Evaluate ML Classifier Performance using Statistical Hypothesis Testing in Python

Have a strong argument why picking a classification algorithm over the other based on significance level in performance

Salma Elshahawy, MSc. Sep 9, 2020 · 7 min read ★

Photo by Alexis Fauvet on Unsplash

Introduction

Picking the right machine learning algorithm is decisive, where it decides the performance of the model. The most dominating factor in choosing a model is the performance, which employs the KFold-cross-validation technique to achieve independence.

The chosen model usually has a higher mean performance. Nevertheless, sometimes it originated through a statistical fluke. There are many **statistical hypothesis-testing** approaches to evaluate the mean performance difference resulting from the cross-validation to address this concern. If the difference is above the significance level `**p-value**` we can reject the null hypothesis that the two algorithms are the same, and the difference is not significant.

I usually include such a step in my pipeline either when developing a new classification model or competing in one of Kaggle's competitions.

. . .

Tutorial Objectives

1. Understanding the difference between statistical hypothesis tests.
2. Model selection based on the mean performance score could be misleading.
3. Why using the Paired Student's t-test over the original Student's t-test.
4. Applying the advance technique of **5X2 fold** by utilizing the **MLxtend** library for comparing the algorithms based on **p-value**

. . .

Table of content

. . .

What does the statistical hypothesis testing mean?

A statistical hypothesis test quantifies how plausible it is to witness two data samples, considering that they have the same distribution. That describes the null hypothesis. We can test this null hypothesis by applying some statistical calculations.

- If the test result infers **insufficient proof** to reject the null hypothesis, then any observed difference in the model scores is a happened by chance.
- If the test result infers **sufficient evidence** to reject the null hypothesis, then any observed difference in model scores is real.

. . .

Types of statistical hypothesis testings

Examining machine learning models via statistical significance tests requires some expectations that will influence the statistical tests used. The most robust way to such comparisons is called **paired designs**, which compare both models (or algorithms) performance on the same data. That way, both models (or algorithms) have to deal with the same difficulty.

In the following, each test has some pros and cons that you should consider during the selection.

P.N: There are other statistical tests used to compare classifiers, but those are the most recommended one.

1. **Independent data samples**: used when having an unlimited dataset. You collect **n samples** for the train, and test dataset. Then calculate ten independent model scores for each method. Finally, apply the t-test to compare models. However, this approach isn't practical because there is no unlimited data in real.

2. Ten-fold cross-validation: It uses the ordinary paired t-test. This method has good repeatability relative to other ways as well as a decent **type II error**. However, it has a high **type I error**; that's why it is not recommended.

Comparing **training** algorithms via cross-validation makes stronger assumptions than comparing specific (fully trained) models' **predictive performance**. Where the resampling validation (to which cross-validation belongs) cannot fully estimate the algorithm comparison's variance uncertainty.

3. McNemar's test:

> *In statistics, **McNemar's test** is a statistical test used on paired nominal data. It is applied to 2 × 2 contingency tables with a*

> *dichotomous trait, with matched pairs of subjects, to determine whether the row and column marginal frequencies are equal (that is, whether there is "marginal homogeneity"). — Wikipedia*

It is recommended for the last twenty years. However, this method's challenge is that you either need to build your functions to implement it or use a third party library where it is not commonly packed up for you in the tools you use.

4. Non-parametric Paired test: this method involves making a few assumptions. For example, assuming the distribution of model accuracies has a normal distribution(Gaussian).

The Wilcoxon signed-rank test is a non-parametric version of the paired Student's t-test. Although the test is non-parametric, it still assumes the independency for observations inside each sample. Although using the k-fold cross-validation would break that assumption.

5. Estimation statistics: A data analysis framework that uses a combination of effect sizes, confidence intervals, precision planning, and meta-analysis to plan experiments, analyze data, and interpret results — Wikipedia. Nevertheless, when evaluating the model using the resampling method, the assumption of independence is broken. As an alternative, other statistical resampling methods like bootstrapping. Bootstrapping could estimate a robust non-parametric confidence interval. Hence, we can interpret the results and compare classifiers.

. . .

The intuition behind the 5X2 fold approach

An approach to evaluate each model on the same k-fold cross-validation split of the data and calculates each split score. That would give a sample of ten scores for ten-fold cross-validation. Then, we can compare those scores using the paired statistical test.

Due to using the same data rows to train the model more than once, the assumption of independence is violated; hence, the test would be biased.

