Stacking Ensemble Machine Learning With Python

by **Jason Brownlee** on April 10, 2020 in **Ensemble Learning**

Stacked Generalization is an ensemble machine learning algorithm.

It uses a meta-learning algorithm to learn how to best combine the predictions from two or more base machine learning algorithms.

The benefit of stacking is that it can harness the capabilities of a range of well-performing models on a classification or regression task and make predictions that have better performance than any single model in the ensemble.

In this tutorial, you will discover the stacked generalization ensemble or stacking in Python.

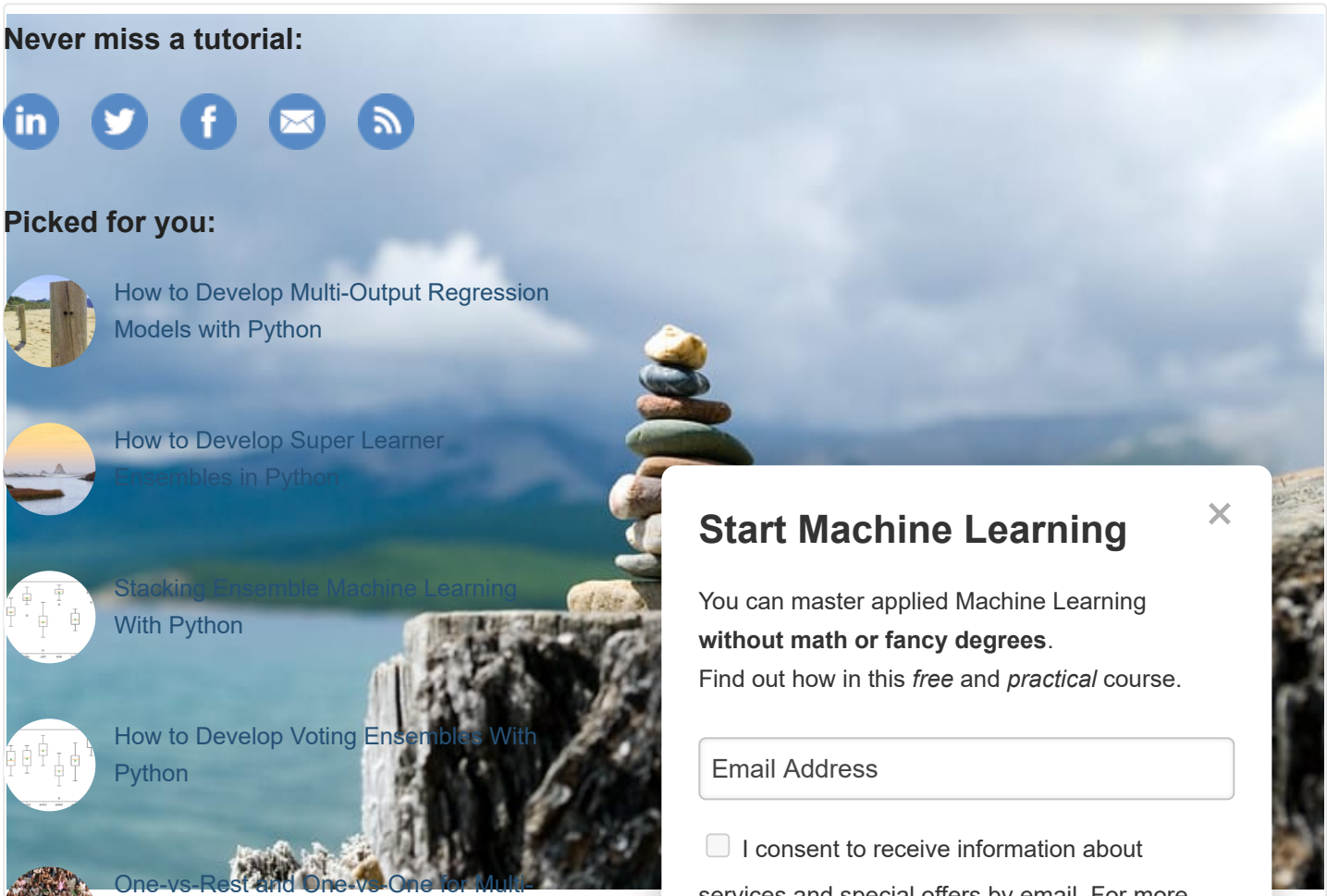
After completing this tutorial, you will know:

- Stacking is an ensemble machine learning algorithm that learns how to best combine the predictions from multiple machine learning models.
- The scikit-learn library provides a standard implementation of the stacking ensemble in Python.
- How to use stacking ensembles for regression and classification predictive modeling.

Kick-start your project with my new book Ensemble Learning Algorithms With Python, including *step-by-step tutorials* and the *Python source code* files for all examples.

Let's get started.

- **Updated Aug/2020**: Improved code examples, added more references.



Stacking Ensemble Mach

Photo by lamoix, sc

Tutorial Overview

This tutorial is divided into four parts; they are:

1. Stacked Generalization
2. Stacking Scikit-Learn API
3. Stacking for Classification
4. Stacl

Stacked Generalization

Stacked Generalization or "Stacking" for short is an ensemble machine learning algorithm.

It involves combining the predictions from multiple machine learning models on the same dataset, like bagging and boosting.

g addresses the question:

- Given multiple machine learning models that are
- u choose which model to use (trust)?

The approach to this question is to use another machi
each model in the ensemble.

- like bagging, in stacking, the models are typica
same dataset (e.g. instead of samples of the train
- Unlike boosting, in stacking, a single model is use
contributing models (e.g. instead of a sequen
models).

The architecture of a stacking model involves two or m
models, and a meta-model that combines the predictions of the base models, referred to as a level-1 model.

- **Level-0 Models (*Base-Models*)**: Models fit on the training data and whose predictions are compiled.
- **Level-1 Model (*Meta-Model*)**: Model that learns how to best combine the predictions of the base models.

The meta
predictions made by base models on out-of-sample data. That is, data not used to train the base models is fed to the base models, predictions are made, and these predictions, along with the expected outputs, provide the input and output pairs of the training dataset used to fit the meta-model.

The outputs from the base models used as input to the meta-model may be real value in the case of regression, and probability values, probability like values, or class labels in the case of classification.

The most common approach to preparing the training dataset for the meta-model is via k-fold cross-validation of the base models, where the out-of-fold predictions are used as the basis for the training dataset for the meta-model.

The training data for the meta-model may also include the inputs to the base models, e.g. input elements of the training data. This can provide an additional context to the meta-model as to how to best combine the predictions from the models.

Once the training dataset is prepared for the meta-model, the meta-model can be trained in isolation on this dataset, and the base-models can be trained on the entire original training dataset.

Stacking is appropriate when multiple different machine learning models have skill on a dataset, but have skill in different ways. Another way to say this is that the predictions made by the models or the errors in predictions made by the models are uncorrelated or have a low correlation.

The models are often complex and diverse. As such, it is often a good idea to use a range of models that make very different assumptions about how to solve the, decision trees, support vector machines, neural netwo so be used as base-models, such as random forests.

- **Base-Models**: Use a diverse range of models tha task.

The meta-model is often simple, providing a smooth in models. As such, linear models are often used as the ion tasks (predicting a numeric value) and logistic regress). Although this is common, it is not required.

- **Regression Meta-Model**: Linear Regression.
- **Classification Meta-Model**: Logistic Regression.

The use of a simple linear model as the meta-model often gives stacking the colloquial name "*blending*." As in the prediction is a weighted average or blending of the predictions made by the base models.

The super learner may be considered a specialized type of stacking.

Stacking is designed to improve modeling performance, although is not guaranteed to result in an improvem

Achieving an improvement in performance depends on the complexity of the problem and whether it is sufficiently well represented by the training data and complex enough that there is more to learn by combining predictions. It is also dependent upon the choice of base models and whether they are sufficiently skillful and sufficiently uncorrelated in their predictions (or errors).

If a base-model performs as well as or better than the stacking ensemble, the base model should be used instead, given its lower complexity (e.g. it's simpler to describe, train and maintain).

Stacking Scikit-Learn API

Stacking can be implemented from scratch, although t

For an example of implementing stacking from scratch

For an example of implementing stacking from scratch

The scikit-learn Python machine learning library provi
learning.

It is available in version 0.22 of the library and higher.

First, confirm that you are using a modern version of the library by running the following script:

```
# check scikit-learn version
import sklearn
print(sklearn.__version__)
```

Running the script will print your version of scikit-learn.

Your version should be the same or higher. If not, you must upgrade your version of the scikit-learn library.

```
0.22.1
```

Stacking is provided via the StackingRegressor and StackingClassifier classes.

Both models operate the same way and take the same arguments. Using the model requires that you specify a list of estimators (level-0 models), and a fina

A list of level-0 models or base models is provided via the "*estimators*" argument. This is a Python list where each element in the list is a tuple with the name of the model and the configured model instance.

For example, below defines two level-0 models:

```
...
models = [('lr',LogisticRegression()),('svm',SVC())
stacking = StackingClassifier(estimators=models)
```


Each model in the list may also be a Pipeline, including any data preparation required by the model prior to fitting the model on the training dataset. For example:


```
...
models = [('lr',LogisticRegression()),('svm',m
stacking = StackingClassifier(estimators=model
```


A meta-model or meta-model is provided via the "
LinearRegression for regression and *LogisticRegressi* lts that you probably do not want to change.

The dataset for the meta-model is prepared using cros
used, although this can be changed via the "*cv*" argum
cross-validation) or a cross-validation object (e.g. *Stra*

Sometimes, better performance can be achieved if the es
inputs to the level-0 models, e.g. the input training dat gh"
argument to True and is not enabled by default.

