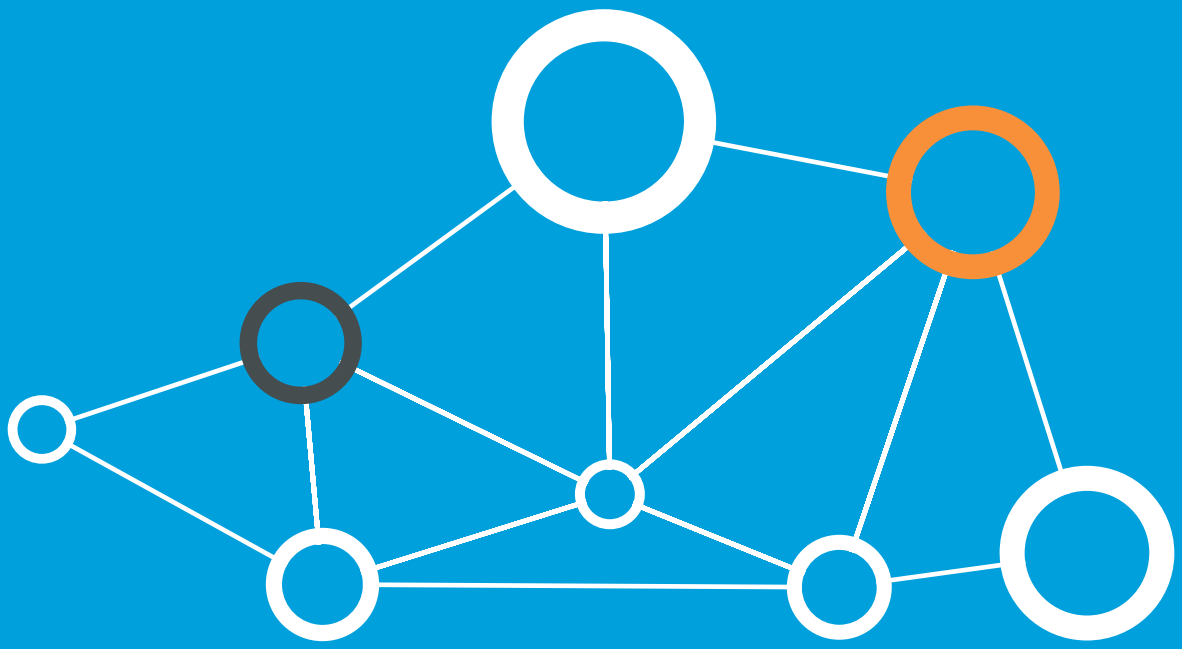
ORACLE® + DATASCIENCE.COM

Testing Predictive Models in Production

Ruslana Dalinina, Jean-René Gauthier, and Pramit Choudhary



Introduction

When a business project requires the development of a predictive model, a data scientist will go through steps of feature engineering and selection, methods comparison, model training, and model deployment (see figure 1). Model deployment means that model predictions are being consumed by an application that is directly affecting business operations. One way to deploy a model is to create a REST API endpoint for that model. A business application can then call that endpoint to score new customers, for example.

Predictive models are usually trained on a historical dataset reflecting the state of the business at some point in time. The model or models showing the best performance on a holdout/validation dataset are presumably deployed. Although this seems like a sensible approach, the assumption here is that the deployed models will replicate their performance characteristics post-deployment on new, unseen data.

In this white paper, we discuss why this may not be case.[1] Moreover, we go over some best practices on how to identify the best performing models post-deployment using a standard A/B testing approach. We refer to this process as model testing.

This white paper is organized as follows: In the first section, we explain some of the root causes of the discrepancies between training and deployed model performances. We then introduce model testing as a way to identify the best deployed models, focusing on A/B testings and bandit approaches. We describe each technique along with its associated pitfalls and limitations. Throughout this white paper, we refer to model testing as testing done on deployed models scoring new, unseen data.

[1] There is also something to be said about having a series of trained models with similar performance metrics (e.g., 0.792, 0.795, 0.789, 0.805). Which one(s) would you deploy? Are the performance metrics statistically different from each other? In practice, you may want to deploy all of them. Indeed, these models could give you very similar performances in a production setting (or very different ones too).

