Embrace Randomness in Machine Learning

by **Jason Brownlee** on September 28, 2016 in **Machine Learning Algorithms**

Last Updated on August 12, 2019

Why Do You Get Different Results On Different Runs Of An Algorithm With The Same Data?

Applied machine learning is a tapestry of breakthroughs and mindset shifts.

Understanding the role of randomness in machine learning algorithms is one of those breakthroughs.

Once you get it, you will see things differently. In a whole new light. Things like choosing between one algorithm and another, hyperparameter tuning and reporting results.

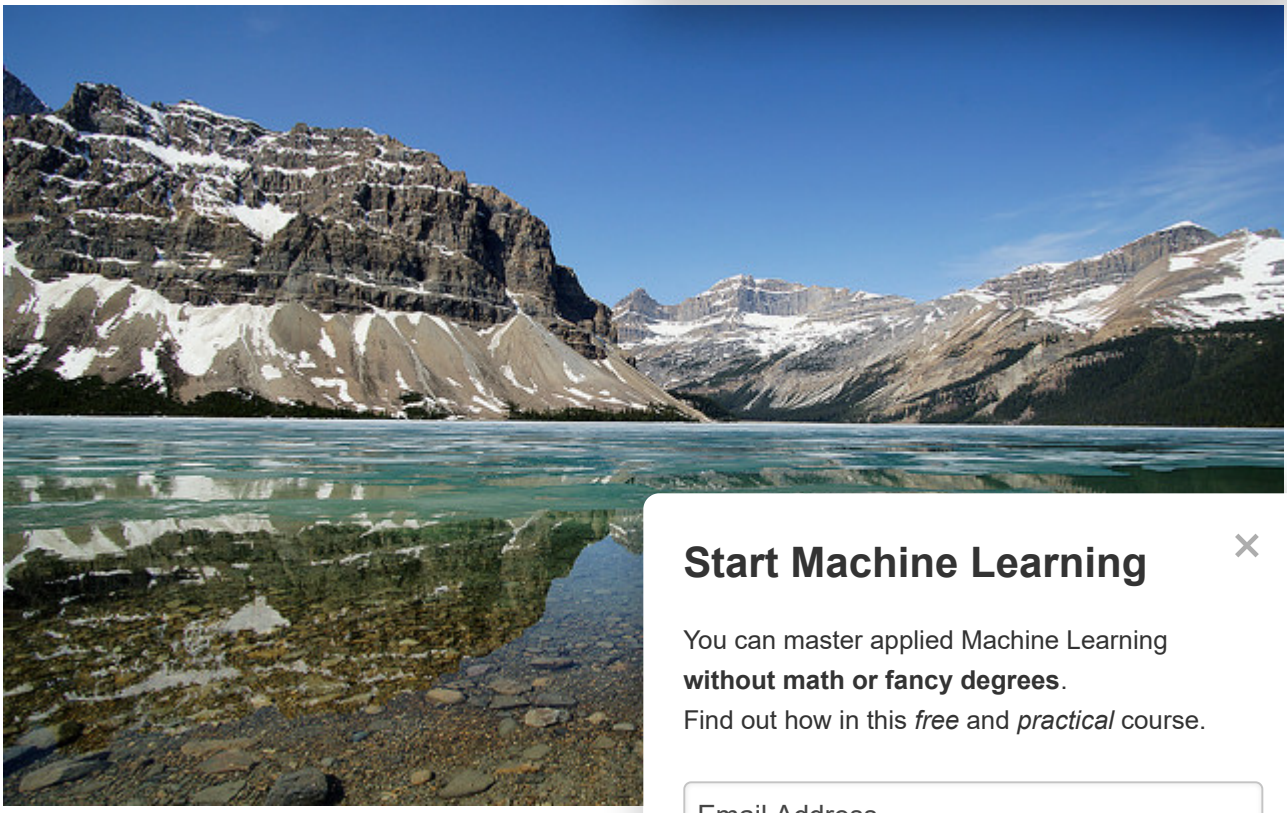
You will also start to see the abuses everywhere. The criminally unsupported performance claims.