Now that we are familiar with the stacking API in scikit-learn, let's look at some worked examples.

Stacking for Classification

In this section, we will look at using stacking for a classification problem.

First, we can use the make_classification() function to create a synthetic binary classification problem with 1,000 examples and 20 input features.

The complete example is listed below.

```
# test classification dataset
from sklearn.datasets import make_classificati
```

```
# define dataset
X, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundant=5, rando
# summarize the dataset
print(X.shape, y.shape)
```

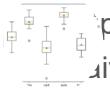
Running the example creates the dataset and summarizes the shape of the input and output components.

```
(1000, 20) (1000,)
```

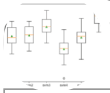
We can evaluate a suite of different machine learning models on the dataset.

Specifically, we will evaluate the following five algorithms:

- Logistic Regression.
- k-Nearest Neighbors.
- Decision Tree.
- Support Vector Machine.
- Naive Bayes.



Each algorithm will be evaluated using default model hyperparameters.



We will define the models we wish to evaluate.

```
# get a list of models to evaluate
def get_models():
    models = dict()
    models['lr'] = LogisticRegression()
    models['knn'] = KNeighborsClassifier()
    models['cart'] = DecisionTreeClassifier()
    models['svm'] = SVC()
    models['bayes'] = GaussianNB()
    return models
```

Each model will be evaluated using repeated k-fold cross-validation.

The *evaluate_model()* function below takes a model instance and returns a list of scores from three repeats of stratified 10-fold cross-validation.

```
# evaluate a given model using cross-validation
def evaluate_model(model, X, y):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
    scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1, error_score='rai
    return scores
```

We can then report the mean performance of each algorithm and also create a box and whisker plot to compare the distribution of accuracy scores for each algorithm.

Tying this together, the complete example is listed below.

```
# compare standalone models for binary classi
from numpy import mean
```

```
from numpy import sta
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from matplotlib import pyplot
# get the dataset
def get_dataset():
    X, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundant=5,
    return X, y
# get a list of models to evaluate
def get_models():
    models = dict()
    models['lr'] = LogisticRegression()
    models['knn'] = KNeighborsClassifier()
    models['cart'] = DecisionTreeClassifier()
    models['svm'] = SVC()
    models['bayes'] = GaussianNB()
    return models
# evaluate a given model using cross-validati
def evaluate_model(model, X, y):
    cv = RepeatedStratifiedKFold(n_splits=10,
    scores = cross_val_score(model, X, y, sco                                    ='ro
    return scores
# define dataset
X, y = get_dataset()
# get the models to evaluate
models = get_models()
# evaluate the models and store results
results, names = list(), list()
for name, model in models.items():
    scores = evaluate_model(model, X, y)
    results.append(scores)
    names.append(name)
    print('>%s %.3f (%.3f)' % (name, mean(scores), std(scores)))
# plot model performance for comparison
pyplot.boxplot(results, labels=names, showmeans=True)
pyplot.show()
```

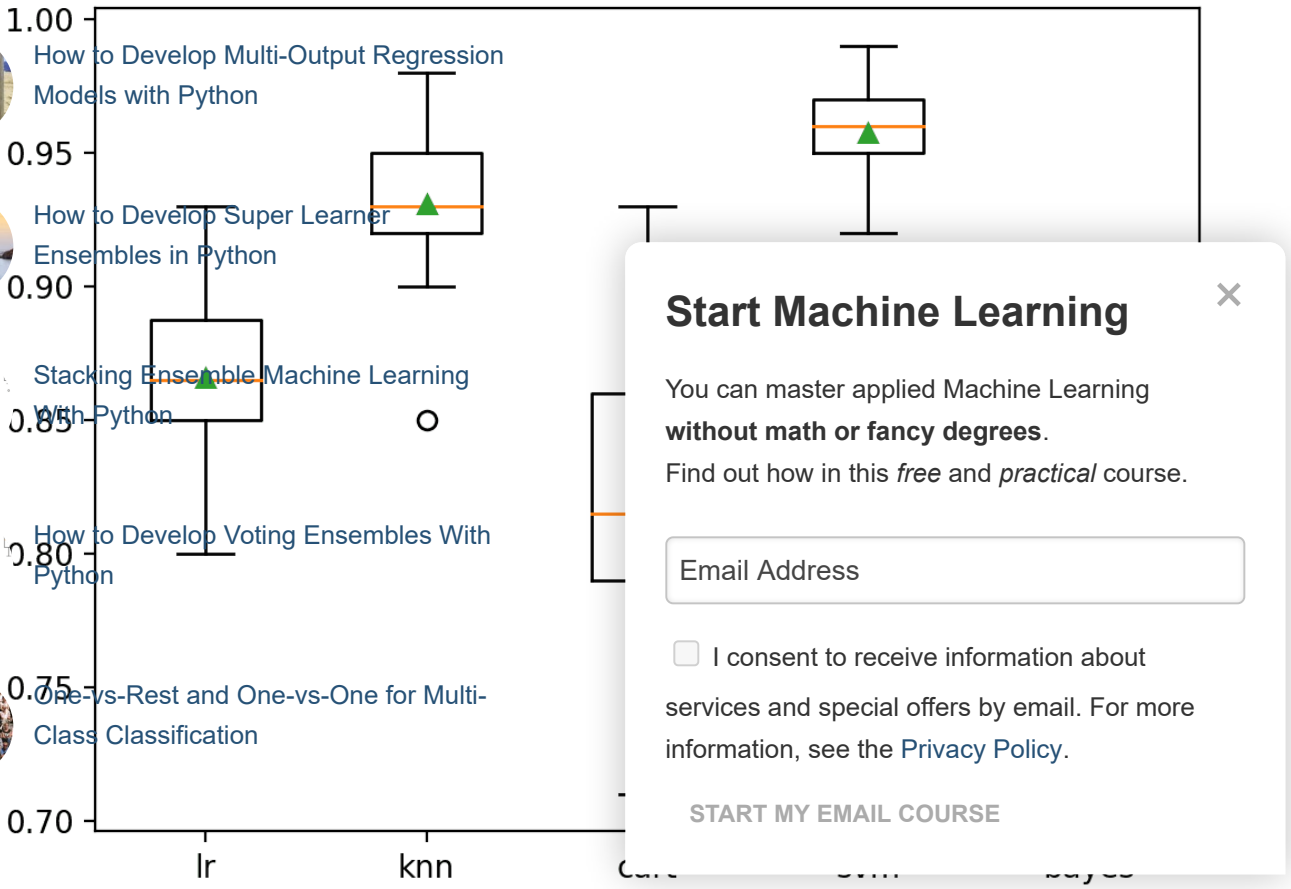

Running t ne mean and standard deviation accuracy for each model.

Note: Your results may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Consider running the example a few times and compare the average outcome.

We can see that in this case, SVM performs the best with about 95.7 percent mean accuracy.

```
>lr 0.866 (0.029)
>knn 0.931 (0.025)
>cart 0.821 (0.050)
>svm 0.957 (0.020)
>bayes 0.833 (0.031)
```

A box-and-whisker plot is then created comparing the distribution accuracy scores for each model, allowing us to clearly see that KNN and SVM perform better on average than LR, CART, and Bayes.



Box Plot of Standalone Model Accuracies for Binary Classification

Here we have five different algorithms that perform well, presumably in different ways on this dataset.

Next, we ive models into a single ensemble model using stacking.

We can use a logistic regression model to learn how to best combine the predictions from each of the separate five models.

The *get_stacking()* function below defines the StackingClassifier model by first defining a list of tuples for the five base models, then defining the logistic regression meta-model to combine the predictions from the base models using 5-fold cross-validation.

```
# get a stacking ensemble of models
def get_stacking():
    # define the base models
    level0 = list()
    level0.append(('lr', LogisticRegression()
```

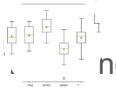
```
    level0.append(('knn', KNeighborsClassifier()))
    level0.append(('cart', DecisionTreeClassifier()))
    level0.append(('svm', SVC()))
    level0.append(('bayes', GaussianNB()))
    # define meta learner model
    level1 = LogisticRegression()
    # define the stacking ensemble
    model = StackingClassifier(estimators=level0, final_estimator=level1, cv=5)
    return model
```

How to Develop Multi-Output Regression Models with Python

We can include the stacking ensemble in the list of models to evaluate, along with the standalone models.

```
# get a list of models to evaluate
def get_models():
    models = dict()
    models['lr'] = LogisticRegression()
    models['knn'] = KNeighborsClassifier()
    models['cart'] = DecisionTreeClassifier()
    models['svm'] = SVC()
    models['bayes'] = GaussianNB()
    models['stacking'] = get_stacking()
    return models
```

Our expectation is that the stacking ensemble will perf



How to Develop Voting Ensembles With Python

not always the case and if it is not the case, the ensemble model.

