Feature Importance and Feature Selection With XGBoost in Python

Brownlee on August 31, 2016 in XGBoost

Updated on August 27, 2020

A benefit of using ensembles of decision tree methods like gradient boosting is that they can automatically provide estimates of feature importance from a trained predictive model.

In this post you will discover how you can estimate the importance of features for a predictive modeling problem using the XGBoost library in Python.

After reading this post you will know:

- How feature importance is calculated using the gradient boosting algorithm.
- How to plot feature importance in Python calculated by the XGBoost model.
- How to use the calculated by XGBoost to perform feature selection.

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

Let's get started.

- **Update Jan/2017**: Updated to reflect changes in scikit-learn API version 0.18.1.
- **Update Mar/2018**: Added alternate link to download the dataset as the original appears to have been taken down.
- **Update Apr/2020**: Updated example for XGBoost 1.0.2.

Feature Importance and Feature Selection With XGBoost in Python
Photo by Keith Roper, some rights reserved.

Feature Importance in Gradient Boosting

A benefit of using gradient boosting is that after the boosted trees are constructed, it is relatively straightforward to retrieve importance scores for each attribute.

Generally, importance provides a score that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model. The more an attribute is used to make key decisions with decision trees, the higher its relative importance.

This importance is calculated explicitly for each attribute in the dataset, allowing attributes to be ranked and compared to each other.

Importance is calculated for a single decision tree by the amount that each attribute split point improves the performance measure, weighted by the number of observations the node is responsible for. The performance measure may be the purity (Gini index) used to select the split points or another more specific error function.

The feature importances are then averaged across all of the the decision trees within the model.

For more technical information on how feature importance is calculated in boosted decision trees, see Section 10.13.1 "*Relative Importance of Predictor Variables*" of the book The Elements of Statistical Learning: Data Mining, Inference, and Prediction, page 367.

Also, see Matthew Drury answer to the StackOverflow question "Relative variable importance for [illegible]" where he provides a very detailed and practical answer.

Manually Plot Feature Importance

A trained XGBoost model automatically calculates feature importance on your predictive modeling problem.

Importance scores are available in the **feature_importances_** member variable of the trained model. For example, they can be printed directly as follows:

```
print(model.feature_importances_)
```

We can plot these scores on a bar chart directly to get a visual indication of the relative importance of each feature in the dataset. For example:


```
# plot
pyplot.bar(range(len(model.feature_importances_)), model.feature_importances_)
pyplot.show()
```

We can d[illegible] an XGBoost model on the Pima Indians onset of diabetes dataset and creating a bar chart from the calculated feature importances.

Download the dataset and place it in your current working directory.

- Dataset File.
- Dataset Details.

```
# plot feature importance manually
from numpy import loadtxt
from xgboost import XGBClassifier
from matplotlib import pyplot
# load data
dataset = loadtxt('pima-indians-diabetes.csv', delimiter=",")
# split data into X and y
X = dataset[:,0:8]
y = dataset[:,8]
# fit model no training data
model = XGBClassifier()
model.fit(X, y)
# feature importance
```