In this post, I want to gently open your eyes to the role of random numbers in machine learning. I want to give you the tools to embrace this uncertainty. To give you a breakthrough.

Discover how machine learning algorithms work including kNN, decision trees, naive bayes, SVM, ensembles and much more in my new book, with 22 tutorials and examples in excel.

Let's dive in.

(*special thanks to Xu Zhang and Nil Fero who promoted this post*)



Embrace Randomness in
Photo by Peter Pham,

Why Are Results Different With

A lot of people ask this question or variants of this question.

You are not alone!

I get an email along these lines once per week.

Here are some similar questions posted to Q&A sites:

- Why do I get different results each time I run my algorithm?
- Cross-Validation gives different result on the same data
- Randomness in Artificial Intelligence & Machine Learning
- Why are the weights different in each running after convergence?
- Does the same neural network with the same learning data and same test data in two computers give different results?

Machine Learning Algorithms Use Random Numbers

Machine learning algorithms make use of randomness.

1. Randomness in Data Collection

Trained with different data, machine learning algorithms will construct different models. It depends on the algorithm. How different a model is with different data is called the model variance (as in the bias-variance trade off).

So, the data itself is a source of randomness. Randomness in the collection of the data.

2. Randomness in Observation Order

The order that the observations are exposed to the model affects internal decisions.

Some algorithms are especially susceptible to this, like neural networks.

It is good practice to randomly shuffle the training data [illegible] hm is not susceptible. It's a best practice.

3. Randomness in the Algorithm

Algorithms harness randomness.

An algorithm may be initialized to a random state. Suc [illegible] rk.

Votes that end in a draw (and other internal decisions) [illegible] on randomness to resolve.

4. Randomness in Sampling

We may have too much data to reasonably work with.

In which case, we may work with a random subsample to train the model.

5. Randomness in Resampling

We sample when we evaluate an algorithm.

We use techniques like splitting the data into a random training and test set or use k-fold cross validation that makes k random splits of the data.

The result is an estimate of the performance of the model (and process used to create it) on unseen data.

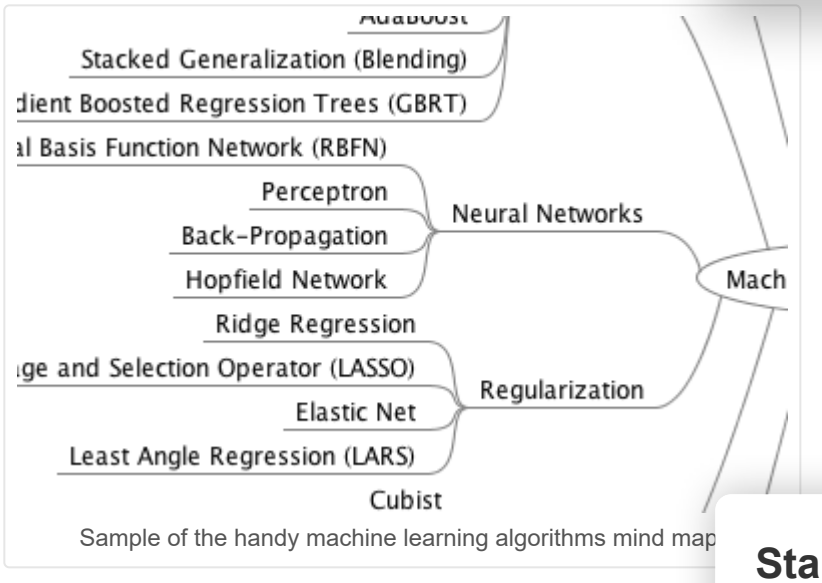
No Doubt

There's no doubt, randomness plays a big part in applied machine learning.

The randomness that we can control, should be controlled.

Get your FREE Algorithms Mind Map

I've created a handy mind map of 60+ algorithms orga



Sample of the handy machine learning algorithms mind map

Run an algorithm on a dataset and get a model.

Can you get the same model again given the same da

You should be able to. It should be a requirement that

We achieve reproducibility in applied machine learning **nce of random numbers**.