One-vs-Rest and One-vs-One for Multi-Class Classification

mplete example of evaluating the stacking ense
below.

```
# compare ensemble to each baseline classifier
from numpy import mean
from numpy import std
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import StackingClassifier
from matplotlib import pyplot

# get the dataset
def get_dataset():
    X, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundant=5,
    return X, y

# get a stacking ensemble of models
def get_stacking():
    # define the base models
    level0 = list()
    level0.append(('lr', LogisticRegression()))
    level0.append(('knn', KNeighborsClassifier()))
    level0.append(('cart', DecisionTreeClassifier()))
    level0.append(('svm', SVC()))
```

```
    level0.append(('bayes', GaussianNB()))
    # define meta learner model
    level1 = LogisticRegression()
    # define the stacking ensemble
    model = StackingClassifier(estimators=level0, final_estimator=level1, cv=5)
    return model

# get a list of models to evaluate
def get_models():
    models = dict()
    models['lr'] = LogisticRegression()
    models['knn'] = KNeighborsClassifier()
    models['cart'] = DecisionTreeClassifier()
    models['svm'] = SVC()
    models['bayes'] = GaussianNB()
    models['stacking'] = get_stacking()
    return models
# evaluate a give model using cross-validation
def evaluate_model(model, X, y):
    cv = RepeatedStratifiedKFold(n_splits=10,
    scores = cross_val_score(model, X, y, sco
    return scores

# define dataset
X, y = get_dataset()
# get the models to evaluate
models = get_models()
# evaluate the models and store results
results, names = list(), list()
for name, model in models.items():
    scores = evaluate_model(model, X, y)
    results.append(scores)
    names.append(name)
    print('>%s %.3f (%.3f)' % (name, mean(sco
# plot model performance for comparison
pyplot.boxplot(results, labels=names, showmea
pyplot.show()
```


Running the example first reports the performance of each model. This includes the performance of each base model, then the stacking ensemble.

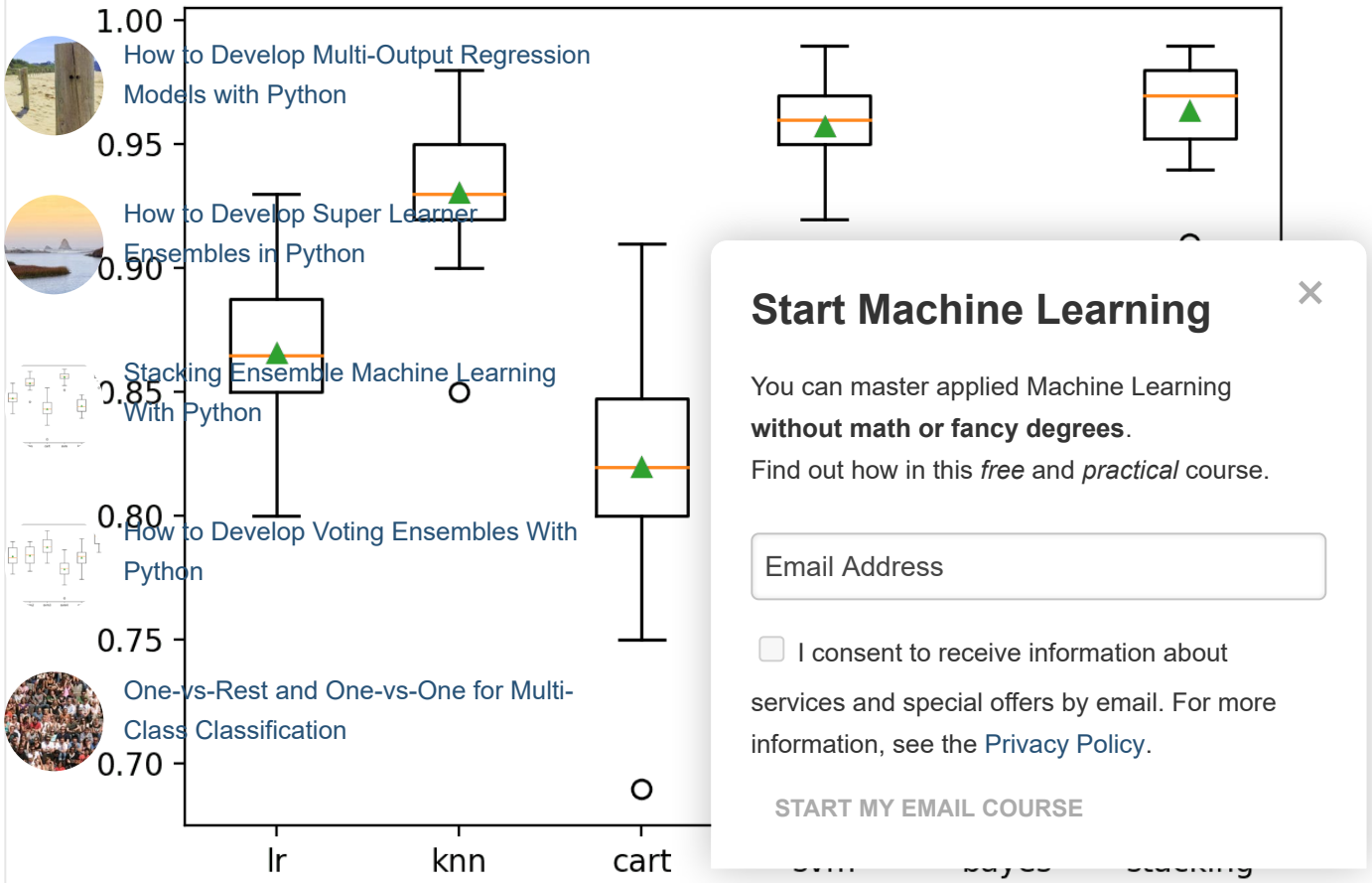
Note: Your results may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Consider running the example a few times and compare the average outcome.

In this case, we can see that the stacking ensemble appears to perform better than any single model on average, achieving an accuracy of about 96.4 percent.

```
>lr 0.866 (0.029)
>knn 0.931 (0.025)
>cart 0.820 (0.044)
>svm 0.957 (0.020)
>bayes 0.833 (0.031)
>stacking 0.964 (0.019)
```

A box plot is created showing the distribution of model classification accuracies.

Here, we can see that the mean and median accuracy for the stacking model sits slightly higher than the SVM model.



Box Plot of Standalone and Stacking Model Accuracies for Binary Classification

If we choose a stacking ensemble as our final model, we can fit and use it to make predictions on new data just like any other model.

First, the stacking ensemble is fit on all available data, then the *predict()* function can be called to make predictions on new data.

The example below demonstrates this on our binary classification dataset.

```
# make a prediction with a stacking ensemble
from sklearn.datasets import make_classification
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
# define dataset
```

```
# define the base models
level0 = list()
level0.append(('lr', LogisticRegression()))
level0.append(('knn', KNeighborsClassifier()))
level0.append(('cart', DecisionTreeClassifier()))
level0.append(('svm', SVC()))
level0.append(('bayes', GaussianNB()))
# define meta learner model
level1 = LogisticRegression()
# define the stacking ensemble
model = StackingClassifier(estimators=level0, final_estimator=level1, cv=5)
# fit the model on all available data
model.fit(X, y)
# make a prediction for one example
data = [[2.47475454,0.40165523,1.68081787,2.88940715,0.91704519,-3.07950644,4.39961206,0.724642
yhat = model.predict(data)
print('Predicted Class: %d' % (yhat))
```

Running the example fits the stacking ensemble mode
r prediction for a new row of data, as we might when us

```
Predicted Class: 0
```


Stacking Ensemble Machine Learning With Python

How to Develop Voting Ensembles With Python

One-vs-Rest and One-vs-One for Multi-Class Classification

Stacking for Regression

In this section, we will look at using stacking for a regression problem.

First, we can use the make_regression() function to create a synthetic regression problem with 1,000 examples and 20 input features.

The complete example is listed below.

```
# test regression dataset
```

```
from sklearn.datasets import make_regression
# define dataset
X, y = make_regression(n_samples=1000, n_features=20, n_informative=15, noise=0.1, random_state=
# summarize the dataset
print(X.shape, y.shape)
```

Running the example creates the dataset and summarizes the shape of the input and output components.

```
(1000, 20) (1000,)
```

Now we can evaluate a suite of different machine learning models on the dataset.

Specifically, we will evaluate the following three algorithms:

- Nearest Neighbors.
- Decision Tree.
- Support Vector Regression.

Note: The test dataset can be trivially solved using a l… …d using a linear model under the covers. As such, we wi… …demonstrate the benefit of the stacking ensemble met… …ake the benefit of the stacking with.

Each algorithm will be evaluated using the default mo… below creates the models we wish to evaluate.

```
# get a list of models to evaluate
def get_models():
    models = dict()
    models['knn'] = KNeighborsRegressor()
    models['cart'] = DecisionTreeRegressor()
    models['svm'] = SVR()
    return models
```

Each model will be evaluated using repeated k-fold cross-validation. The *evaluate_model()* function below takes a model instance and returns a list of scores from three repeats of 10-fold cross-validation.

```
# evaluate a given model using cross-validation
def evaluate_model(model, X, y):
    cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
    scores = cross_val_score(model, X, y, scoring='neg_mean_absolute_error', cv=cv, n_jobs=-1, e
    return scores
```

We can then report the mean performance of each algorithm and also create a box and whisker plot to compare the distribution of accuracy scores for each algorithm.

In this case, model performance will be reported using the mean absolute error (MAE). The scikit-learn library inverts the sign on this error to make it maximizing, from -infinity to 0 for the best score.