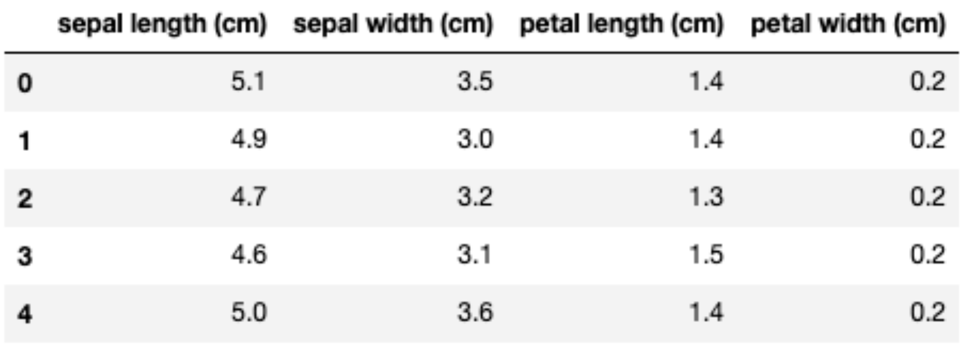
This statistical test could be adjusted to overcome the lack of independence. Also, the number of folds and repeats of the method can be configured to achieve a better sampling of model performance.

> *Thomas Dietterich proposed this approach in the "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms." — 1998*

. . .

Loading dataset

For this tutorials, I will make use of the `load_iris` dataset within the `sklearn` library. However, the steps are the same for any ML problem.



	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

peek into the iris dataset

. . .

Steps to extract the best two models

In this steps, I will conduct a comparison between four different algorithms based on performance accuracy score. Then will select the two models with the highest score to conduct hypothesis testing between them.

```
# Spot-Check Algorithms
models = []
models.append(('LR', LogisticRegression(max_iter=1000)))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('DSC', DecisionTreeClassifier(random_state =
1, max_depth=2)))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = RepeatedStratifiedKFold(n_splits=10, n_repeats
= 3, random_state=1)
    cv_results = cross_val_score(model, X, y, cv=kfold,
scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %.2f (%.3f)" % (name, cv_results.mean(),
cv_results.std())
    print(msg)
```

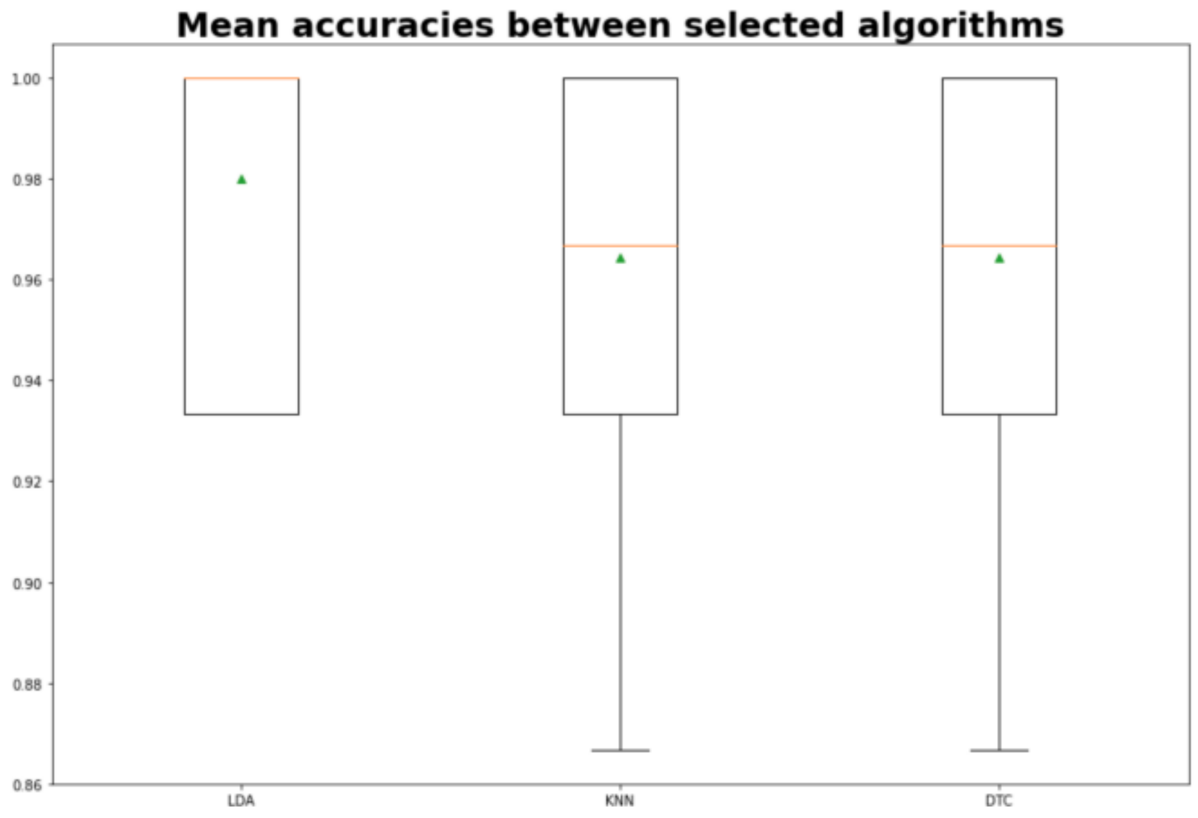
Results:

```
LR: 0.96 (0.041)
LDA: 0.98 (0.031)
KNN: 0.96 (0.037)
DSC: 0.94 (0.051)
SVM: 0.96 (0.045)
```