Random numbers are generated in software using a pretend random number generator. It's a simple math function that generates a sequence of numbers that are random enough for most applications.

This math function is deterministic. If it uses the same starting point called a seed number, it will give the same sequence of random numbers.

Problem solved.
Mostly.

We can get reproducible results by fixing the random number generator's seed before each model we construct.

In fact, this is a best practice.

We should be doing this if not already.

In fact, we should be giving the same sequence of random numbers to each algorithm we compare and each technique we try.

It should be a default part of each experiment we run.

Machine Learning Algorithms a

If a machine learning algorithm gives a different model with a different sequence of random numbers, then which model do we pick?

Ouch. There's the rub.

I get asked this question from time to time and I love it.

It's a sign that someone really gets to the meat of all this applied machine learning stuff – or is about to.

- Different runs of an algorithm with…
- Different random numbers give…
- Different models with…
- Different performance characteristics…

But the differences are within a range.

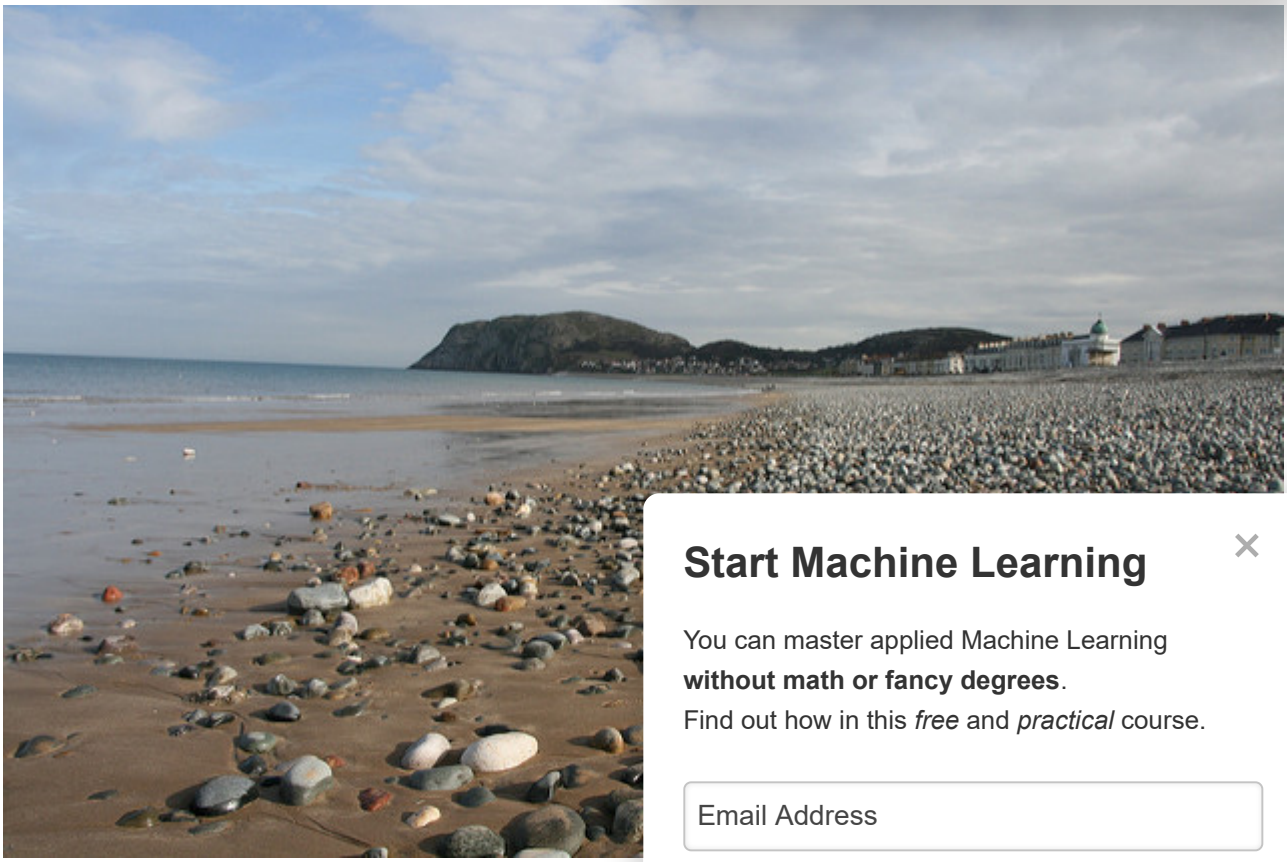
A fancy name for this difference or random behavior w