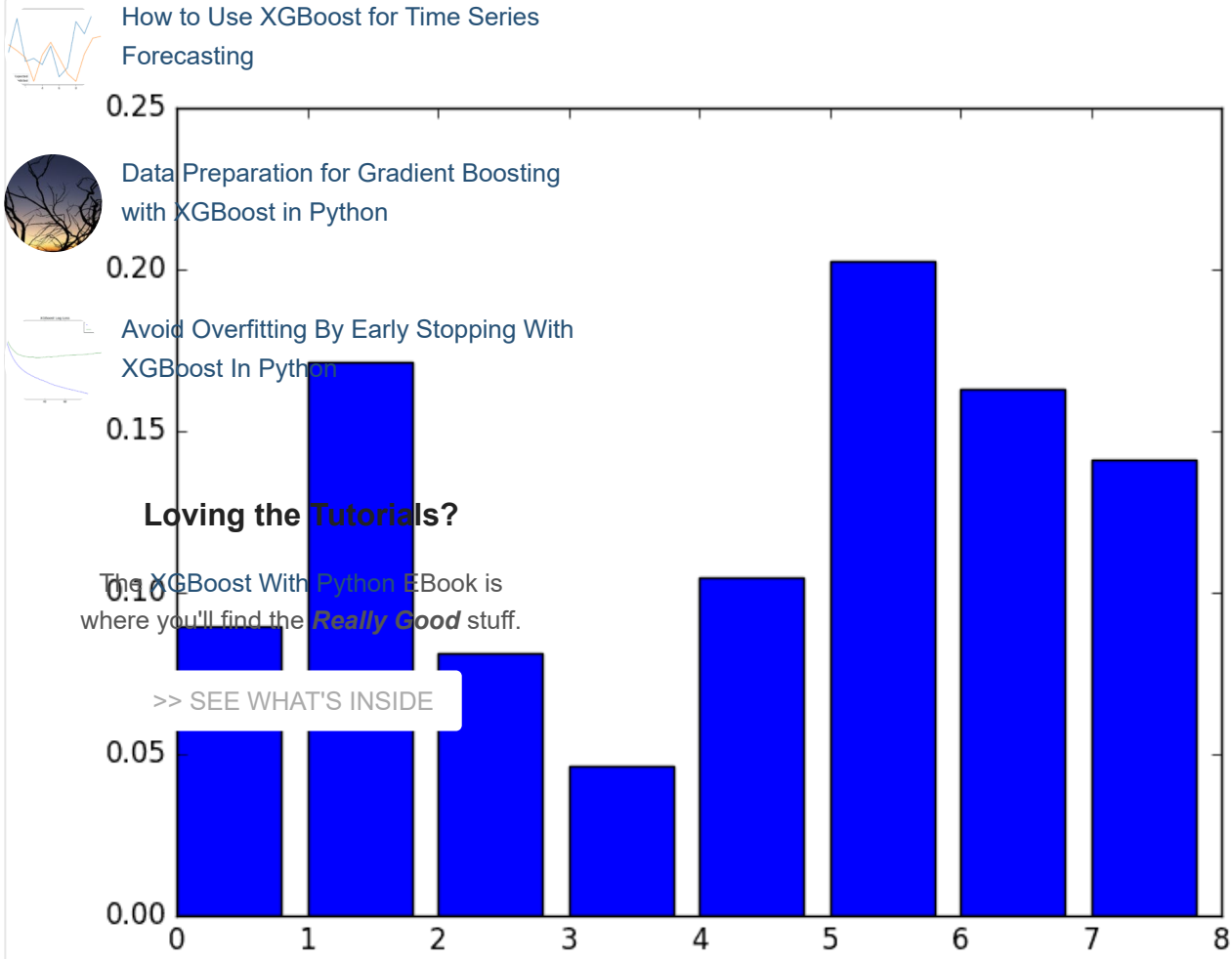
```
print(model.feature_importances_)
# plot
pyplot.bar(range(len(model.feature_importances_)), model.feature_importances_)
pyplot.show()
```

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.

Running this example first outputs the importance scores.

```
[ 0.089701  0.17109634  0.08139535  0.04651163  0.10465116  0.2026578  0.1627907   0.1411960
```

We also get a bar chart of the relative importances.



Manual Bar Chart of XGBoost Feature Importance

A downside of this plot is that the features are ordered by their input index rather than their importance. We could sort the features before plotting.

Thankfully, there is a built in plot function to help us.

Using the Built-in XGBoost Feature Importance Plot

The XGBoost library provides a built-in function to plot features ordered by their importance.

The function is called **plot_importance()** and can be used as follows:

