**Mastering P-values in Machine Learning**

**Understanding P-values and ML use cases**

A p-value is a statistical metric that helps statisticians decide whether they should accept or reject the null hypothesis. The p-value measures the probability there is no relationship between variables. A low p-value gives evidence against the null hypothesis. P-values are often misinterpreted. For example, it often leads people…

For example, if a drug company is testing a new drug in clinical trials, it may observe that the drug is effective at treating the symptoms of a specific illness. Despite these observations, there is a chance that the drug effect that is observed is due to chance and the drug is actually ineffective. To conclude that observation isn’t due to chance and represents a real effect, p-values can be used to measure the probability that the observation is due to random chance.

P-values are traditionally used across many statistical techniques including ANOVA, t-tests, and regression. In addition to traditional techniques, while less common, p-values can be used to test hypotheses when building machine learning models. In my experience, the necessity for p-values as a data scientist lies at the boundary between traditional statistical methods and machine learning. For example, regression techniques such as linear regression and logistic regression often necessitate p-value calculations for regression coefficients. Linear regression and logistic regression are used by both statisticians and data scientists. Another example is feature selection. P-values can be used to test feature selection techniques including ANOVA (traditional statistics approach) and tree-based feature selection (data science/ML approach). Further, p-values can be used to assess confidence levels in machine learning predictions and feature importance calculations. Specifically, p-values can be used to determine if the difference in performance between the two models is statistically significant.

Another distinction worth noting is between inference (traditional statistics) and prediction (machine learning). P-values are often thought to be unnecessary with machine learning because of this distinction. In inferential statistics, the emphasis is often theory behind a relationship while in machine learning the emphasis is more or predicting on unseen data accurately. Inferential statistics employ metrics like p-values, R2, and F-statistic for model validation. Machine learning models necessitate cross-validation and out-of-sample testing for model validation.

To understand this distinction a bit further, let’s consider our drug example from earlier. In the context of **statistical inference,** we would like to be able to have statistical evidence against the statement “Drug X is ineffective at treating disease Y”. In **machine learning**, we would be more interested using predictions to screen a catalog of potential drugs and label them as effective or ineffective. The former is more about **validating claims about a relationship**, while the latter is about **validating how well you can predict** unseen data. Despite these distinctions, there is still good reason for data scientists to consider understanding and using p-values and for statisticians to consider using non-parametric machine learning models that may or may not necessitate p-values.

While p-values are used across the natural and social sciences they have their limitations. The limitations of p-values have been a recent popular topic of discussion amongst data scientists and statisticians. For example, p-values don’t measure effect size, importance, or even if an effect is 100% true. It is more so used to support claims, rather than solidify claims as truth. Despite this, data scientists should have an understanding of their place in traditional statistics as well as how they can be used with machine learning.

Here we will walk through how to calculate p-values for coefficients in linear and logistic regression models. We will discuss how to calculate p-values for machine learning predictions. We can use p-values to compare a new model to a baseline and compare the performance of different ML algorithms.

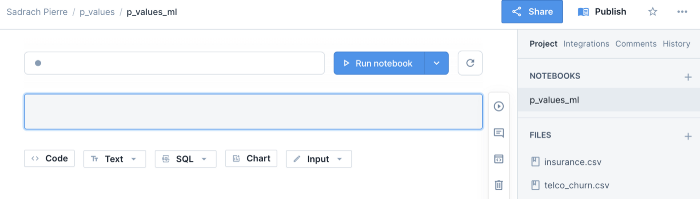
For this work, I will be writing code in [Deepnote](https://deepnote.com/), which is a collaborative data science notebook that makes running reproducible experiments very easy.

For our regression models, we will be working with the [Medical Cost dataset](https://www.kaggle.com/datasets/mirichoi0218/insurance). We will use patient attributes such as age, body mass index, and the number of children to predict medical costs. The data is publicly free to use, modify and share under the [Database Contents License](https://opendatacommons.org/licenses/dbcl/1-0/) (DbCL: Public Domain).

For our classification models, we will work with the fictitious [Telco Churn](https://www.kaggle.com/datasets/blastchar/telco-customer-churn) data set, which is publicly available on Kaggle. The data set is free to use, modify and share under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0).