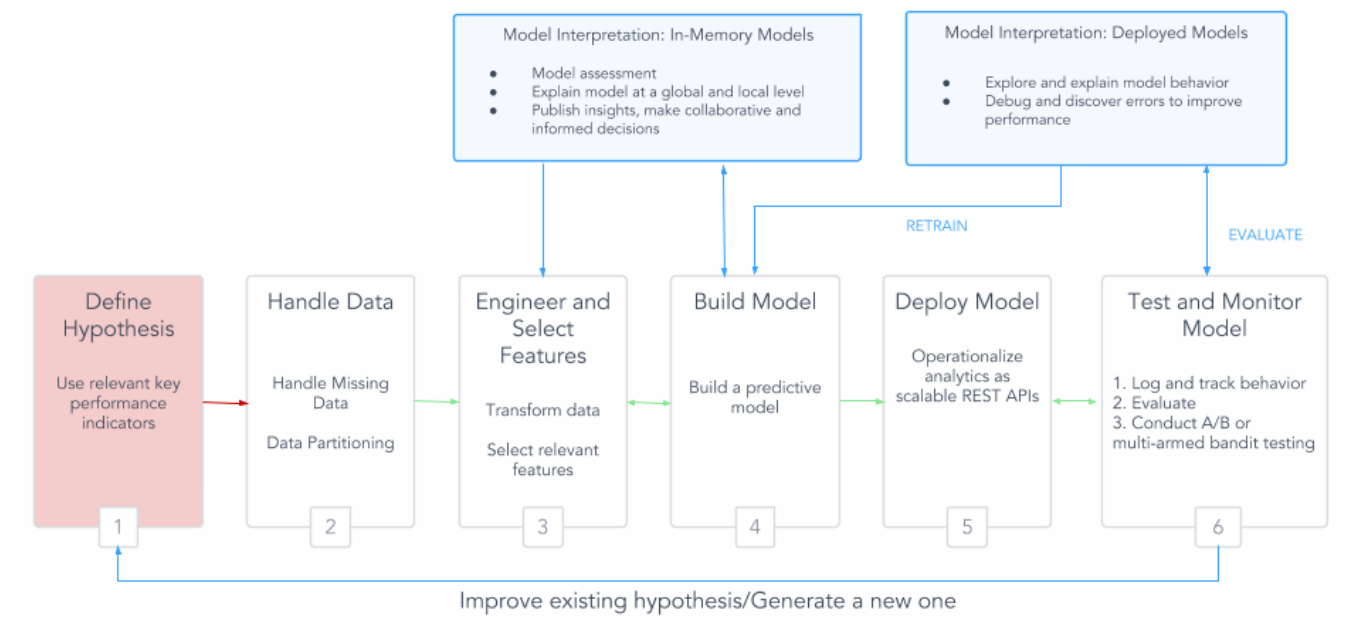


Figure 1. Illustration of a predictive model lifecycle. An analyst/data scientist will go through several steps before deploying a model, including defining a hypothesis and associated performance metrics, data cleanup, feature engineering and selection, and model building. In the post-deployment steps, models should be monitored, tested, and interpreted. In this white paper, we focus on testing models in Step 6.

The Motivation for Model Testing

In an idealistic setting, the behaviors of models trained on historical data would be reproduced on new, unseen data in the post-deployment phase. In practice, this is rarely the case. A model's performance post deployment is often different from its performance with a holdout dataset during the training phase. Let us explore some of the common causes of this discrepancy.

The Predictors are Changing

The distributions of input features can change over time and result in unexpected patterns not present in historical datasets. New, unseen feature levels might appear or data collection procedure may lead to a shift in distribution. For example, your business may use new acquisition channels not seen before by your click-through-rate model, or your model may be better at predicting churn for male customers, thus lowering the performance of your deployed model during a promotional campaign to attract female customers.

Models trained on historical data may deliver worse results under these new conditions. Model testing on live data in your production environment, however, can help you assess whether or not the model performs as expected.

Performance Metrics May Differ

During training, it is likely that the model was optimized on a standard machine learning performance measure, such as AUC, F1 score, or RMSE. In contrast, the model may be deployed to help improve a business key performance indicator (KPI) such as reducing customer churn or increasing conversions. In an ideal world, these business metrics would be optimized during the model training step, but that is often difficult to do.

Estimating the Impact of Model Implementation

Finally, business projects are complex. What might seem like an isolated and well-contained project may influence many other parts of the system in unforeseen ways. For example, user actions produced by a recommender engine might influence the predictions of a ranking model that displays products on the main website page. Your historical dataset may not capture these complex interactions.