Tying this together, the complete example is listed bel…

```
# compare ensemble to base models
from numpy import mean
from numpy import std
from sklearn.datasets import make_regression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from matplotlib import pyplot

# get the dataset
def get_dataset():
    X, y = make_regression(n_samples=1000, n_features=20, n_informative=15, noise=0.1, random_s
    return X, y

# get a list of models to evaluate
def get_models():
    models = dict()
    models['knn'] = KNeighborsRegressor()
    models['cart'] = DecisionTreeRegressor()
    models['svm'] = SVR()
    return models

# evaluate a given model using cross-validati
def evaluate_model(model, X, y):
    cv = RepeatedKFold(n_splits=10, n_repeats
    scores = cross_val_score(model, X, y, sco                                    -1,
    return scores

# define dataset
X, y = get_dataset()
# get the models to evaluate
models = get_models()
# evaluate the models and store results
results, names = list(), list()
for name, model in models.items():
    scores = evaluate_model(model, X, y)
    results.append(scores)
    names.append(name)
    print('>%s %.3f (%.3f)' % (name, mean(scores), std(scores)))
# plot model performance for comparison
pyplot.boxplot(results, labels=names, showmeans=True)
pyplot.show()
```

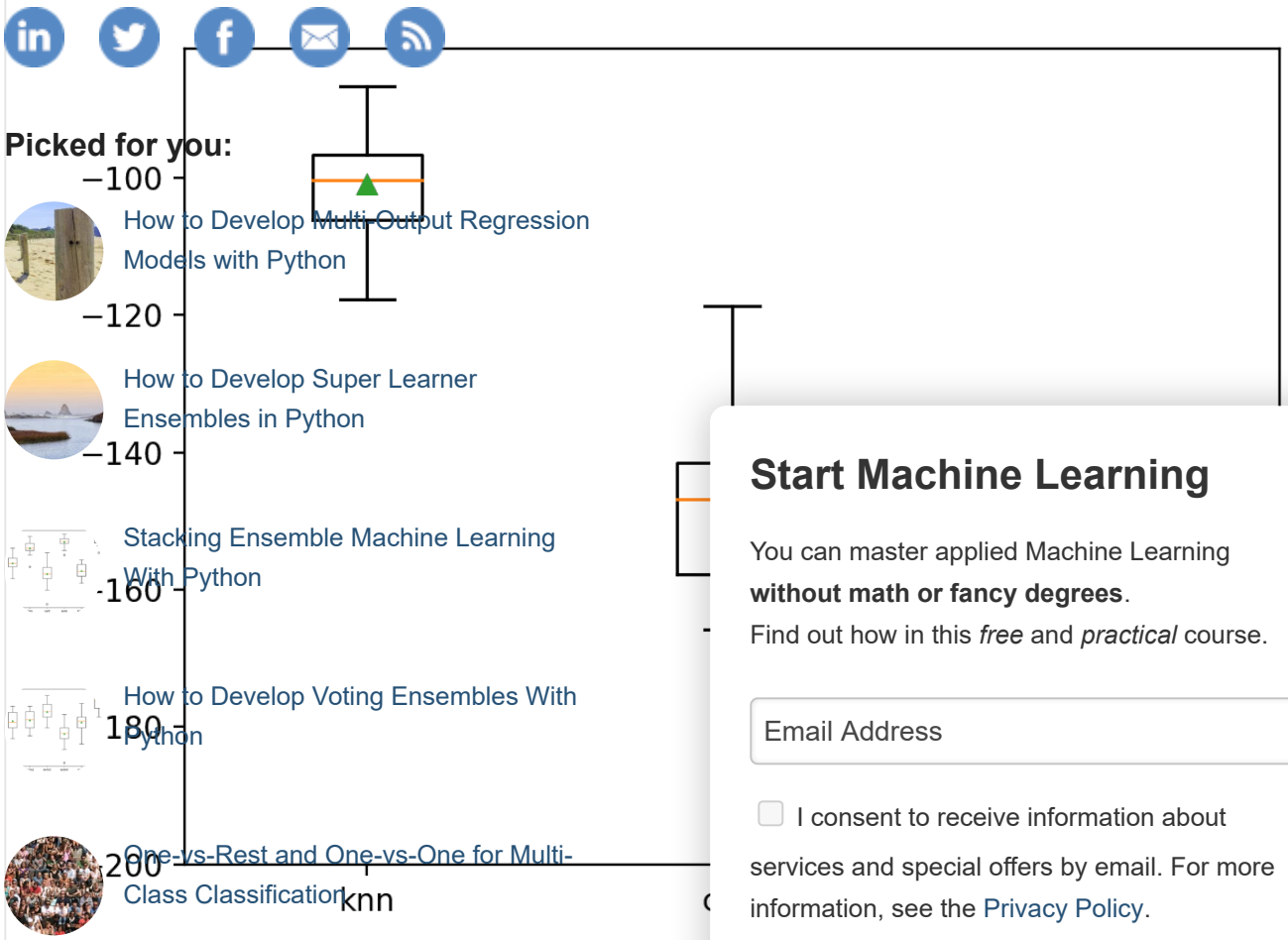

Running t ne mean and standard deviation MAE for each model.

Note: Your results may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Consider running the example a few times and compare the average outcome.

We can see that in this case, KNN performs the best with a mean negative MAE of about -100.

```
>knn -101.019 (7.161)
>cart -148.100 (11.039)
>svm -162.419 (12.565)
```

A box-and-whisker plot is then created comparing the



Box Plot of Standalone Model Negativ

Here we have three different algorithms that perform well, presumably in different ways on this dataset.

Next, we can try to combine these three models into a single ensemble model using stacking.

We can use a linear regression model to learn how to best combine the predictions from each of the separate

The ***get_stacking()*** function below defines the StackingRegressor model by first defining a list of tuples for the three base models, then defining the linear regression meta-model to combine the predictions from the base models using 5-fold cross-validation.

```
# get a stacking ensemble of models
def get_stacking():
    # define the base models
    level0 = list()
    level0.append(('knn', KNeighborsRegressor()))
    level0.append(('cart', DecisionTreeRegressor()))
    level0.append(('svm', SVR()))
    # define meta learner model
    level1 = LinearRegression()
```

```
    # define the stacking ensemble
    model = StackingRegressor(estimators=level0, final_estimator=level1, cv=5)
    return model
```


can include the stacking ensemble in the list of models to evaluate, along with the standalone models.

```
# get a list of models to evaluate
def get_models():
    models = dict()
    models['knn'] = KNeighborsRegressor()
    models['cart'] = DecisionTreeRegressor()
    models['svm'] = SVR()
    models['stacking'] = get_stacking()
    return models
```

pectation is that the stacking ensemble will perf

This is not always the case, and if it is not the case, th
ble model.

The complete example of evaluating the stacking ens
listed below.

```
# compare ensemble to each standalone models
from numpy import mean
from numpy import std
from sklearn.datasets import make_regression
from sklearn.model_selection import cross_val
from sklearn.model_selection import RepeatedK
from sklearn.linear_model import LinearRegres
from sklearn.neighbors import KNeighborsRegre
from sklearn.tree import DecisionTreeRegresso
from sklearn.svm import SVR
from sklearn.ensemble import StackingRegressor
from matplotlib import pyplot

# get the dataset
def get_dataset():
    X, y = make_regression(n_samples=1000, n_features=20, n_informative=15, noise=0.1, random_s
    return X, y

# get a stacking ensemble of models
def get_stacking():
    # define the base models
    level0 = list()
    level0.append(('knn', KNeighborsRegressor()))
    level0.append(('cart', DecisionTreeRegressor()))
    level0.append(('svm', SVR()))
    # define meta learner model
    level1 = LinearRegression()
    # define the stacking ensemble
    model = StackingRegressor(estimators=level0, final_estimator=level1, cv=5)
    return model

# get a list of models to evaluate
def get_models():
    models = dict()

    models['knn'] = KNeighborsRegressor()
```

```
    models['cart'] = DecisionTreeRegressor()
    models['svm'] = SVR()
    models['stacking'] = get_stacking()
    return models

# evaluate a given model using cross-validation
def evaluate_model(model, X, y):
    cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
    scores = cross_val_score(model, X, y, scoring='neg_mean_absolute_error', cv=cv, n_jobs=-1,
    return scores

# define dataset
X, y = get_dataset()
# get the models to evaluate
models = get_models()
# evaluate the models and store results
results, names = list(), list()
for name, model in models.items():
    scores = evaluate_model(model, X, y)
    results.append(scores)
    names.append(name)
    print('>%s %.3f (%.3f)' % (name, mean(sco
# plot model performance for comparison
pyplot.boxplot(results, labels=names, showmea
pyplot.show()
```