Machine learning algorithms are stochastic in practice

- Expect them to be stochastic.
- Expect there to be a range of models to choose fr
- Expect the performance to be a range and not a s

These are very real expectations that you MUST address in practice.

What tactics can you think of to address these expectations?



Machine Learning Algorith
Photo by Pete, sor

Tactics To Address The Uncertainty of Stochastic Algorithms

Thankfully, academics have been struggling with this challenge for a long time.

There are 2 simple strategies that you can use:

1. Reduce the Uncertainty.
2. Report the Uncertainty.

Tactics to Reduce the Uncertainty

If we get different models essentially every time we run an algorithm, what can we do?

How about we try running the algorithm many times and gather a population of performance measures.

We already do this if we use *k*-fold cross validation. We build *k* different models.

We can increase *k* and build even more models, as long as the data within each fold remains representative of the problem.

We can also repeat our evaluation process *n* times to get even more numbers in our population of performance measures.

This tactic is called random repeats or random restarts.

It is more prevalent with stochastic optimization and neural networks, but is just as relevant generally. Try it.

Tactics to Report the Uncertainty

Never report the performance of your machine learning algorithm with a single number.

If you do, you've most likely made an error.

You have gathered a population of performance measures. Use statistics on this population.

This tactic is called report summary statistics

The distribution of results is most likely a Gaussian, so
standard deviation of performance. Include the highes

In fact, this is a best practice.

You can then compare populations of result measures

- Choosing between algorithms.
- Choosing between configurations for one algorith

You can see that this has important implications on the
algorithm to use on your problem and for tuning and choosing algorithm hyperparameters.

Lean on statistical significance tests. Statistical tests can determine if the difference between one population of result measures is significantly different from a second population of results.

Report the significance as well.

This too is a best practice, that sadly does not have enough adoption.

Wait, What About Final Model Selection

The final model is the one prepared on the entire training dataset, once we have chosen an algorithm and configuration.

It's the model we intend to use to make predictions or deploy into operations.

We also get a different final model with different sequences of random numbers.

I've had some students ask:

> *Should I create many final models and select the one with the best accuracy on a hold out validation dataset.*

"*No*" I replied.

This would be a fragile process, highly dependent on the quality of the held out validation dataset. You are selecting random numbers that optimize for a small sample of data.

Sounds like a recipe for overfitting.

In general, I would rely on the confidence gained from the above tactics on reducing and reporting uncertainty. Often I just take the first model, it's just as good as any other.

Sometimes your application domain makes you care more.

In this situation, I would tell you to build an ensemble
number seed.

Use a simple voting ensemble. Each model makes a p
as the final prediction.

Make the ensemble as big as you need to. I think 10,

Maybe keep adding new models until the predictions b
variance of the predictions tightens up on some holdo

Summary

In this post, you discovered why random numbers are integral to applied machine learning. You can't really escape them.

You learned about tactics that you can use to ensure that your results are reproducible.

You learned about techniques that you can use to embrace the stochastic nature of machine learning algorithms when selecting models and reporting results.

For more information on the importance of reproducible results in machine learning and techniques that you can use, see the post:

- Reproducible Machine Learning Results By Default

Do you have any questions about random numbers in machine learning or about this post?

Ask your question in the comments and I will do my best to answer.