Given these concerns, a data scientist should approach model deployment cautiously. A good strategy is to deploy a subset of the best models trained on a historical dataset.

Model Testing: An Overview

As we mentioned above, deploying models to production and letting them run is not the end of the story. It should be the beginning of a careful testing process that addresses the following questions:

1. Does your model behave as expected?
2. Is the best trained model indeed the best model, or does a different model perform better on new, unseen data?

Broadly speaking, model testing is a set of procedures that answers these questions. The idea is to provide a framework to identify the best performers among a competing set of models. In turn, this framework allows analysts to favor high-performing models that maximize business KPIs; for example, directing web traffic to the most accurate click-through-rate model endpoint.

Model testing can be accomplished with techniques similar to A/B testing or bandit algorithms. These methods have been extensively used for website optimization, but in this context, you would compare the different models making predictions and their respective performance metrics rather than A/B variants of a website feature.

Hereafter, we will refer to the process of setting up and performing an A/B test over a set of deployed models as an "experiment."

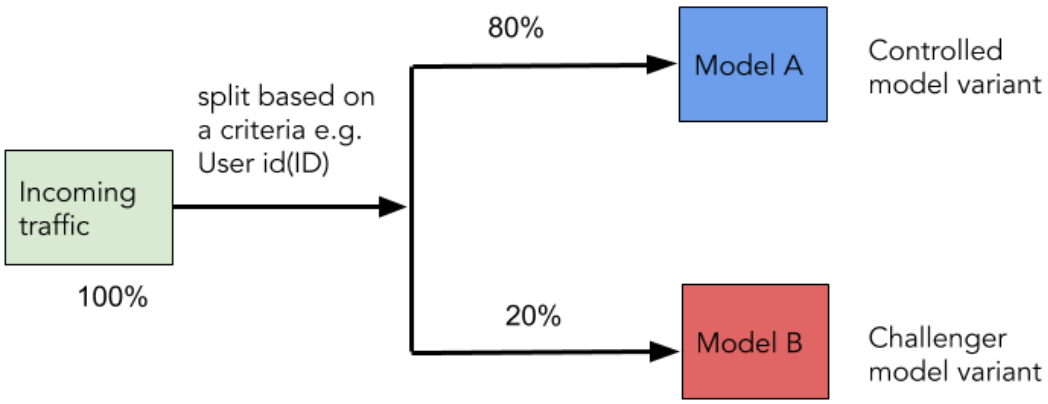
Other Considerations

In any experiment where more than one model is deployed, some portion of your test units will potentially receive suboptimal treatment. If your models directly impact users, the focus should be on not hurting the user experience or the bottom line. Think about costs and benefits associated with each experiment.

Key Takeaways

- The relevant performance metrics of a model may differ offline versus online. Historical data may not capture complex interdependencies between different model predictions.
- Choosing a conservative approach and deploying multiple models is a good way to minimize the risk of using a single, potentially suboptimal model.
- Focus on experiments that are valuable and do not damage the user experience.

A/B Model Testing



As a data scientist brings deployed models to production, he or she might want to compare the performance of a currently deployed model (let's call the current champion Model A) to that of a new version (Model B, the challenger model). Alternatively, he or she may want to simultaneously deploy and compare multiple models with similar training performances. When the number of models to be compared is more than one, in which case Model A could be compared to Models B, C, D, … N, an A/B test becomes an A/B/C/D/.../N test.

After the key performance metric is chosen (e.g., model accuracy, RMSE, mean revenue per user, etc.), the data scientist can apply statistical inference to evaluate performance. Typically, statistical hypothesis testing involves setting up a null hypothesis (e.g., there is no significant performance difference between Model B and Model A) and an alternative hypothesis (e.g., Model B is performing better than Model A).

The experiment is then run for a fixed period of time, determined by a sample size threshold. During the experiment, a portion of data is exposed to Model B ("treatment"), and comparable data is exposed to Model A ("control"). The resulting performance metrics are compared using statistical tests and the null hypothesis is either rejected or retained.

A/B testing has been written about extensively and is widely used in website optimization. Despite the wealth of information on A/B tests, it is still tricky to get right. In the next section, we provide a short guide to implementing your A/B/.../N test correctly. First, we will define some of the associated terminology.