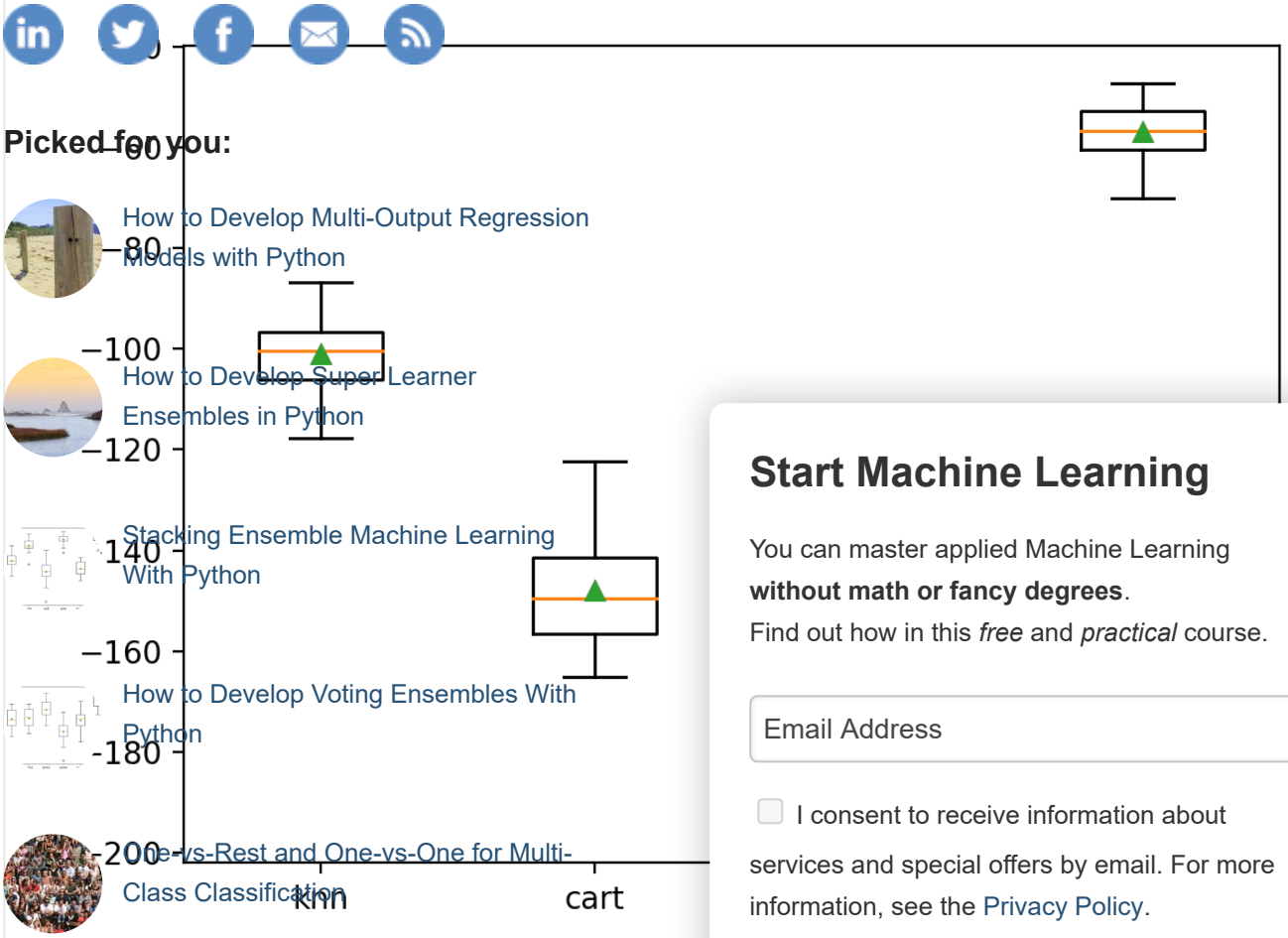

g the example first reports the performance of h
odel, then the stacking ensemble.

Note: Your results may vary given the stochastic natu
ces in numerical precision. Consider running th
e.

In this case, we can see that the stacking ensemble a
average, achieving a mean negative MAE of about -56.

```
>knn -101.019 (7.161)
>cart -148.017 (10.635)
>svm -162.419 (12.565)
>stacking -56.893 (5.253)
```

A box plo istribution of model error scores. Here, we can see that the mean and median scores for the stacking model sit much higher than any individual model.



Box Plot of Standalone and Stacking Model I

If we choose a stacking ensemble as our final model, we can fit and use it to make predictions on new data just like any other model.

First, the stacking ensemble is fit on all available data, then the *predict()* function can be called to make predictions on new data.

The example below demonstrates this on our regression dataset.

```
# make a prediction with a stacking ensemble
from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.ensemble import StackingRegressor
# define dataset
X, y = make_regression(n_samples=1000, n_features=20, n_informative=15, noise=0.1, random_state
# define the base models
level0 = list()
level0.append(('knn', KNeighborsRegressor()))
level0.append(('cart', DecisionTreeRegressor(
```

```
level0.append(('svm', SVR()))
# define meta learner model
level1 = LinearRegression()
# define the stacking ensemble
model = StackingRegressor(estimators=level0, final_estimator=level1, cv=5)
# fit the model on all available data
model.fit(X, y)
# make a prediction for one example
data = [[0.59332206,-0.56637507,1.34808718,-0.57054047,-0.72480487,1.05648449,0.77744852,0.0736
yhat = model.predict(data)
print('Predicted Value: %.3f' % (yhat))
```

Running the example fits the stacking ensemble model on the entire dataset and is then used to make a prediction on a new row of data, as we might when using the model in an application.

```
Predicted Value: 556.264
```


Further Reading

This section provides more resources on the topic if you are looking to go deeper.

Related Tutorials

- How to Implement Stacked Generalization (Stacking) From Scratch With Python
- How to Develop a Stacking Ensemble for Deep Learning Neural Networks in Python With Keras
- How to Develop Super Learner Ensembles in Python
- How to Use Out-of-Fold Predictions in Machine Learning
- A Gentle Introduction to k-fold Cross-Validation

Never miss a tutorial:

Picked for you:

How to Develop Multi-Output Regression Models with Python

How to Develop Super Learner Ensembles in Python

Papers

- Stacked Generalization, 1992.
- Stacked Generalization: When Does It Work?, 19
- Issues in Stacked Generalization, 1999.

Books

- Data Mining: Practical Machine Learning Tools ar
- The Elements of Statistical Learning, 2017.
- Machine Learning: A Probabilistic Perspective, 20

APIs

- sklearn.ensemble.StackingClassifier API.
- sklearn.ensemble.StackingRegressor API.
- sklearn.datasets.make_classification API.
- sklearn.datasets.make_regression API.

Articles

Summary

In this tutorial, you discovered the stacked generalization ensemble or stacking in Python.

Specifically, you learned:

- Stacking is an ensemble machine learning algorithm that learns how to best combine the predictions from multiple well-performing machine learning m
- The scikit-learn library provides a standard impler
- How to use stacking ensembles for regression an

have any questions?

Ask your questions in the comments below and I will c

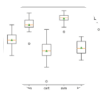
Never miss a tutorial:

Picked for you:

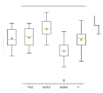
How to Develop Multi-Output Regression Models with Python

On This Topic

How to Develop Super Learner Ensembles in Python



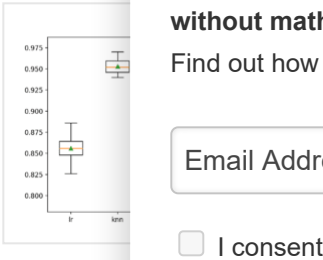
Stacking Ensemble Machine Learning With Python



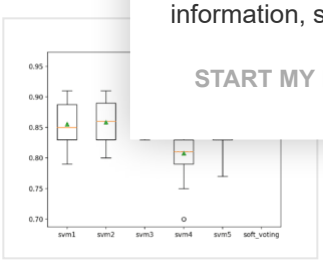
How to Develop Voting Ensembles With Python

One-vs-Rest and One-vs-One for Multi-Class Classification

Essence of Stacking Ens



Growing and Prunir



How to Develop Voting Ensembles With Python

How to Develop a Weighted Average Ensemble With Python

Never miss a tutorial:

Picked for you:

Ensemble Machine Learning With Python (7-Day Mini-Course)

How to Develop Multi-Output Regression Models with Python

Stacking Ensemble for Deep Learning Neural Networks…

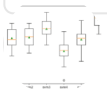
How to Develop Super Learner Ensembles in Python

Stacking Ensemble Machine Learning With Python

About Jason Brownlee

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

View all posts by Jason Brownlee →



How to Develop Voting Ensembles With Python

‹ 4 Types of Classification Tasks in Machine Learning

…ion ›

One-vs-Rest and One-vs-One for Multi-Class Classification

119 Responses to *Stacking Ensemble M…*

Pawaz Mokbal April 10, 2020 at 6:58 am #

REPLY

Thanks for this great explanation.

I have a question, how can I pass a specific type of data (e.g. specific class) to a specific algorithm, where each m… …'ar data. then gathering all models (stacking) to predict best re…

Jason Brownlee April 10, 2020 at 8:38 am #

REPLY

You can fit different models perhaps manually, then use another model to combine the predictions. E.g. all maually.

Francisco April 10, 2020 at 7:12 am #

REPLY

Hi Jason, thanks for the blog and all the information in a digested form, I am a beginner in all aspects and I can really see the advantages of using the method of stacking. I am yet to start doing my first project but before that I plan on reading as much as possible from your articles.

Thank you and be safe, I hope to learn as much as posible from you.

Cheers from Mexico!!

Jason Brownlee April 10, 2020 at 8:39 am #

REPLY

You're welcome.

Jason Brownlee April 10, 2020 at 10:11 am #

REPLY

that's a lot of coding which can be reduce into

https://github.com/IBM/AutoMLPipeline.jl

Jason Brownlee April 10, 2020 at 10:11 am #

REPLY

Thanks for sharing.

Akshay April 10, 2020 at 9:42 am #

REPLY

Great article, Jason! Keep these coming. Thanks!

Jason Brownlee April 10, 2020 at 10:11 am #

REPLY

You're welcome!

Tolga Karahan April 10, 2020 at 9:29 pm #

REPLY

Thanks for sharing. I suppose when cv=n argument provided to the StackingClassifier, it implicitly trains base models on training data and then trains StackingClassifier with predictions of base models on out-of-sample data right?

Jason Brownlee April 11, 2020 at 6:18 am #

REPLY

ds ms an internal cv process.

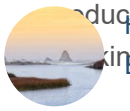
Yaniv Rotaru May 1, 2020 at 9:08 pm #

REPLY

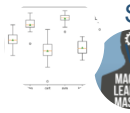
How to Develop Multi-Output Regression Models with Python

Thanks a lot for the info! though I have a question:

I have tried your stacking for classification on the given dataset, I took the first 800 samples for training and duced the box plot, however, when I test each model's accuracy results on the 200 test samples the king model doesn't provide the best results, do you have an explanation for this?