It seems that `LR,` `KNN` and `SVM` has same mean with slightly different standard deviation. However, `LDA` shows a higher performance, in the other hands, `DTC` shows the lowest performance over the rest of algorithms. Let's build a boxplot betweeen `KNN` , `DTC,` and `LDA` as a visualization for more interpretation.

```
import matplotlib.pyplot as plt

plt.figure(figsize = (15, 10))
plt.grid(False)
plt.title("Mean accuracies between the best two selected
algorithms", fontsize = 25, fontweight = 'bold')
```



mean accuracies between algorithms

It seems that LDA and DTC perform the same, so Let's pick those two.

> *For this classification problem, you could implement logistic regression. However, I went for more complex classification algorithms to show the idea of hypothesis testing.*

. . .

Steps to hypothesis testing

The first step would be to to state the null hypothesis statement.

> *H0: Both models have the same performance on the dataset.*
>
> *H1: Both models doesn't have the same performance on the dataset.*
>
> *Significance level is 0.05*

let's assume a significance threshold of **α=0.05** for rejecting the null hypothesis that both algorithms perform equally well on the dataset and conduct the 5x2_cv _t_test.

```
# evaluate model 1
model1 = LinearDiscriminantAnalysis()
cv1 = RepeatedStratifiedKFold(n_splits = 10, n_repeats = 3,
random_state = 1)
scores1 = cross_val_score(model1, X, y, scoring =
'accuracy', cv = cv1, n_jobs = -1)
print('LDA Mean Accuracy: %.1f%% +/-(%.3f)' %
(mean(scores1*100), std(scores1)))

# evaluate model 2
model3 = DecisionTreeClassifier(random_state = 1,
max_depth=2)
cv2 = RepeatedStratifiedKFold(n_splits = 10, n_repeats = 3,
random_state = 1)
scores3 = cross_val_score(model2, X, y, scoring =
'accuracy', cv = cv2, n_jobs = -1)
print('DecisionTreeClassifier Mean Accuracy: %.1f%% +/-
(%.3f)' % (mean(scores3*100), std(scores3)))

# plot the results
plt.boxplot([scores1, scores2], labels=['LDA', 'DTC'],
showmeans=True)
plt.show()
```

Results

```
LDA Mean Accuracy: 98.0% +/-(0.031)
DecisionTreeClassifier Mean Accuracy: 96.4% +/-(0.037)
```

Seems that `LDA` has a better performance over the `DTC` where `LDA` has a higher accuracy.

. . .

5 by 2 CV using MLxtend package

You can implement the 5X2 CV fold from scratch; however, there is a nice package that saves you much time called MLxtend. I will use the paired_ttest_5x2cv function from the evaluation module to calculate the t and p value for both models.

```
from mlxtend.evaluate import paired_ttest_5x2cv
# check if difference between algorithms is real
t, p = paired_ttest_5x2cv(estimator1=model1,
                          estimator2=model2,
                          X=X,
                          y=y,
                          scoring='accuracy',
                          random_seed=1)
# summarize
print(f'The P-value is = {p:.3f}')
print(f'The t-statistics is = {t:.3f}')
# interpret the result
if p <= 0.05:
    print('Since p<0.05, We can reject the null-hypothesis
that both models perform equally well on this dataset. We
may conclude that the two algorithms are significantly
different.')
else:
    print('Since p>0.05, we cannot reject the null
hypothesis and may conclude that the performance of the two
algorithms is not significantly different.')
```

Results:

```
The P-value is = 0.027
The t-statistics is = 3.101
Since p<0.05, We can reject the null-hypothesis that both
models perform equally well on this dataset. We may
conclude that the two algorithms are significantly
different
```

Now you have a strong argument about why picking LDA over the DTC.

. . .

Summary

Finally, I hope this tutorial gave you a nice illustration of how to use hypothesis testing to develop a more makes sense model. My suggestion is to include algorithms comparison in your classification pipeline. Try to iterate as well as trying different algorithm performance comparisons.

Thanks for reading!

References

- Kaggle competitions
- MLxtend library
- Approximate statistical tests for comparing supervised classification learning algorithms — Thomas Dietterich, 1998
- MLxtend.evaluate.5X2cv paired t test API
- Sklearn general datasets API
- Executable kaggle notebook

Thanks to Anne Bonner.

Classification Algorithms Machine Learning Hypothesis Testing Statistics Python