Basic Terminology

Statistical Power (Sensitivity): The probability of correctly rejecting the null hypothesis, i.e., correctly identifying an effect when there is one. This helps determine the sample size used for the experiment and identify significant difference when it exists. Often, the desired power of an experiment is between 80-95% (see Kohavi et al. 2009).

Confidence Level: The probability of correctly retaining the null hypothesis when there is no difference in effects. Often, confidence level is set to 95%. The statistical significance of the test (α) corresponds to 1 - Confidence Level. For a confidence level of 0.95, α=0.05.

Effect Size: The difference between the two models' performance metrics, often normalized by standard deviation or by a baseline value if the effect size is a relative one.

A/A Testing

The idea behind A/A testing is simple: design your experiment, split data into two halves, and use the same model for both halves. Since both halves are exposed to the same model, we would expect to see no statistically significant differences between the two models at the confidence level chosen (see section on Determining Sample Size). While this may seem excessive, investing in A/A testing will have huge payoff in the long term.

In fact, A/A testing is a good way to validate the experiment design setup. If the experiment is designed and randomized correctly, there should be no difference in model performance. The likelihood that you would see, by chance, a statistically significant difference between the two models is 5% if the confidence level is set at 95%. A/A tests can help expose any data collection issues present, or whether the two data halves differ in some ways and are therefore not properly randomized.

Moreover, A/A tests can be used to estimate the variability of your model's performance metric or to check for pre-existing differences in the experimental setup. Often, true population variance is unknown, but a variance estimate is needed for sample size and confidence interval calculations. Running an A/A test will provide real data and a way to estimate population variance.

It is highly recommended to run A/A test before any A/B test. Keep in mind, the analyst should ensure that test units in group A are not selected differently from units in group B.

Designing a Model A/B Test

At a high level, designing an A/B test for models involves the following steps:

- Deciding on a performance metric. It could be the same as the one used during the model training phase (e.g., F1, AUC, RMSE, etc.).
- Deciding on test type based on your performance metric.
- Choosing a minimum effect size you want to detect.
- Determining the sample size N, based on your choice of selected minimum effect size, significance level, power, and computed/estimated sample variance.
- Running the test until N test units are collected.

In the following subsections, we discuss the choice of sample size, test type, and effect size.

Determining Sample Size and Early Stopping

One of the most common mistakes in A/B testing is declaring success when statistical significance is first reached and before N units have been collected. Most automated A/B testing software will start to indicate significance as soon as it is able to, and it is very tempting to get excited and declare that the new model is a resounding success.

In fact, most statistical tests rely on the fact that N is fixed and known before the experiment begins. Stopping the experiment before N is reached simply does not produce reliable results. It's equivalent to preferentially selecting outcomes that are significant only by chance. It is a best practice to make sure you know N before running your test. In table 1, we showcase a scenario that illustrates the importance of reaching N before stopping.

Table 1. Here's a simple thought experiment: You are running four independent A/B tests where both A and B are the same models. If you pick a significance level α=0.05, you expect to see significant results (Sig) in one of 20 independent tests **for a fixed and identical N (N=1000)**. If you stop as soon as significance is reached, you preferentially select spurious false positives. In the example in the chart, you stopped early for two of the four tests and ignored all results that occured after that spurious detection. Only one of the four tests is actually significant when N=1000, but you won't know that if you stop early – instead, you will believe that three of the four tests are significant.

Test	Samples Collected				
	200	400	600	800	1000
Test 1	Not Sig	Sig - STOP	~~Not Sig~~	~~Not Sig~~	~~Not Sig~~
Test 2	Sig - STOP	~~Not Sig~~	~~Not Sig~~	~~Not Sig~~	~~Not Sig~~
Test 3	Not Sig	Not Sig	Not Sig	Not Sig	Not Sig
Test 4	Not Sig	Not Sig	Not Sig	Not Sig	**Sig**

Desired sample size is based on several quantities: effect size, standard deviation, power, and confidence level. Effect size indicates the desired difference in selected performance metrics or evaluation criteria we wish to detect during the experiment. See List, Sadoff & Wagner (2011) for more details and examples of sample size calculations.

Combined, these quantities calculate the sample size needed to detect an effect size given variability in the data, some minimum probability of detecting true differences (power), and the retention of the null hypothesis (confidence level). See Kohavi et al. (2009) for more details on sample size formula calculations.

Is It Worth the Implementation Cost?