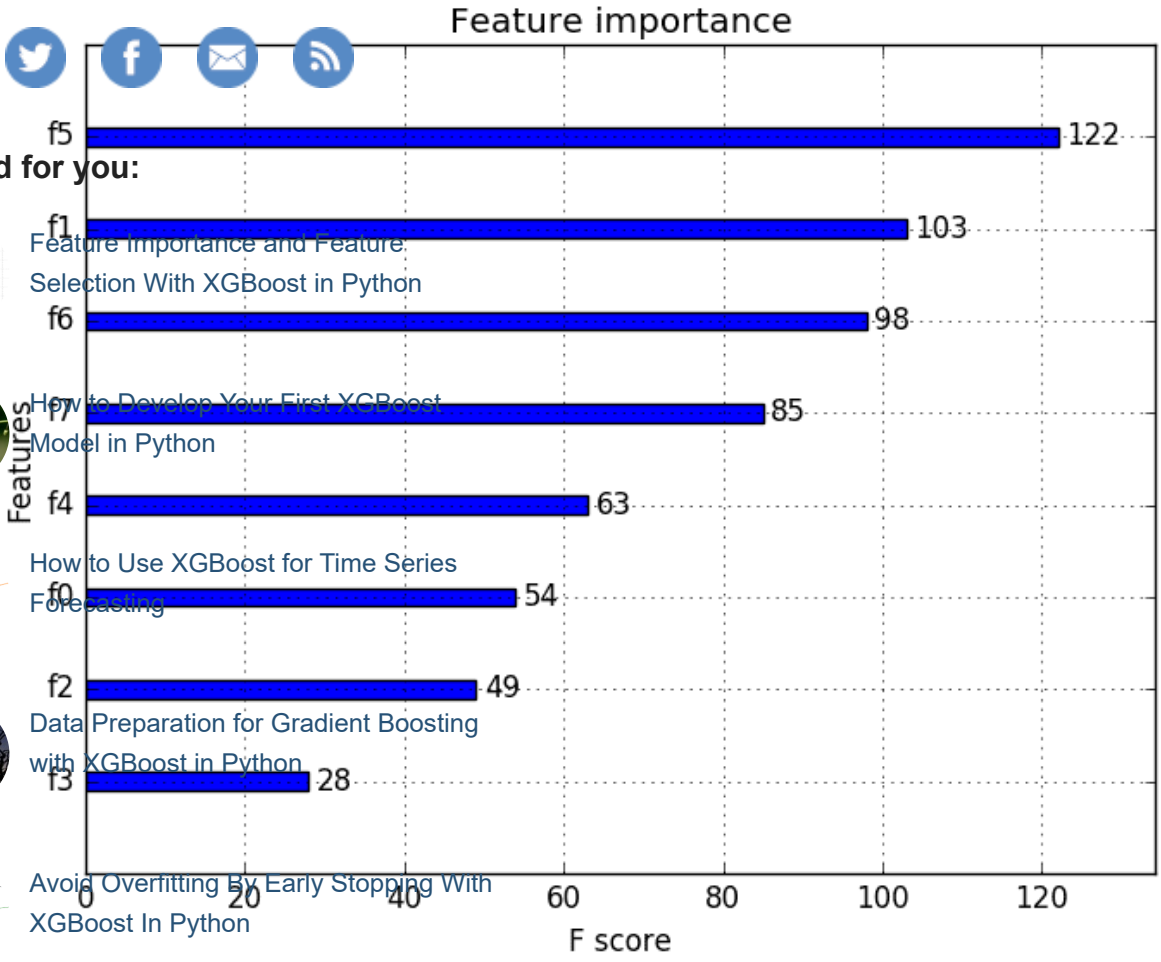
```
# plot feature importance
plot_importance(model)
pyplot.show()
```

For example, below is a complete code listing plotting the feature importance for the Pima Indians dataset using the built-in **plot_importance()** function.

```
# plot feature importance using built-in function
from numpy import loadtxt
from xgboost import XGBClassifier
from xgboost import plot_importance
from matplotlib import pyplot
# load data
dataset = loadtxt('pima-indians-diabetes.csv', delimiter=",")
# split data into X and y
X = dataset[:,0:8]
y = dataset[:,8]
# fit model no training data
model = XGBClassifier()
model.fit(X, y)
# plot feature importance
plot_importance(model)
pyplot.show()
```


Note: You may see slightly different results 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.

Running the example gives us a more useful bar chart.



XGBoost Feature Importance Bar Chart

You can see that features are automatically named according to their index in the input array (X) from F0 to F7.

Manually mapping these indices to names in the problem description, we can see that the plot shows F5 (body mass index) has the highest importance and F3 (skin fold thickness) has the lowest importance.

Feature Selection with XGBoost Feature Importance Scores

Feature importance scores can be used for feature selection in scikit-learn.

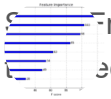
This is done using the SelectFromModel class that takes a model and can transform a dataset into a subset with selected features.