About Jason Brownlee

Jason Brownlee, PhD is a machine learnin th modern machine learning methods via han

View all posts by Jason Brownlee →

‹ Python Machine Learning Mini-Course

Deploy Your Predictive Model To Production ›

62 Responses to *Embrace Randomness in Machine Learning*

Cameron DP September 28, 2016 at 6:09 am #

REPLY

A question that I've faced is whether to use a fixed seed in a production algorithm. If the results are customer facing, is there any harm in using a seed so the results are consistent? (Maybe not consistent, but the variance is atleast reduced)

Jason Brownlee September 28, 2016 at 7:43 am #

REPLY

Indeed Cameron, a tough one.

I have used fixed seeds in production before. I want variability to come from the data.

Today, I might make a different call. I might deploy an ensemble of the same model and drive stability via consensus.

Peter P September 7, 2017 at 1:41 am #

REPLY

Hello Dr. Brownlee,

I kind of get the idea of creating an ensemble of the same model, but how do you implement the voting process?

Jason Brownlee September 7, 201

Count the predictions for each cla
implementation like that in Weka or sklearn

Alan Beckles MD MS October 4, 2016 at 4:41 pm

This is an excellent thought provoking post. Best practice may be your recommendation to build an ensemble of models using different random seeds, then use voting.

Jason Brownlee October 5, 2016 at 8:24 am #

REPLY

Thanks Alan, I'm glad you found it useful.

Rajesh October 7, 2016 at 4:48 am #

REPLY

Really amazing blog it is a lot we can learn here, still we have to agree that handling data is always a tricky and mystery.

Wes Turner January 14, 2017 at 6:22 pm #

REPLY

In addition to specific libraries' function parameters like 'seed', Python, for example, reads 0 or a decimal value from the PYTHONHASHSEED environment variable.

From "Ten Simple Rules for Reproducible Computation
http://journals.plos.org/ploscompbiol/article?id=10.137

Start Machine Learning

> Rule 6: For Analyses That Include Randomness, Note Underlying Random Seeds

Are random seed(s) are just another parameter to optimize? (I tend to agree with your ensembling recommendation)

Jason Brownlee January 15, 2017 at 5:28 am #

REPLY

Thanks Wes.

Sophia August 2, 2017 at 12:53 pm #

Sometimes, the training results are all 0 and t
a line vs the iterations. But if I train it again without any
problem? And how can I avoide the failed training? Th

Jason Brownlee August 3, 2017 at 6:43 am #

Consider repeating the experiment multip
model performance.

For example, see this post:
http://machinelearningmastery.com/evaluate-skill-deep-learning-models/

Sharps September 14, 2017 at 9:06 am #

REPLY

Hello Jason!
Thanks for you're all posts/analysis/documentations/answers!!
I roam in NNs coming from sql dev and i see a new dimension now! ☺

Jason Brownlee September 15, 2017 at 12:09 pm #

REPLY

I'm glad they helped.

Nil December 11, 2017 at 11:00 pm #

REPLY

Hi, Dr. Jason,

Thank you, this post helped me so much, to handle wit

Your posts a really good and helpfully.

Best Regards.

REPLY

Jason Brownlee December 12, 2017 at 5:33 am #

I'm glad to hear that.

REPLY

Dinto January 2, 2018 at 10:40 am #

Do you suggest an ensemble of models also f

Jason Brownlee January 2, 2018 at 3:59 pm

For sure.

Varun February 7, 2018 at 11:47 pm #

Hi Jason,

Thanks for all your valuable information.

I have a specific query related to the Neural Network and its implementation. In a typical setup, I try to tune parameters like :

• Methodology (Feed Forward, back propagation) based on multiple decisive weights
• Lag values
• # of Nodes (Input , output , hidden)
• # of Neural layers (Input, output, hidden)
• Key Activation Function
• # of Iterations based on above

Again, my problem is that I need to forecast profit for ~50,000 customers and the input data (~3 years time series) trend is mostly non-linear, sporadic with no specific seasonal pattern and trend. My query:

1). How should I control the above parameters and train the model for all individual customers?

2). I would have to run this model every month for all customers. Should I fix the neural network model once I have trained it and run the same model each month? The problem is that the input weights change in each run (I am doing it in SAS)

3). Can I fix the other parameters (except weig

to production? I am really apprehensive of the

month.

Appreciate your reply!