Is the effect size worth the potential costs of implementation? For example, will a 3% accuracy improvement be beneficial to your business? What if this slightly better model requires a much larger computational infrastructure to support it in the long term? Confidence intervals for effect size and cost-benefit analyses are both great tools for determining whether to implement and deploy a new model.

Novelty and Primacy Effects

In addition to fixed sample size, it is not uncommon to see what are known as novelty (an initial positive effect that regresses) and primacy (a positive effect happens over a period of time) effects (Kohavi et al. 2012). These phenomena show larger effect size during the beginning of an experiment, or even an effect in the opposite direction than the long term effect. This can be explained by the users' responses to the new experience provided by the model. For example, when encountering a new product recommendation model, some users might simply be curious about the new recommendations and click or even buy some of them (novelty effect). However, after a while, if the model is not truly improving the recommendations this effect will wear off. On the other hand, some users might not like the new recommendation experience in the beginning but get used to it and start responding positively after some time (primacy effect).

A simple solution for controlling these early effects is to apply the new model/algorithm to only new users (when applicable) who do not have set expectations yet. Alternatively, you could run the test longer and use only the data after the effect size seems to have stabilized.

Finally, make sure to run the test long enough to capture any seasonality effects. For example, if you suspect there is a day-of-the-week effect, make sure to run the test for at least a couple of weeks.

Selecting a Relevant Statistical Test

The process of deciding on a test type to assess your model will largely be driven by the type of data and metric being tested. Most model performance metrics can be categorized as continuous or discrete.

Continuous metrics are relevant to regression problems. Each user/test unit has a difference between observation and prediction that can take on a wide range of values. These metrics, such as RMSE, can be compared using tests such as t-test or a Wilcoxon rank-sum test.

Discrete metrics are usually relevant to classification problems. For each user/test unit, a "success" is defined as when the observed class is equal to predicted class. The proportion of successes in the treatment and control groups can be compared using tests like chi-squared or z-test for proportions.

Table 2. Statistical Tests

Type of Problem	Example Metrics	Metric Type	Metric Example Data	Statistical Test
Regression	RMSE, Absolute % error, Percent error	Continuous	Treatment: [2.1, 2.3, 3.4, …] Control: [0.8, 1.7 3.4, …]	Parametric tests, e.g., t-test (1-sample, 2-sample, paired), z-test, F-test Non-parametric tests, e.g., Wilcoxon-Mann-Whitney test
Classification	Accuracy, Precision, Recall	Discrete	Treatment: [0,1,0,1] Control: [0,0,1,1]	Difference in proportion test Nonparametric tests, e.g., chi squared, etc.

One- vs Two-tailed Tests

In addition to metric type, you also need to decide on whether to use a one- or two-tailed test.

In a one-tailed test, you are only testing whether there is an effect in one, usually positive, direction. One-tailed tests tend to achieve significance faster and have more power, which is why it is often tempting to default to a one-tailed test. However, by doing this, you are ignoring the whole other side, i.e., testing whether there a negative effect.

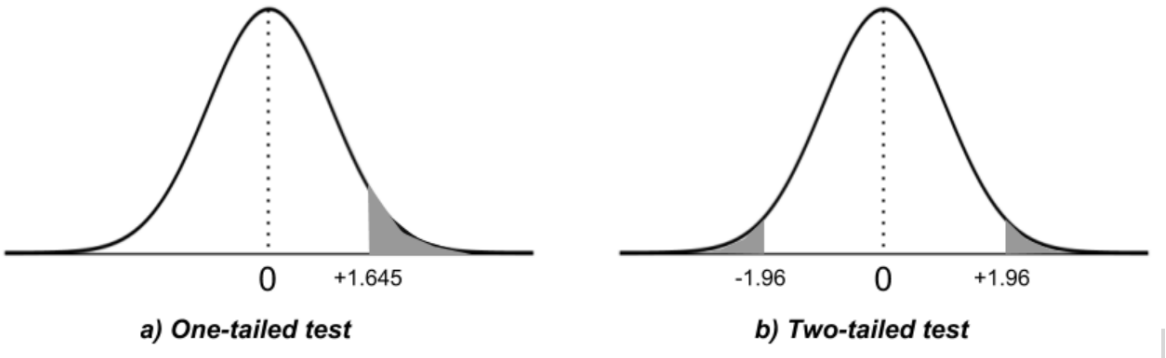


Figure 2. Example of a one-tailed (a) versus a two-tailed (b) test. In this example, the user is comparing the mean of some quantity to 0. In both tests, a significance level of α=5% was pre-selected. In the one-tailed test, the hypothesis is that the mean is strictly greater than 0. You can see that in a one-tailed test, significance is reached faster (there is a difference of 1.645). In the two-tailed test, significance is reached for larger differences (> |1.96|) even though α is the same.