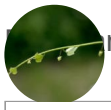
This class can take a pre-trained model, such as one trained on the entire training dataset. It can then use a threshold to decide which features to select. This threshold is used when you call the **transform()**

method on the **SelectFromModel** instance to consistently select the same features on the training dataset and the test dataset.

In the example below we first train and then evaluate an XGBoost model on the entire training dataset and test datasets respectively.

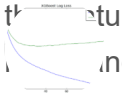
Using the feature importances calculated from the training dataset, we then wrap the model in a SelectFromModel instance. We use this to select features on the training dataset, train a model from the selected subset of features, then evaluate the model on the testset, subject to the same feature selection scheme.






```
# select features using threshold
selection = SelectFromModel(model, threshold=thresh, prefit=True)
select_X_train = selection.transform(X_train)
# train model
selection_model = XGBClassifier()
selection_model.fit(select_X_train, y_train)
# eval model
select_X_test = selection.transform(X_test)
y_pred = selection_model.predict(select_X_test)
```

For interest, we can test multiple thresholds for selecting features by feature importance. Specifically, the feature importance of each input variable, essentially allowing us to test each subset of features by importance, starting with all features and ending with a subset with the most important feature.



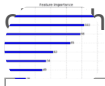
The complete code listing is provided below.


```
# use feature importance for feature selection
from numpy import loadtxt
from numpy import sort
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.feature_selection import SelectFromModel
# load data
dataset = loadtxt('pima-indians-diabetes.csv', delimiter=",")
# split data into X and y
X = dataset[:,0:8]
Y = dataset[:,8]
# split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state=7)
# fit model on all training data
model = XGBClassifier()
model.fit(X_train, y_train)
# make predictions for test data and evaluate
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
# Fit model using each importance as a threshold
thresholds = sort(model.feature_importances_)
for thresh in thresholds:
 # select features using threshold
 selection = SelectFromModel(model, threshold=thresh, prefit=True)
 select_X_train = selection.transform(X_train)
 # train model
 selection_model = XGBClassifier()
```

```
selection_model.fit(select_X_train, y_train)
# eval model
select_X_test = selection.transform(X_test)
y_pred = selection_model.predict(select_X_test)
predictions = [round(value) for value in y_pred]
accuracy = accuracy_score(y_test, predictions)
print("Thresh=%.3f, n=%d, Accuracy: %.2f%%" % (thresh, select_X_train.shape[1], accuracy*10
```

Picked for you:

Note, if you are using XGBoost 1.0.2 (and perhaps other versions), there is a bug in the XGBClassifier that results in the error:



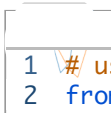
Feature Importance and Feature Selection With XGBoost in Python

```
KeyError: 'weight'
```

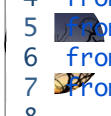
How to Develop Your First XGBoost Model in Python

This can be fixed by using a custom *XGBClassifier* class that returns *None* for the *coef_* property.

The complete example is listed below.



How to Use XGBoost for Time Series Forecasting



Data Preparation for Gradient Boosting XGBoost In Python

```
# use feature importance for feature selection, with fix for xgboost 1.0.2
from numpy import loadtxt
from numpy import sort
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.feature_selection import SelectFromModel

# define custom class to fix bug in xgboost 1.0.2
class MyXGBClassifier(XGBClassifier):
    @property
    def coef_(self):
        return None

# load data
dataset = loadtxt('pima-indians-diabetes.csv', delimiter=",")
# split data into X and y
X = dataset[:,0:8]
Y = dataset[:,8]
# split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state=7)
# fit model on all training data
model = MyXGBClassifier()
model.fit(X_train, y_train)
# make predictions for test data and evaluate
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
# Fit model using each importance as a threshold
thresholds = sort(model.feature_importances_)
for thresh in thresholds:
    # select features using threshold
    selection = SelectFromModel(model, threshold=thresh, prefit=True)
    select_X_train = selection.transform(X_train)
    # train model
    selection_model = XGBClassifier()
    selection_model.fit(select_X_train, y_train)
    # eval model
    select_X_test = selection.transform(X_test)
    predictions = selection_model.predict(select_X_test)
    accuracy = accuracy_score(y_test, predictions)
    print("Thresh=%.3f, n=%d, Accuracy: %.2f%%" % (thresh, select_X_train.shape[1], accuracy*10
```

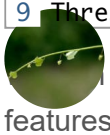
Note: Your results may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Consider running the example a few times and compare the average

outcome.