Regards,
Varun

Jason Brownlee February 8, 2018 at 8:27 am #

REPLY

That is a lot of questions. In general, I have advice for lifting model skill here:
http://machinelearningmastery.com/improve-deep-learning-performance/

Varun February 8, 2018 at 10:13 p

Hi Jason,

Okay. I'll narrow it down.How do I batc
have 1 model for entire base as the ur

Again, your reply would be crucial for r
performance as mentioned in the abov

Thanks a lot!

Regards,
Varun

Jason Brownlee February 9, 2018 at 9:08 am #

There are many ways to model your problem.

You could have one model per customer, one for groups of customers or one for all customers, or all 3 ensembled togehter. I recommend testing a few approaches and see what works best for your specific data.

Varun February 9, 2018 at 6:15 pm #

REPLY

Yep. Started. Will take time but I'll let you know how it panned out.

Thanks for your wonderful posts. You are doing a great job!

Jason Brownlee February 10, 2018 at 8:53 a

Thanks.

Ben June 3, 2018 at 11:18 am #

REPLY

Hi Jason do you have any write ups for reporting uncertainty from a statistical stand point in results?

Jason Brownlee June 4, 2018 at 6:20 am #

REPLY

This method for testing algorithms might h
https://machinelearningmastery.com/evaluate-skill-

And this post on confidence intervals for summariz
https://machinelearningmastery.com/confidence-in

Kingsley Udeh July 6, 2018 at 3:15 am #

Hi Jason,

I created my final LSTM model and made predictions with it. Those predictions were fair or above average. When I increased the epochs, in a bit to improve the model's performance, and retrained the model, that is, created another final model, I got different results, worse than the former. Could the different results have anything to do with stochastic nature of the model? If so, how could I overcome the variances?

P.S. I'm working on sequence prediction probe, specifically, time series. I'm wondering if k-fold cross validation resampling method would more effective than train\test split, but it seems one your responses to answers on your blog says that k-fold cross validation resampling method isn't ideal for time series problem or LSTM.

Jason Brownlee July 6, 2018 at 6:47 am #

REPLY

Good question.

I often recommend fitting 5-10 final models and using them in an ensemble to reduce the variance of predictions. I've also seen good papers on checkpoint ensembles.

I have a post scheduled on this topic.

No, I recommend walk-forward validation for evaluating models on time series:
https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/

Kingsley Udeh July 6, 2018 at 11:39 am #

REPLY

Thank you

Kingsley July 6, 2018 at 7:24 pm #

REPLY

Hi Jason,
I appreciate the fact that you are taking time out of your busy schedules to address questions. Thank you, once again.

I have some issues bothering me on creating final mod

1. I understand that to overcome different prediction ou
data and network architecture, I should take an averag

```
# make a prediction
yhat1 = model.predict(test_X)
yhat2 = model.predict(test_X)
yhat3 = model.predict(test_X)
yhat4 = model.predict(test_X)
yhat5 = model.predict(test_X)
```

yave = (yhat1 + yhat2 + yhat3 + yhat4 + yhat5)/5 – Per
averaging perfectly.

Now, I can compare average predictions with true observations, right?

2. In the creation of the final model you demonstrated, the whole data is used to train the model, such as;

model.fit(X, y, epochs=100, shuffle=False, verbose=0)

Where X is the input sequence, and y, the output sequence, since there is no concept of train/test splits, or k-fold cross validation. Why did we still make predictions on the same X that have been used to fit the final model when we load the model for prediction? The true observations, y, have also been exposed to the model. I must have missed out in the tutorial.

3. Honestly, I have gone through the link you provided for Walk-forward validation, but I haven't yet grasped the concept. Does it mean that in Walk-forward Validation method, we don't have to create a final model or find the average of the models? That is, all we care about is the model that performs best amongst the models, and such is saved as a final model, right? If this is the case, in my situation, I have over 14,000 observations and I'm wondering How I would create 14,000 models and chose the best from them.

Thanks in advance.