In a two-tailed test, you are testing the probability of observing an effect in either direction, positive or negative. Unless you are are sure that the detecting effect in the opposite direction is of no interest or is impossible, our recommendation is to use a two-tailed test.

Key Takeaways

- A/A tests are a valuable tool in detecting experiment design flaws.
- A/A tests can also be used to collect data needed for variance estimation.
- Never stop experiment and draw conclusion as soon as you see statistical significance. Always run the experiment for enough time to collect data and correct for newness effects.
- Statistical test type will depend on data type and model being tested.
- Two-tailed tests are recommended as safe option for testing direction of the effect.

The Pitfalls of A/B Testing

A/B testing is an intuitive concept that is deceptively simple. In practice, A/B tests are hard to get right. In this section, we go over some common A/B testing pitfalls you should be aware of.

Multiple Hypothesis Comparison Problem

When multiple models are deployed and compared via statistical tests, more than one comparison is performed. Performing multiple comparisons increases the chances of detecting a false positive (hereafter FP or Type I error). In other words, you are more likely to find a spurious effect when there is none.

A common way to avoid a higher incidence of FP is to apply a Bonferroni correction (e.g., Dunn 1958). This correction essentially forces the analyst to use a more a stringent statistical significance level (α) to take into account the number of comparisons performed. This is done by dividing α by the number of models to compare m (α/m). In other words, the Bonferroni correction is one way to adopt a more conservative approach when multiple comparisons between models are performed.

The Bonferroni correction can sometimes be overly conservative, leading to a high rate of false negatives. An alternative is a procedure proposed by Benjamini and Hochberg (1995). This method involves a false discovery rate correction procedure based on the ranking of p-values.

Changing Experiment Settings (Simpson's Paradox)

You should avoid the urge to change experiment settings during a running experiment. Changing experiment settings (e.g., traffic allocation between models A and B) will skew the results in an unwanted direction. This is known as Simpson's paradox (e.g., Good and Mittal 1987). Simpson's paradox, also referred to as reversal paradox, is a statistical phenomenon that occurs when trends appear in different groups but disappear or reverse when the groups are aggregated.

For example, it is not uncommon to roll out a new model gradually, ramping up the fraction of users affected with each day or week for the treatment group. It's also possible that users might be segmented during the test. In these scenarios, weights across segments may significantly differ. In turn, this implies that significant aggregate results of these segments loose significance when they are analyzed at the segment level. An example of such a scenario is shown in Table 3 (see also Crook et al. 2009) where events are segmented by day.

Table 3. An illustration of Simpson's paradox from Crook et al. (2009). Two models are considered: C is control and T is treatment. Conversion rates are computed daily and the weighted average is in column titled Total. Even though the conversion rate for T is higher on both days, the weighted average is not. You can see that although the data collected for T on Friday show a high conversion rate, that rate has negligible impact on the weighted average.

	Friday C/T split: 99%/1%	Saturday C/T split: 50%/50%	Total
C	20,000/990,000 = 2.02%	5,000/500,000 = 1.00%	25,000/1,490,000 = 1.68%
T	230/10,000 = 2.30%	6,000/500,000 = 1.20%	6,230/510,000 = 1.20%

In Table 3, Treatment (T) outperforms Control (C) on both days. In contrast, the aggregated values indicate a different conclusion. This apparent contradiction occurs because the aggregated conversation rate is a weighted average of each day. One day could have a high conversion rate, yet its weighted conversion rate contributing to the aggregate value can be negligible. This is exactly what happened to T on Friday when only 10K events were recorded.

To correct this paradox, you can use weighted summary statistics or simply discard ramp-up days and use data with stable allocation among model variations. For more details on Simpson's paradox, please review the paper on the topic by Liddell (2016).

Always Report Confidence Intervals

Running a statistical test and reporting the resulting significance gives only a boolean yes/no answer to whether one model is better than another. In order to quantify the certainty in this decision and possible range of magnitude of the difference, the confidence interval should always be reported along with a test statistic. We invite the reader to take a look at Crook et al. (2009) for more details on confidence intervals calculations.