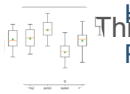
How to Develop Super Learner Ensembles in Python



Stacking Ensemble Machine Learning with Python

Jason Brownlee May 2, 2020 at 5:44 am #

Yes, stacking is not guaranteed to outperf



How to Develop Voting Ensembles With Python

This is why we must use controlled experiments as

One-vs-Rest and One-vs-One for Multi-Class Classification

Siva Sai May 16, 2020 at 5:11 am #

Hi Jason Brownlee, your articles really helped

In a paragraph you stated that " That is, data not used predictions are made, and these predictions, along with the expected outputs, provide the input and output pairs of the training dataset used to fit the meta-model."

Here what do you mean by expected outputs ??

y 16, 2020 at 6:24 am #

REPLY

I'm happy to hear that!

Expected outputs are target values in the dataset.

Siva Sai May 16, 2020 at 3:23 pm #

REPLY

So you mean that during model.fit(x_train,y_train) (training phase), the meta-model will not learn anything, only at the time of model.score(x_test,y_test) (testing phase),the meta-model trains on the test inputs and predicted values/labels made by base estimator.

Is my understand correct ?

on wnlee May 17, 2020 at 6:27 am #

REPLY

No, I don't quite follow your summary.

The stacking model is fit during the call to fit(). The call to cross_val_score() will fit and evaluate k models which internally involves calls to fit().

farubgogh May 19, 2020 at 1:04 am #

REPLY

hi Jason, thank you for your great tutorials!

I want to use CalibratedClassifier in Level0 estimators,

```
model1=XGBClassifier()
model2=…..
rated1 = CalibratedClassifierCV(model1, cv=5)
brated1.fit(X_train, y_train)
calibrated2 = ….
alibrated2.fit(…)
nators = [('xgb', calibrated1),('…', calibrated2)]
clf = StackingClassifier(estimators=estimators, final_es
```


Jason Brownlee May 19, 2020 at 6:08 am #

REPLY

Thanks.

Calibrating level0 classifiers does not make sense to me, sorry. You might have to use some trial and err

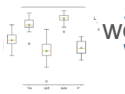
Xiao June 9, 2020 at 7:15 pm #

REPLY

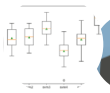
Hi,Jason!Thanks for your sharing. I used this tutorial for my binary classification task.I have 1000 image samples,and each sample has 4000 dimmensions that extrcted by CNN. However, the stacking result is lower than lr or bayes classifier, and also the super learning in your another tutorial. Could you tell me the reason? Hope for your reply.Many thanks!

How to Develop Multi-Output Regression Models with Python

How to Develop Super Learner Ensembles in Python



Stacking Ensemble Machine Learning With Python



How to Develop Voting Ensembles With Python

One-vs-Rest and One-vs-One for Multi-Class Classification

Jason Brownlee June 10, 2020 at 6:11 am #

REPLY

wer why estions it too hard/intractable.

Often the best we can do is use controlled experiments and present results to support decision of what model or modeling pipeline works well/best.

Vinayak July 10, 2020 at 12:50 am #

REPLY

Thanks for the wonderful tutorial.

ee in this tutorial each base learner evaluated using
implement the base learner with hyperparameters tuni

we use boosting and bagging classifiers in level 0

Yes, tuning is a good idea.

You can use ensembles as base models if you like

Jason Brownlee July 10, 2020 at 6:01 am #

REPLY

Thanks!

Vinayak July 10, 2020 at 7:00 pm #

REPLY

Thank you.

1. From below piece of code which you used in this tutorial, I believe we are getting accuracy on Test data set, please confirm?

```
# evalu                                    ss-validation
def evaluate_model(model):
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(model, X, y, scoring='neg_mean_absolute_error', cv=cv, n_jobs=-1,
error_score='raise')
return scores
```

2. I am working on a classification problem and please find my different classification algorithm results below.

Method Train Acuracy Test Acuracy
0 Logistic Regression 0.8504762 0.8426966
1 KNN 0.9330952 0.9550562
2 Naive Bayes 0.7882540 0.7640449

3 SVM 0.9573810 0.9775281
4 Decision Tree 1.0000000 0.9438202
5 Random Forest 0.9477778 0.9550562
6 Stacking 1.0000000 0.9775281

Base learners – 0 to 5, Meta model – 6

If you look at Stacking classifier, it is completely over-fit on training data and I don't think this is best model to consider. Is that right?

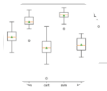
How to Develop Multi-Output Regression Models with Python

e: I tried removing Decision tree from my base learners list because it's over-fit on training data and but still I get below results from stacking classifier.So again this is not best model, is that right?

How to Develop Super Learner Ensembles in Python

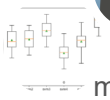
uracy of Training data set: 1.0000 %
uracy of Test data set: 0.9888 %

Perhaps I will go for SVM as my final model.



Stacking Ensemble Machine Learning With Python

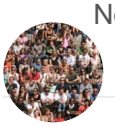
Jason Brownlee July 11, 2020 at 6:09 am #



How to Develop Voting Ensembles With Python

There is no train/test sets. Instead, we are
model performance.

Not sure I would agree it is overfit. You can focus o



One-vs-Rest and One-vs-One for Multi-Class Classification

Vinayak July 11, 2020 at 2:52 pm #

Sorry, could you please clarify what is "hold out set performance"?

Also, please answer below question.

1. From below piece of code which is used in this tutorial, I believe we are getting accuracy on Test data set, please confirm?

evalu ss-validation
def eva
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(model, X, y, scoring='neg_mean_absolute_error', cv=cv, n_jobs=-1, error_score='raise')
return scores

Jason Brownlee July 12, 2020 at 5:42 am #

REPLY

No, we are using cross-validation.

Yes, you can learn more about how cross-validation works here:
https://machinelearningmastery.com/k-fold-cross-validation/

Y can m abo t of fold predictions (on the hold out set) here:
https://machinelearningmastery.com/out-of-fold-predictions-in-machine-learning/

Alex August 12, 2020 at 5:24 pm #

REPLY

Hello,

t of all, great guide. I found it really useful.

se predict_proba for the stacking classifier, will it u
level1 model? Or will it use only the ones from each te
making a lot of sense. I am looking at this guide and th
derstand if it is the same idea, or if it isnt, which o
pipeline that also does feature selection?

Im a bit new to machine learning and python in genera

nks a lot!

Jason Brownlee August 13, 2020 at 6:07 am

Thank you.

Regardless of predicting labels or probabilities, the
the base models.

Does that help?

gust 16, 2020 at 1:38 am #

REPLY

Yes, thanks. So, tell me if I understand corectly because I want to be sure: in this guide the meta model is fitted on the out of fold predictions, just like in your other guide for out of folds. Im asking because in this guide you dont create the meta dataset like in that one. Is this being done by the stacking classifier?

Jason Brownlee August 16, 2020 at 5:56 am #

REPLY

Correct on both accounts.

The sklearn hides the complexity away an
love it!

Kat[illegible]k D[illegible]lan August 22, 2020 at 1:54 am #

REPLY

Working on a multiclass classification problem and the dataset is imbalanced I have used SMOTE to over sample. Using xgboost I could achieve an f1 score of about 90.67 affer tuning as well. Would stacked classifiers along with xgboost help increase the f1 score.

Jason Brownlee August 22, 2020 at 6:17 am #

REPLY

Perhaps try it and see.

Salome August 22, 2020 at 9:45 am #

Thank you Mr. Jason for your guide. I used th[illegible] ic datasets. For the hyperparameter tunning I use[illegible] del? How will I use it to predict outcome on the test

Thank you.

Jason Brownlee August 22, 2020 at 1:33 pm

You're welcome.

Well done! The gridsearchcv will provide access to the best configuration as follows:


```
...
# summarize best
print('Best MAE: %.3f' % results.best_score_)
print('Best Config: %s' % results.best_params_)
```

You can then fit a new model using the printed configuration, fit your model on all available data and call predict() for new data.

If making predictions is new for you, see this:
https://machinelearningmastery.com/make-predictions-scikit-learn/

Anthony The Koala August 30, 2020 at 1:38 am #

REPLY

Dear Dr Jason,
Again many thanks for your tutorials.

I understand the listing directly under the heading “The complete example of evaluating the stacking ensemble model alongside the standalone models is listed below”

Wha[illegible]ve[illegible]ned[illegible]n th[illegible]del from the get_models() and **get_stacking**().

* **get_stacking** is a model.

* get_models gets a list of models. That includes the **get_stacking** which is a model.

I could use the code under “The complete example of evaluating the stacking ensemble model alongside the standalone models is listed below” to make a prediction.

making the prediction can be done underneath the lines for the boxplot.

```
model = get_stacking(); # this is a model, which is a stacked model consisting of the leve
data = [[2.47475454,0.40165523,1.68081787,2.88940715,0.91704519,-3.07950644,4.39961206,0.7
model.fit(X,y); # X and y generated earlier
model.predict(data)[0]
0
```

nclusion by following the example at “The complete
gs with the standalone models is listed below” we ca
model consisting of level 0 models and level1 LR mode

Thank you,

ony from Sydney

Jason Brownlee August 30, 2020 at 6:43 am

Thanks.

REPLY

Loving the Tutorials?

The Ensemble Learning With Python EBook is where you'll find the ***Really Good*** stuff.

>> SEE WHAT'S INSIDE

Thanks for your explain about stacking! I have a question about it. Several days ago I did a experiment that made the best params for every base model and the default params for every base model. In my opinion, if every b better performance, so that the meta model would get a higher accuracy by the training data combined by the predictions generated the base models. But in fact, the result was a little worse. So could you help me to explain the complex phenomenon?

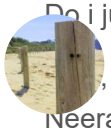
Best regards,

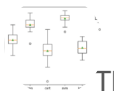
Jason Brownlee September 4, 2020 at 6:26 am #

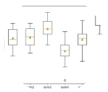
REPLY

This is common.