Never miss a tutorial:

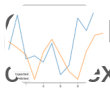
Running this example prints the following output.

```
Accuracy: 77.95%
Thresh=0.071, n=8, Accuracy: 77.95%
Thresh=0.073, n=7, Accuracy: 76.38%
Thresh=0.084, n=6, Accuracy: 77.56%
Thresh=0.090, n=5, Accuracy: 76.38%
Thresh=0.128, n=4, Accuracy: 76.38%
Thresh=0.160, n=3, Accuracy: 74.80%
Thresh=0.186, n=2, Accuracy: 71.65%
Thresh=0.208, n=1, Accuracy: 63.78%
```



How to Develop Your First XGBoost Model in Python

We can see that the performance of the model generally decreases with the number of selected features.

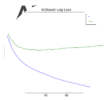


How to Use XGBoost for Time Series Forecasting

In this problem there is a trade-off of features to test set accuracy and we could decide to take a less complex model (fewer attributes such as n=4) and accept a modest decrease in estimated accuracy from 77.95% down to 76.38%.

Data Preparation for Gradient Boosting with XGBoost in Python

This is likely to be a wash on such a small dataset, but may be a more useful strategy on a larger dataset and using cross validation as the model evaluation scheme.



Avoid Overfitting By Early Stopping With XGBoost In Python

Summary

Loving the Tutorials?

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

>> SEE WHAT'S INSIDE

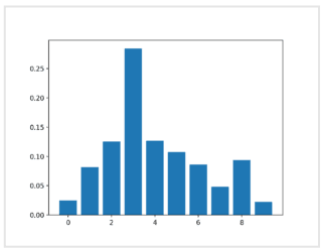
In this post you discovered how to access features and use importance in a trained XGBoost gradient boosting model.

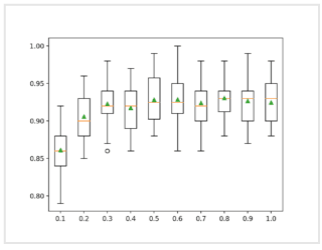
Specifically, you learned:

- What feature importance is and generally how it is calculated in XGBoost.
- How to access and plot feature importance scores from an XGBoost model.
- How to use feature importance from an XGBoost model for feature selection.

Do you have any questions about feature importance in XGBoost or about this post? Ask your questions in the comments and I will do my best to answer them.

More On This Topic

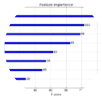

How to Calculate Feature Importance With Python


Extreme Gradient Boosting (XGBoost) Ensemble in Python

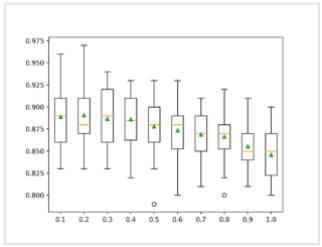
Never miss a tutorial:

Picked for you:

A Gentle Introduction to XGBoost for Applied Machine…

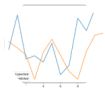


Feature Importance and Feature Selection With XGBoost in Python

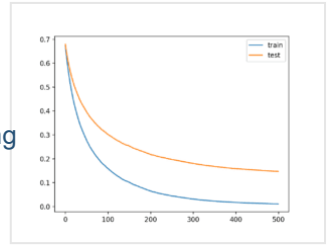


How to Develop Your First XGBoost Model in Python

How to Develop Random Forest Ensembles With XGBoost

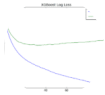


How to Use XGBoost for Time Series Forecasting



Data Preparation for Gradient Boosting with XGBoost in Python

Tune XGBoost Performance With Learning Curves



Avoid Overfitting By Early Stopping With XGBoost In Python

Loving the Tutorials?

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

A Gentle Introduction to XGBoost Loss Functions

ownlee

Jason Brownlee, PhD is a machine learning specialist who teaches developers how to get results with modern machine learning methods via hands-on tutorials.