Verify the Underlying Assumptions of the Test

All of the statistical tests mentioned in Table 2 assume independent and identically distributed test units. It is important to make sure that the groups exposed to different models do not differ in any significant ways. Group populations should be drawn randomly from the same underlying population. This means that units should be picked completely at random. For example, if users can come from different countries, the different model variations should have users coming from these countries in similar proportions. Or, in the event that only users of Germany are of interest, first filter on country and then apply randomized assignment to the different models.

Isolate Experiment Impact

When running multiple experiments, beware of "cross-pollination" of users between the experiments, i.e., the same user receiving multiple treatments. To accurately estimate model impact, groups of users should ideally be isolated between experiments. If that is not feasible in your application, consider a factorial experiment design.

A/B testing is a pretty robust technique which has been widely adopted for evaluating models in an online experimental setting. In the section above, we articulated some of the common pitfalls to watch out for when deploying an A/B testing setup for online model evaluation. In the next section, we will briefly explore other robust alternatives to A/B testing. Before we do that, let's take a step back and think conceptually about A/B experimental setup for a bit. While deciding between A and B versions of a model, technically you are "exploring" versions. Upon making a choice, you intend to "exploit" the best model.

A Few Words on Bandit Algorithms

An alternative approach to model testing is the use of multi-arm bandits, or MABs (also known as K- or N-armed bandits). The best way to understand MABs is to imagine a gambler (bandit) trying to maximize the sum of his winnings (profits) at slot machines. The gambler may initially decide to explore different slot machines by pulling various arms for some time, but then exploit the slot machine giving him the best results.

Exploration is defined as the process of using different models versions to score incoming data with the goal of finding the best model. **Exploitation** is the process of using the best performing model to maximize the KPI (reward). A winning model variant is determined by optimizing the overall reward associated with the experiment.

Technically, we can think of A/B/ … N testing as an experimental design that includes a short period of pure exploration in which incoming traffic is randomly assigned and evaluated against Model A (80% of traffic) or Model B (20% of traffic), followed by long period of pure exploitation in which 100% of traffic is scored using the better performing variant.

Unlike the explore, stop, and jump-to-exploit paradigm of A/B testing, MABs have a smoother transition from the explore to exploit phase (see Figure 3). They continuously optimize on the KPI by minimizing regret (the model variants that are not performing well) and balancing between exploitation and exploration (e.g., Silver 2017). Bandit tests are adaptive in nature.

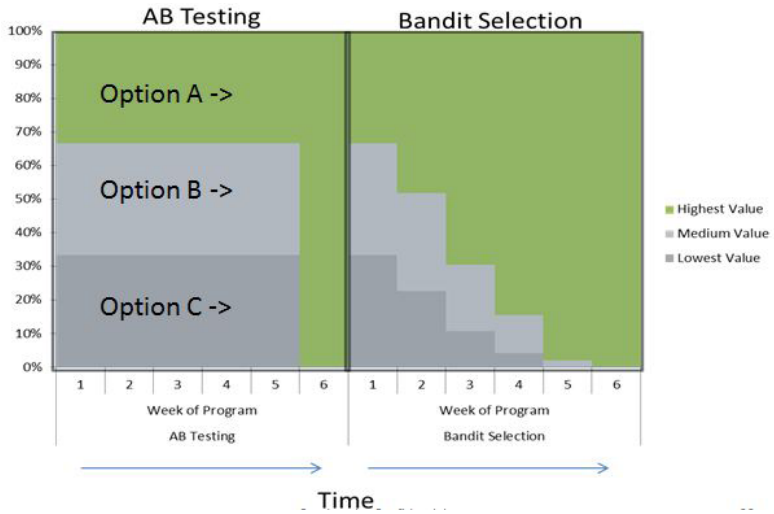


Figure 3. In the left panel, we show a mock A/B test. During the test (week 1 to 5), you explore the different model options. At the end of the test, you concentrate all the test units to the best performing model (Option A). A/B test represents a sharp transition between exploration and exploitation. In contrast, we show a bandit selection in the right panel. You see continuous transition between exploration and exploitation. As the test progresses, more and more test units are allocated to Option A. (**image reference Matt Gershoff, "Balancing Earning with Learning: Bandits and Adaptive Optimization")

Key Takeaways

- Correct your significance level to take into account multiple hypothesis testing.
- Beware of Simpson's paradox when using traffic ramp-up or user segmentation.
- Always report confidence intervals.
- Use A/A tests and careful experiment design to make sure that the test units exposed to different models are independent and identically distributed.