The reason is highly tuned models are fragile to small changes.







Neeraj Gaur September 9, 2020 at 4:23 pm # REPLY

Hi J[illegible]n

I found your script and explanation very helful and as I am new in the feild.

Not really sure how to cite your script in my manuscript.

Do i just cite it as any mormal website is cited or is there there some other method?

Neeraj Gaur

Jason Brownlee September 10, 2020 at 6:23 [illegible]

thanks!

This can help you cite a blog post:

https://machinelearningmastery.com/faq/single-faq

elham October 14, 2020 at 12:51 am # REPLY

Hello

[illegible]w to obtain feature importance from a Stacking Ens[illegible]

I tried but it was not possible.

Jason Brownlee October 14, 2020 at 6:20 am # REPLY

Off hand, I don't think stacking offers this capability.

elham October 26, 2020 at 11:28 pm # REPLY

How do you think I can do this?

I am a beginner, please help me how to write the code

Jason Brownlee October 27, 2020 at 6:45 am # REPLY

Yes!

This will help you install what you need:

https://machinelearningmastery.com/setup-python-

anaconda/

Never miss a tutorial:

This will help you copy the code:

https://machinelearningmastery.com/faq/single-faq/how-do-i-copy-code-from-a-tutorial

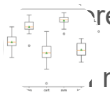
This will help you run it:

https://machinelearningmastery.com/faq/single-faq/how-do-i-run-a-script-from-the-command-line

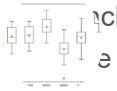
Picked for you:

How to Develop Multi-Output Regression Models with Python

How to Develop Super Learner Ensembles in Python



Stacking Ensemble Machine Learning With Python



How to Develop Voting Ensembles With Python

One-vs-Rest and One-vs-One for Multi-Class Classification

Linda October 30, 2020 at 4:32 am #

REPLY

Hi Jason

nk you very much for your tutorials. Please help m

I want to stack 3 different algorithms after performing f

rent features.

need to train my sub-models using same features

The response variable is crop yield. I have 24 different

cMSE) takes 8 variables out of the 24. Lasso take

e models if I want to stack them?

Jason Brownlee October 30, 2020 at 6:59 am

No, as long as the 3 models have the san

data any way you like for each model.

Loving the Tutorials?

The Ensemble Learning With Python EBook is where you'll find the ***Really Good*** stuff.

Linda October 30, 2020 at 9:42 am #

REPLY

Thank you for your prompt reply…But, I am not sure if you understand what I am saying.

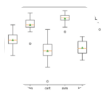
nt variables. Feature selection resulted in 8 variables for Random Forest, 10 variables for lasso, and 6 for XGBoost…..my question is: Do I need to use the same number and the same independent variables in each of my 3 models before I stack the models.

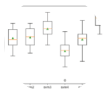
my best-fit models(RF, Lasso, & XGB) were trained on different independent variables: RF with 8 independent variables, Lasso with 10 independent variables, and XGB with 6 independent variables…. I am not referring to data splitting.

QUESTION: .do I need to use the same independent variables across the 3 models? Instead of 8, 6, and 10…must I train the models using the same independent variables…same across the 3 models??

The response variable is Corn Yield. RF is trained with 8 variables: Fertilizer, rainfall, temperature, seed type, Field size, Altitude, Slope, soil pH, F

Must I use the same variables in all the 3 sub-models?….is it ok to train the other models with 10 and 6 variables as I have explained above?





Linda October 30, 2020 at 9:48 am #

REPLY

Random Forest drew 8variables out c

Lasso selected 10 variables out of 24

Xgb selected 10 variables out of 24

The 24 has nothing to do with the 8,the 6, and
variables. Different feature selection methods
respectively

Jason Brownlee October 30, 2020

Yes, that is fine. As long as you test the model on data not used during training.

mark November 15, 2020 at 4:05 am #

REPLY

Thanks very you much for the detailed explanation on Stacking; it was very helpful to me.

Please I have a question concerning this paragraph in your article:

“The meta-model is trained on the predictions made by base models on out-of-sample data. That is, data not used to train the base models is fed to the base models, predictions are made, and these predictions, along with the expected outputs, provide the input and output pairs of the training dataset used to fit the meta-model.”

From the above paragraph, this is my understanding:

1. Given a training dataset, (X_{train}, Y_{train}), during training the base-models are only fed with X_{train}.

2. The base models make predictions, Xbase_{predicti

3. Xbase_{prediction} is then fed to the meta-model as input. However, it will also have the desired output for the predictions Y_{train}. This way, the meta-model is fit on the training dataset.

Please is my understanding of the above paragraph in your article right?

Thanks

Mark

REPLY

Jason Brownlee November 15, 2020 at 6:30 am #

Cross-validation is used where the out of input for the stacked model.

REPLY

mark November 16, 2020 at 9:34 am #

Okay, thanks.

REPLY

PWB December 1, 2020 at 6:53 pm #

Let's say that I have used RandomSearchCV but I set the scoring parameter to optimize for 'precisio am unharmed by the existence of false negatives).

The docs for StackingClassifier don't include a scoring parameter. The docs only mention training the ensemble to get the best *accuracy*.

Does this mean that StackingClassifier will optimize for a metric I don't want to optimize (accuracy) instead of the metric I do want to optimize (precision)?

REPLY

Jason Brownlee December 2, 2020 at 7:40 am #

If that is the case predict the positive class in all cases and achieve the best precision.

The stacking ensemble itself does not optimize a score. I think you are referring to the score() function, which is one way to evaluate the model.

REPLY

sukhpal December 2, 2020 at 11:21 pm #

how to remove this error as sklearn is already

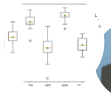
No module named 'sklearn.cross_validation'

Never miss a tutorial:

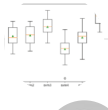
Picked for you:

How to Develop Multi-Output Regression Models with Python

How to Develop Super Learner Ensembles in Python



Stacking Ensemble Machine Learning With Python



How to Develop Voting Ensembles With Python

One-vs-Rest and One-vs-One for Multi-Class Classification

Jude Leonard December 20, 2020 at 6:05 pm #

REPLY

Thank you very much for this. This is no doubt the best way I've seen it explained

Jason Brownlee December 21, 2020 at 6:36

Thanks!

Muhammad January 26, 2021 at 11:59 pm #

REPLY

Instead of cross validation cv=5 can we use tr

Jason Brownlee January 27, 2021 at 6:08 am #

REPLY

If you like

It would be called blending:
https://machinelearningmastery.com/blending-ensemble-machine-learning-with-python/

Pablo January 27, 2021 at 7:48 pm #

REPLY

Hi Jason,

How it would be the best way for ensemble 3 Random Forests with 3 different datasets? Could you help me?

Thanks!

Pablo January 27, 2021 at 7:49 pm #

REPLY

Sorry again, the problem has 3 classes

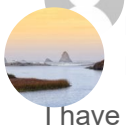
REPLY

Jason Brownlee January 28, 2021 at 5:55 am #

Perhaps start using an average of the models.

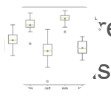
REPLY

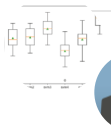
Pablo January 27, 2021 at 8:04 pm #



Hi Jason

I have an imbalanced dataset multiclass, and I have ba
methods for feature selection. Thus, i want to do 3 Ran
rent column number and after ensemble this three
se?





Jason Brownlee January 28, 2021 at 5:56 am

Evaluate a suite of approaches and disco

Fu January 29, 2021 at 4:09 pm #

Hi Jason Brownlee

can i ask how to use RMSE to evaluate the results?
could give me some example?

REPLY

uary 30, 2021 at 6:32 am #

See this tutorial on how to calculate metrics like RMSE:
https://machinelearningmastery.com/regression-metrics-for-machine-learning/

REPLY

David Stein February 5, 2021 at 3:08 am #

Hi Dr. Brownlee,

Thank you for your excellent tutorials. I am working on a stacking architecture and I'm stuck on a particular idea. All of my level 0 models require different threshh
threshholds individually in a stacking architecture or to

Thanks!

Jason Brownlee February 5, 2021 at 5:47 am #

REPLY

You're welcome.

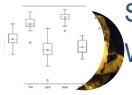
How to Develop Multi-Output Regression Models with Python

Not as far as I know.

You can implement the stacking manually and use threshold values that pass on to the next level.

Alternately, you could drop your thresholds and let the level 1 model learn how to best use the predictions from level 0.

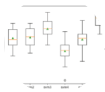
How to Develop Super Learner Ensembles in Python



Stacking Ensemble Machine Learning With Python

bryson_je February 13, 2021 at 7:15 am #

Thanks for sharing



How to Develop Voting Ensembles With Python

Jason Brownlee February 13, 2021 at 8:27 a

One-vs-Rest and One-vs-One for Multi-Class Classification

You're welcome!

Balaji February 16, 2021 at 3:01 pm #

Thanks for sharing such a wonderful tutorial. How do you save the model for later use? Joblib or pickle gives an error when trying to reuse the model to predict in another code.

ɔruary 16, 2021 at 3:12 pm #

REPLY

Good question, this will show you how:
https://machinelearningmastery.com/save-load-keras-deep-learning-models/

Angelos Chatzimparmpas February 26, 2021 at 9:45 pm #

REPLY

Thank you Jason for this amazing article. It helped me a lot with my research.

I would like to leave here a link about a paper we wrote related to Stacked Generalization. Specifically, it is about how you could combine base models with the guidance from visualizations in order to build powerful and diverse Stacking ensembles.

StackGenVis: https://doi.org/10.1109/TVCG.2020.3030352

Jason Brownlee February 27, 2021 at 6:02 am #

REPLY

Thanks for sharing.