View all posts by Jason Brownlee →

‹ How to Visualize Gradient Boosting Decision Trees With XGBoost in Python

Avoid Overfitting By Early Stopping With XGBoost In Python ›

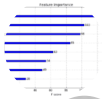
208 Responses to *Feature Importance and Feature Selection With XGBoost in Python*

Trupti December 9, 2016 at 5:23 pm #

REPLY

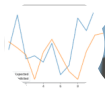
Never miss a tutorial:

Picked for you:



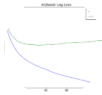
Feature Importance and Feature Selection With XGBoost in Python

How to Develop Your First XGBoost Model in Python



How to Use XGBoost for Time Series Forecasting

Data Preparation for Gradient Boosting with XGBoost in Python



Avoid Overfitting By Early Stopping With XGBoost In Python

Loving the Tutorials?

Trupti December 9, 2016 at 5:28 pm #

REPLY

sorry the error is "TypeError: only length-1 arrays can be converted to Python scalars".

Jason Brownlee December 10, 2016 at 8:04 am #

REPLY

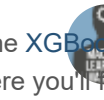
Check the shape of your X_train, e.g. print(X_train.shape)

You may need to reshape it into a matrix.

Cengiz July 3, 2021 at 8:30 am #

REPLY

How would you solve this? I have the same issue



Jason Brownlee July 4, 2021 at 5:58 am #

REPLY

This may help:

gmastery.com/index-slice-reshape-numpy-arrays-machine-learning-

sa January 5, 2017 at 3:44 pm #

REPLY

I tried to select features for xgboost based on this post (last part which uses thresholds) but since I am using gridsearch and pipeline, this error is reported:
select_X_train = selection.transform(X_train)
File "C:\Users\Markazi.co\Anaconda3\lib\site-packages\sklearn\feature_selection\base.py", line 76, in transform
mask = self.get_support()
File "C:\Users\MM.co\Anaconda3\lib\site-packages\sklearn\feature_selection\base.py", line 47, in get_support
mask = self._get_support_mask()
File "C:\Users\Markazi.co\Anaconda3\lib\site-packages\sklearn\feature_selection\from_model.py", line 201, in _get_support_mask
scores = _get_feature_importances(estimator)

File “C:\Users\Markazi.co\Anaconda3\lib\site-packages\sklearn\feature_selection\from_model.py”, line 32, in _get_feature_importances
% estimator.__class__.__name__)
ValueError: The underlying estimator method has no `coef_` or `feature_importances_` attribute. Either pass a fitted estimator to SelectFromModel or call fit before calling transform.

regards,

REPLY

Jason Brownlee January 6, 2017 at 9:05 am #

Hi sa,

Consider trying the example without Pipelines first, get it working, then try adding in additional complexity.

REPLY

sa January 6, 2017 at 5:53 pm #

Hello Mr. Brownlee
Thanks

I already tried the example without Pipelines , and it works well. After adding pipeline, it could extract feature importance but after that it fails. Thanks.

Best regards,

REPLY

Johnn January 15, 2017 at 12:28 pm #

Thanks for the post. I don't understand what's the meaning of “F-score” in the x-axis of the feature importance plot. And what is the number next to each of the bar?

REPLY

Jason Brownlee January 16, 2017 at 10:36 am #

Hi Johnn,

You can learn more about the F1 score here:
https://en.wikipedia.org/wiki/F1_score

The number is a scaled importance, it really only has meaning relative to other features.

REPLY

Gonçalo Abreu June 5, 2017 at 10:32 pm #

Hey Jason,

Are you sure the F score on the graph is realted to the tradicional F1-score?

I found this github page where the owner presents many ways to extract feature importance meaning from xgb. His explanation abou the F measure seems to have no relation to F1

https://github.com/Far9n/xgbfi

Jason Brownlee June 6, 2017 at 9:36 am #

REPLY

Importance scores are different from F scores. The above tutorial focuses on feature importance scores.

Domi April 20, 2020 at 9:29 pm #

Hi Johnn,

Gonçalo has right , not the F1 score was the question. F1 score is totally different from the F score in the feature importance plot.

F score in the feature importance context simply means the number of times a feature is used to split the data across all trees. at least, if you are using the built-in feature of Xgboost.

Resource: https://github.com/dmlc/xgboost/blob/b4f952b/python-package/xgboost/core.py#L1639-L1661

I hope it helped to clarify things.

Cheers,

D.

Jason Brownlee April 21, 2020 at 5:54 am #

Thanks for sharing!

tuttoaposto June 23, 2020 at 4:34 pm #

REPLY

plot_importance() by default plots feature importance based on importance_type = 'weight', which is the number of times a feature appears in a tree.