Jason Brownlee July 7, 2018 at 6:15 am #

REPLY

Hi Kingsley,

Start Machine Learning

Yes, using an ensemble of models is a good way to overcome high variance in some models.

It was just an example in the tutorial, you can make predictions on new data.

Walk-forward validation is for estimating the skill of the model on new data. We must still create a final model.

Does that help?

REPLY

Neel June 2, 2019 at 9:54 pm #

Dr Jason, when you say we must still create a final model, what do you mean? When we are using a Walk Forward validation, wouldn't t
final model?

Jason Brownlee June 3, 2019 at 6:

The last model is still making pred

A final model is fit on all available data, mo
https://machinelearningmastery.com/train-f

REPLY

Olivier Blais July 10, 2018 at 5:02 am #

Great blog post as always!

I have deployed a gradient boosting model in prod, using joblib to persist it. I was surprised to see slightly different results in prod vs. in dev. Is this because of randomness or it is a problem with my script? I thought that persistence removed randomness.

Please let me know ☺
Thanks

REPLY

Jason Brownlee July 10, 2018 at 6:53 am #

It may be, you should be able to narrow dow n the cause.

REPLY

Anam Habib August 28, 2018 at 2:23 pm #

Hy Jason,
Can random be used as a hyperparameter? Or can i p

increase the accuracy of the model? Is it valid?

Jason Brownlee August 29, 2018 at 8:01 am #

REPLY

No, the seed is not a hyperparameter.

Anam Habib August 29, 2018 at 2:48 pm #

REPLY

Thanx Jason for your response

era November 4, 2018 at 1:10 am #

i have my dataset from a company that sells p
make the best analysing of this data.
thank you

Jason Brownlee November 4, 2018 at 6:28 a

Maybe this process will help:
https://machinelearningmastery.com/start-here/#process

Gena November 6, 2018 at 2:37 am #

REPLY

Hi, thank you for your blog posts, they're really helpful.
Concerning randomness, I still have a hard time understanding why I get different models even though I: 1) set the same seed 2) use exactly the same data to train and test. Yet the accuracy reported on the test set is always somewhat different.

The only explanation I can think of is that using numpy.random.seed() is not the only seed that tensorflow and keras might be using (e.g. for random parameter initialization). Is this true? Can I make another few bla-bla-bla.seed() calls to make it totally reproducible?

Jason Brownlee November 6, 2018 at 6:34 am #

REPLY

It is hard to seed Keras models as a number of different pseudorandom number generators are used.

More details here:
https://machinelearningmastery.com/reproducible-results-neural-networks-keras/

Abhishek G.L November 13, 2018 at 1:19 pm #

REPLY

Hey Jason Brownlee, your tutorials are really very helpful and easy to understand, Cheers to you.

My question is about statistical significance test, is it valid to perform statistical significance on two different variety of systems. example: Bayesian optimization and genetic algorithms (autosklearn and tpot use these methods) a statistical significance test between the final models which these two libraries produce, can the significance yield any useful information

Jason Brownlee November 13, 2018 at 1:44

Perhaps. The specifics matter.

A good place to start is here:
https://machinelearningmastery.com/statistical-sign
algorithms/

Abhishek Govindappa lakshminarayan November 15, 2018 at 4:06 am #

REPLY

How viable is performing a statistical significance test between two different systems generally speaking?

Jason Brownlee November 15, 2018 at 5:36 am #

REPLY

Depends.

Neel Bhalaria June 2, 2019 at 9:44 pm #

REPLY

Hi Jason!

Excelled article, as always! I have learned a lot from your tutorials. Your reply to my comments have been extremely helpful in advancing of the project. Thanks again!

I am building a LSTM model with Time Series Classification (Cross Entropy) using the Walk Forward Approach based on your tutorials.

For predictions, I am extracting values from classification matrix & confusion matrix to compute further a custom matrix (CM) which would tell me the accuracy levels in terms of an integer value. I am using this value to evaluate my model selection

I am currently in the process of tuning my hyper-parameters (Neurons, L2 rate, Dropout, etc.).