In Summary

In this white paper, we provided an overview of model testing for deployed models in production. We discussed some of the root causes of the differences between performance metrics obtained during model training and the ones measured post-deployment on new, unseen data. We introduced the concept of model testing, motivated its use, and provided an overview of A/B tests that can be used to identify the best deployed models.

This white paper is by no means a comprehensive discussion of model testing and model monitoring. Our intention is to provide data scientists with a basic understanding of why model testing is important and how you can start applying this methodology to models deployed in production.

References

- Benjamini, Y., Hochberg, Y., 1995, *"Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing"*, J. R. Statist. Soc. B., 57, 1, pp 289-300
- Crook, T., Frasca, B., Kohavi, R. et al., 2009, *"Seven Pitfalls to Avoid when Running Controlled Experiments on the Web"*, KDD'09 Paris France, ACM
- Dunn, O.J. 1958, *"Estimation of the Means for Dependent Variables"*, Annals of Mathematical Statistics, 29, 4, pp. 1095-1111
- Good, I.J., Mittal, Y. 1987. *"The Amalgamation and Geometry of Two-by-Two Contingency Tables",* The Annals of Statistics, 15, 2, pp. 694-711.
- Kohavi R., Deng, A., Frasca, B., et al., 2012, *"Trustworthy Online Controlled Experiments: Five Puzzling Outcomes Explained"*, KDD'12 Beijing China, ACM
- Kohavi, R., Longbotham, R., Sommerfield, D., Henne, R.M., 2009, *"Controlled Experiments on the web: survey and practical guide"*, Data Mining and Knowledge Discovery, 18, 1, pp 140-181
- Kohavi, R et al. 2007. Retrieved from http://exp-platform.com/hbr-the-surprising-power-of-online-experiments/
- Liddel, M. (2016, 01, 14), How statistics can be misleading. Retrieved from https://www.youtube.com/watch?time_continue=1&v=sxYrzzy3cq8
- List, J.A., Sadoff, S., Wagner, M. 2011, *"So you want to run an experiment, now what? Some simple rules of thumb for optimal experimental design"*, Exp. Econ., 14, 439
- Silver, D. 2017, *"Exploration and Exploitation"*, Retrieved from http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/XX.pdf
- White, J.M., 2013, *"Bandit Algorithms for Website Optimization"*, O'Reilly Media Inc.
- Zheng, A., 2015, *"Evaluating Machine Learning Models"*, O'Reilly Media, Inc.

Quick Model Testing Cheat Sheet

- Perform an A/A test. At α=0.05, a significant result should be seen 5% of the time.
- Do not turn off the test as soon as you detect an effect. Stick to your pre-calculated sample size. Often, there is a novelty effect in first few days of model deployment and a higher risk of false positives.
- Use a two-tailed test instead of a one-tailed test.
- Control for multiple comparisons.
- Beware of cross-pollination of users between experiments.
- Make sure users are identically distributed. Any segmentation (traffic source, country, etc.) should be done before randomization is needed.
- Run tests long enough to capture variability such as day of the week seasonality.
- If possible, run the test again. See if the results still hold.
- Beware of Simpson's paradox.
- Report confidence intervals; they are different for percent change or non-linear combinations of metrics.

Managing Models With DataScience.com

Oracle's DataScience.com focuses on making enterprise data science teams more productive and impactful across entire organizations. That's why we're incorporating a powerful programmatic system into our enterprise data science platform to track, evaluate, and operationalize models from development to production. This new feature will help your team:

- Efficiently develop and select high-performing models
- Track the model lifecycle, from development to production
- Visualize, compare, and interpret models
- Link workflows together and automate deployments
- Monitor deployed model performance

Contact us today to learn how the DataScience.com Platform can elevate your model management capabilities using the open source stack your team already loves.

About DataScience.com

Oracle's DataScience.com platform makes it easy and intuitive for data scientists to work collaboratively with IT and business teams on the data-driven projects that transform how companies do business. Explore and visualize data, share analyses, deploy models into production, and track performance – all from one place.

ORACLE + DATASCIENCE.COM

Connect with us on social media:

@DataScienceInc

https://www.linkedin.com/company/datascienceinc

@DataScienceInc

https://www.facebook.com/datascience

Oracle Corporation, World Headquarters 500 Oracle Parkway Redwood Shores, CA 94065 USA