**Adding Datasets**

To start, let’s navigate to Deepnote and create a new project (you can sign-up for free if you don’t already have an account).  
Let’s create a project called ‘p\_values’ and a notebook within this project called ‘p\_values\_ml’. Also, let’s drag and drop the insurance.csv & telco\_churn.csv files on the left-hand panel on the page where it says ‘FILES’:



Screenshot taken by author

**P-values for Linear Regression Coefficients**

We will use p-values to test the following:

***Alternative Hypothesis (H1):*** The features age, BMI, and children are important for predicting cost.

***Null Hypothesis (H0)***: The features age, BMI and children have no relationship to cost.

We will start by installing the stats module in python. This will allow us to calculate p-values for our linear model since Scikit-learn does not provide p-values for model coefficients.

Embedding created by author

Next, let’s import the Pandas library, read our insurance data into a Pandas dataframe, and display the first five rows:

Embedding created by author

Next, let’s define our inputs. Let’s define our inputs as age, BMI, and the number of children and output as charges. Let’s also split our data for training and testing:

Embedding created by author

Next, let’s import the stats models API and fit our model to our training data. The result is a table of statistical metrics summarizing our model. Here we are only interested in the p-values column, though it also provides other metrics such as R-squared, t-test, and standard error:

Embedding created by author

We see that the p-values <0.05 for age, BMI, and the number of children. With this, we have evidence against the null hypothesis, which states that there is no relationship between our inputs and output.

**P-values for Logistic Regression Coefficients**

We will use p-values to test the following:

***Alternative Hypothesis (H1):*** The features tenure, monthly charges, and tenure\_squared are important for predicting churn.

***Null Hypothesis (H0)***: The features tenure, monthly charges, and tenure\_squared have no relationship to churn.

We can do something similar for a logistic regression model. Let’s read our churn data into a data frame, generate coded churn labels, and display the first five rows:

Embedding created by author

Let’s define our input as tenure and monthly charges. Let’s also calculate the square of each feature and use them as model inputs. The output will be the churn column. Let’s also split our data for training and testing:

Embedding created by author

We can train our logistic regression model and print the summary:

Embedding created by author

The p-values < 0.05 gives evidence to reject the null hypothesis (there is no relationship between input & output).

**P-values for comparing Random Forest Regression Models**

***Alternative Hypothesis (H1):*** There is a real difference in model performance.

***Null Hypothesis (H0)***: There is no difference in model performance.

Suppose we’d like to compare two random forest regression models. We can use P-values to give evidence for rejecting the null hypothesis. Here, the null hypothesis would be that model performance does not differ between the models. Let’s build one model using default random forest parameters, and a second using n\_estimators =5 and a max\_depth =5. Let’s generate predictions for each on the test set:

Embedding created by author

Next, we need to install the [mlxtend](https://rasbt.github.io/mlxtend/) package. This will allow us to compare model performance:

Embedding created by author

Next, let’s use the [paired\_ttest\_5x2cv](http://rasbt.github.io/mlxtend/user_guide/evaluate/paired_ttest_5x2cv/) method in mlxtend to generate our p-value. We pass both estimators and parameter values as well as the training data:

Embedding created by author

We see that we don’t have evidence to reject the null hypothesis here. This means that a difference in model performance may be due to chance.

**P-values for comparing different ML algorithms**

Let’s compare our random forest model with default parameters to our linear regression model:

Embedding created by author

We see that we have evidence to reject the null hypothesis here since p <0.05. This means that a difference in model performance is unlikely due to chance.

The code from this post is available on [GitHub](https://github.com/spierre91/deepnote/blob/main/p_values_ml.ipynb).

**CONCLUSIONS**

In this post, we discuss how to apply p-values to machine learning use cases. First, we looked at how to generate p-values for linear regression coefficients. We then discussed how to generate p-values for logistic regression. Next, we considered more modern ML applications. We showed how to generate p-values to determine whether the difference in performance between two random forest models was significant. Finally, we used p-values to compare different ML algorithm types. Specifically, we calculated p-values to compare random forest and linear regression models.