1) Is it okay if I create a random seed based on your article for ALL my models with Different hyperparameters and then compare them based on my custom matrix (CM) ? My thought was that then I would be able to compare them on a common scale. Would this work based on your experience?

```
from numpy.random import seed
seed(1)
from tensorflow import set_random_seed
set_random_seed(2)
```

2) From a couple of articles that you have posted, you
and then take a mean of all the values and check the v

2a) Take a hyperparameter, train the algo 10/20/30 tim
then for that hyperparameter considered the mean of a
compare the hyperparameter individual mean values fo

2b) Run all hyperparameters on a common seed and s
my custom metric. Take that model and run it (10/20/30
I need to account for when I make the model live.

Thanks!

Jason Brownlee June 3, 2019 at 6:41 am #

REPLY

No, I recommend using repeated evaluation, described here:
https://machinelearningmastery.com/evaluate-skill-deep-learning-models/

Neel June 3, 2019 at 10:27 pm #

REPLY

From what I understood from the article is that in “repeated evaluation” we split the data randomly and fit it to the model and check the scores. Correct?

If yes, wouldn’t this replace the Walk Forward Validation technique for model training?

Jason Brownlee June 4, 2019 at 7:52 am #

REPLY

Yes, for time series one should use repeated evaluation of walk-forward validation, not random splits.

Neel June 4, 2019 at 4:23 pm #

Understood. This helps a lot! Thank you ☺

Neel June 3, 2019 at 10:40 pm #

REPLY

Or are you suggesting that I use the Walk Forward Validation technique and run the experiments (10/30/100) times with no seed to check for interval, lower and upper for each class (I have 3) and consider those values as the model's performance for that particular set of parameters?

chirag August 15, 2019 at 6:28 pm #

why the score value is comes out to be negati
keras.wrapper.scikit_learn import KerasRegressor ?

Jason Brownlee August 16, 2019 at 7:49 am

sklearn makes mse negative for optimizat

More here:
https://machinelearningmastery.com/faq/single-faq/why-are-some-scores-like-mse-negative-in-scikit-learn

chirag August 17, 2019 at 12:39 am #

REPLY

ignoring is ok but why scikit-learn api gives that negative sign. What it's significance.

Jason Brownlee August 17, 2019 at 5:49 am #

REPLY

It inverts the sign so that all optimization problems are maximizing, e.g. accuracy->100, neg_mse -> 0, etc.

chirag August 25, 2019 at 12:55 pm #

Thank you

Prajakta September 17, 2019 at 8:26 pm #

REPLY

Unable to download ML map.
Fulfilled all requirements still it's not working ????

Jason Brownlee September 18, 2019 at 6:06 am #

REPLY

It can take a few minutes.

Checking my system, I can see that you received t

Yamini October 18, 2019 at 3:59 pm #

What is the difference btw multiple-restart sea
https://machinelearningmastery.com/why-initialize-a-ne
models? Or are they two terms for the same thing?

Jason Brownlee October 19, 2019 at 6:28 am #

They are different.

Multiple restarts is one model fit repeatedly over time with the goal of getting the best single model.

Ensembles is fits multiple models, keeps them all and combines their predictions.

Omar January 11, 2020 at 10:04 am #

REPLY

Greetings Jason,

I'm not sure about "Reducing the Uncertainty".

If I run the model many times to get the mean and standard deviation from the metrics, do I have to set the random seed or not?

Thanks

Jason Brownlee January 12, 2020 at 7:58 am #

REPLY

You do not.

Leave a Reply

Name (required)

Email (will not be published) (requ

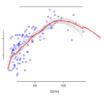
Website

SUBMIT COMMENT

Welcome!
My name is *Jason Brownlee* PhD, and I **help developers** get results with **machine learning**.
Read more

Never miss a tutorial:

Picked for you:



A Tour of Machine Learning Algorithms

Supervised and Unsupervised Machine Learning Algorithms


Logistic Regression Tutorial for Machine Learning


Simple Linear Regression Tutorial for Machine Learning


Bagging and Random Forest Ensemble Algorithms for Machine Learning

Loving the

The Machine Learnir
is where I keep the

SEE WHAT

Start Machine Learning