Dalila March 2, 2021 at 7:45 am #

REPLY

Hi Jason,

Thanks for the tutorial it's really interesting.

Could you do a tutorial to show how to Random Search

regression please ?

Thanks !

Jason Brownlee March 2, 2021 at 8:15 am #

Thanks for the suggestion!

Dalila March 3, 2021 at 4:57 am #

You're welcome !

Keren March 19, 2021 at 2:26 am #

REPLY

Hi Jason! Thank you very much for your tutorial!

I have a base model that requires evaluation set for early stopping, how would I used it inside the stacked ensemble?

Jason Brownlee March 19, 2021 at 6:24 am #

REPLY

You might just have to use a separate dataset as the validation set.

Keren April 22, 2021 at 7:33 pm #

REPLY

thanks Jason

just to clarify- you mean to use a seperate validation to find the right number of estimators required before and right? and then run the stack model using this number

Jason Brownlee April 23, 2021 at 5:01 am #

REPLY

How to Develop Multi-Output Regression Models with Python

Perhaps. Some experimentation may be required.

How to Develop Super Learner Ensembles in Python

Anthony The Koala April 4, 2021 at 2:08 am #

REPLY

Dear Dr Jason,

I varied the number of folds = cv from 2 to 20 in the line

Stacking Ensemble Machine Learning With Python

```
model = StackingClassifier(estimators=level0
```

I found that there was very little variation in mean a

How to Develop Voting Ensembles With Python

n varied between 0.962 when cv=2, and 0.964 wh

Thank you

Anthony of Sydney

One-vs-Rest and One-vs-One for Multi-Class Classification

Jason Brownlee April 4, 2021 at 6:53 am #

Nice work! Sounds like the system is stable.

Anthony The Koala April 4, 2021 at 2:57 am #

REPLY

I combined the evaluation and prediction for each model by combining the last two segments of the stacked regressor:

```
for name, model in models.items():
    scores = evaluate_model(model, X, y)
    results.append(scores)
    names.append(name)
    print('>%s %.3f (%.3f)' % (name, mean(scores), std(scores)))
    #print('predicting the output for %s'  % (name))
    model.fit(X, y)
    # make a prediction for one example
    data = [[0.59332206,-0.56637507,1.34808718,-0.57054047,-0.72480487,1.05648449,0.77744
    yhat = model.predict(data)
    print('Predicted Class: %d' % (yhat)
,
```

The respective output produced significantly different predictions:

```
>knn -101.019 (7.161)
Predicted Class: 320
>cart -148.432 (10.934)
Predicted Class: 419
>svm -162.419 (12.565)
Predicted Class: 31
>stacking -56.749 (5.404)
Predicted Class: 556
```

Range of predictions from 31 to 556

on. Have to use a range of models with the least score. In this case the stacked regression model uced the smallest score.

Thank you,

hony of Sydney

Jason Brownlee April 4, 2021 at 6:53 am #

Thanks for sharing.

Anthony The Koala April 4, 2021 at 3:12 am #

Dear Dr Jason,

Another observation on when to use the RepeatedKFold and RepeatedStratifiedKFold when using resampling techniques for cv.

RepeatedKFold – use for output which is continuous, eg for stacked regressor

RepeatedStratifiedKFold – use for output which is discrete, eg for stacked classifier

Thank you,

Anthon

Jason Brownlee April 4, 2021 at 6:54 am #

REPLY

Yes, these descriptions are likely in the API docs.

Anjula Paulus April 27, 2021 at 6:31 am #

REPLY

Hi, love your work. Is it possible to use stack regressor with pretrained regression models such as XGBoost and Catboost. I am working on optimizing pre

when it comes to errror metrics mae, mape. How could i combine it to get an optimized prediction? Is is possible to use pretrained models? Would linear regression work?

REPLY

Jason Brownlee April 28, 2021 at 5:53 am #

Thanks.

You may have to write custom code, I don't think sklearn can handle pre-trained models.

Anjula Paulus May 2, 2021 at 5:38 am #

Hi, Can you point in the direction on h
concept. Any code that i could refer? Thanks i

Jason Brownlee May 3, 2021 at 4:5

Sorry, I don't think I have example
prototypes in order to discover the best pa

varsha April 28, 2021 at 5:45 pm #

Hi,
Can we display confusion matrix after applying stacking ? if yes how ? As here only accuracy parameter is displayed if we want to display other performance metrices how to get these?

REPLY

>> SEE WHAT'S INSIDE ril 29, 2021 at 6:24 am #

Yes, use the entire model to make predictions on new data and calculate the confusion matrix for the predictions.

REPLY

John Lee May 3, 2021 at 12:03 am #

Great lesson! Thanks.

REPLY

Jason Brownlee May 3, 2021 at 4:57 am #

REPLY

Abdul May 10, 2021 at 11:55 pm #

How to Develop Multi-Output Regression Models with Python

Hi Jason,

Thank you so much for the guidance. this was indeed helpful to me. However, I tried implementing stacking ensemble classification on IsolationForest, OCSVM, and LOF.

How to Develop Super Learner Ensembles in Python

I usually got the following error.

"ValueError: The estimator IsolationForest should be a

Stacking Ensemble Machine Learning With Python

uestion is: Does that mean Isolation Forest and O

regression, SVM, Decision Tree classifiers?

Hope someone will respond as soon as possible. Thar

How to Develop Voting Ensembles With Python

REPLY

Jason Brownlee May 11, 2021 at 6:43 am #

One-vs-Rest and One-vs-One for Multi-Class Classification

Those models are not classifiers, you may

REPLY

Keren June 8, 2021 at 6:52 pm #

Hi Jason

Your Ensemble Learning With Python ebook is where you'll find the ***Really Good*** stuff.

Thanks!

do you have any insights on how to tune the hyperparameters of a stacking model? (for eg. if the base models need to be tuned first and then the meta model).

thanks.

REPLY

Jason Brownlee June 9, 2021 at 5:42 am #

You're welcome!

Yes, it would just be different level 0 models and different level 1 models.

MK July 26, 2021 at 6:30 pm #

Great post! One question – If I wish to check the statistical significance between performance (accuracy) differences between each base model & stacked model, how would I do that? That is, if we have 5 base models, then how significantly does each of these 5 base models differ from stacked model performance? Please guide. Thanks

REPLY

Jason Brownlee July 27, 2021 at 5:05 am #

Thanks!

This will help:
https://machinelearningmastery.com/statistical-significance-tests-for-comparing-machine-learning-algorithms/

How to Develop Super Learner Ensembles in Python

Stacking Ensemble Machine Learning With Python

How to Develop Voting Ensembles With Python

One-vs-Rest and One-vs-One for Multi-Class Classification

Bhavani July 27, 2021 at 11:33 pm #

Good evening sir,
I am working on stacked ensemble learning. C
challenges about this algorithm in classificatio

Jason Brownlee July 28, 2021 at 5

I don't know. Perhaps you can rea

REPLY

Assaf August 25, 2021 at 12:22 am #

Great content, as always!

I believ this algorithm is that the parameters of the meta-model and each of the base models, are estimated separately (in our example there are six different optimizations). Single optimization for the whole structure will probably improve accuracy. How can we implement this approach?

thanks,
Assaf

REPLY

Adrian Tam August 25, 2021 at 6:09 am #

I don't think scikit-learn allows this. After all, having all models independent to each other in training is the idea of ensemble.

print('>%s %.3f (%.3f)' % (name, mean(scores), std(scores)))

not all arguments converted during string formatting.

se how can I resolved this challenge?

REPLY

Adrian Tam September 11, 2021 at 6:12 am #

Just do print(name, mean(scores), std(sc

REPLY

Abbas Yakubu Yahaya September 11, 2021 at 5

Thank you sir.

ly it works for me.

Nut November 2, 2021 at 3:55 am #

Instead of using cross_val_score(), how can y scoring metrics?

REPLY

Adrian Tam November 7, 2021 at 7:09 am #

ps://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Look at the "scoring" parameter.

REPLY

Jian Wang January 20, 2022 at 1:44 am #

If base learner A significantly outperform base learner B, whether the stacking cannot further improve the performance. Here, 'significantly' means that the ROC curve of A enclosed that of B.

Never miss a tutorial:

Leave a Reply

Picked for you:


How to Develop Multi-Output Regression Models with Python


How to Develop Super Learner Ensembles in Python


Stacking Ensemble Machine Learning With Python


How to Develop Voting Ensembles With Python


One-vs-Rest and One-vs-One for Multi-Class Classification

Name (required)

Email (will not be published) (requ

SUBMIT COMMENT

Welcome!

I'm *Jason Brownlee* PhD and I **help developers** get results with **ma**

Read more

Loving the Tutorials?

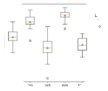
The Ensemble Learning With Python EBook is where you'll find the ***Really Good*** stuff.

>> SEE WHAT'S INSIDE

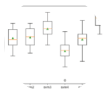
Picked for you:

How to Develop Multi-Output Regression Models with Python

How to Develop Super Learner Ensembles in Python



Stacking Ensemble Machine Learning With Python



How to Develop Voting Ensembles With Python

One-vs-Rest and One-vs-One for Multi-Class Classification

Never miss a tutorial:

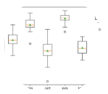
dIn itte ceb Ne ter | RSS

Privacy | Disclaimer | Terms | Contact | Sitemap | Search

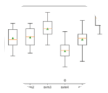
Picked for you:

How to Develop Multi-Output Regression Models with Python

How to Develop Super Learner Ensembles in Python



Stacking Ensemble Machine Learning With Python



How to Develop Voting Ensembles With Python

One-vs-Rest and One-vs-One for Multi-Class Classification