackle these questions. It's okay if you didn't do it exactly the same way. First describe bias. Well, bias is error caused by making assumptions. It's defined by taking the difference of the expected, the average, value estimated and the actual value. And the expected estimated value is the average of values if a model is trained multiple times. High-bias models require many assumptions that is linear regression mode, Naive Bayes, and so on. And these models can miss certain relationships in the data.

Next describe variance. Well, variance is error caused by fluctuations in the data and that's commonly thought of as noise. High variance can cause models to be prone to overfitting. And training the same model on different datasets will yield different results when the variance is high. Finally, describe the tradeoffs between bias and variance. Well, in machine learning, every model has some bias and has some variance. Decreasing variance increases the bias and decreasing bias increases the variance. Some models are more prone to different types of errors. So changing model parameters can help mitigate one type of error but not both. And hybrid models can be used to mitigate both types of error. I hope you found this exercise helpful.