It is confusing when compared to clf.feature_importance_, which by default is based on normalized gain values.

You can check the correspondence between the plot and the feature_importance_ values using this code:

```
# How to get back feature_importances_ (gain based) from plot_importance fscore
# Calculate two types of feature importance:
# Weight = number of times a feature appears in tree
# Gain = average gain of splits which use the feature = average all the gain values of the feature if it
```

appears multiple times
# Normalized gain = Proportion of average gain out of total average gain
k = clf.get_booster().trees_to_dataframe()
group = k[k['Feature']!='Leaf'].groupby('Feature').agg(fscore = ('Gain', 'count'),
feature_importance_gain = ('Gain', 'mean'))
# Feature importance same as plot_importance(importance_type = 'weight'), default value
group['fscore'].sort_values(ascending=False)
Feature importance same as clf.feature_importance_ default = 'gain'
group['feature_importance_gain_norm'] =
group['feature_importance_gain']/group['feature_importance_gain'].sum()
group['feature_importance_gain_norm'].sort_values(by='feature_importance_gain_norm',
ascending=False)

Feature importance same as plot_importance(importance_type = 'gain')
group[['feature_importance_gain']].sort_values(by='feature_importance_gain', ascending=False)

Note:

1. Features with zero feature_importance_ don't show in trees_to_dataframe(). You can check what they are with:
X_train.columns[[x not in k['Feature'].unique() for x in X_train.columns]]

2. The feature importance ranks for 'weight' and 'gain' types can be quite different. Be careful when choosing features based on the plot. I would choose gain over weight because gain reflects the feature's power of grouping similar instances into a more homogeneous child node at the split.

Jason Brownlee June 24, 2020 at 6:23 am # REPLY

Fantastic comment!

Soyoung Kim April 20, 2017 at 2:39 am # REPLY

Hi Jason,

Your postings are always amazing for me to learn ML techniques!
Especially this XGBoost post really helped me work on my ongoing interview project.
The task is not for the Kaggle competition but for my technical interview! 🙂

I used your code to generate a feature importance ranking and some of the explanations you used to describe techniques.
You can find it here: https://www.kaggle.com/soyoungkim/two-sigma-connect-rental-listing-inquiries/rent-interest-classifier

I also put your link in the reference section.

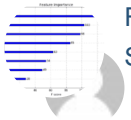
Please let me know if it is not appropriate for me to use your code.

Never miss a tutorial:

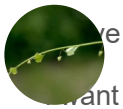
I do

As long as you cite the source, I am happy.

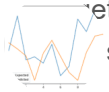
Picked for you:



Feature Importance and Feature Selection With XGBoost in Python

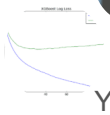


How to Develop Your First XGBoost Model in Python



How to Use XGBoost for Time Series Forecasting

Data Preparation for Gradient Boosting with XGBoost in Python



Avoid Overfitting By Early Stopping With XGBoost In Python

Loving the Tutorials?

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

---

ztlara April 27, 2017 at 3:42 pm # REPLY

Hi Jason,

I have some questions about feature importance.

I want to use the features that selected by XGBoost in other classification models, and I got confused on how to get the right scores of features, I mean that is it necessary to adjust parameters to get the best model and obtain the corresponding scores of features? In other words, how can I get the scores of features in the model?

Thanks a lot.

Jason Brownlee April 28, 2017 at 7:36 am # REPLY

The scores are relative.

You can use them as a filter and select all features with a score above x, e.g. 0.5.

---

max January 14, 2018 at 12:00 pm # REPLY

Hi Jason, I know that choosing a threshold (like 0.5) is always arbitray …but is there a thanks a lot.

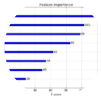
Jason Brownlee January 15, 2018 at 6:55 am # REPLY

Yes, start with 0.5, tune if needed.

Joe Butkovic August 26, 2019 at 10:58 pm #

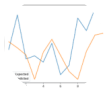
Hi Jason,

Is there a score which should be discounted? For example, my highest score is 0.27, then 0.15, 0.13… Should I discount the model all together? Thanks!



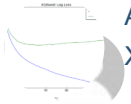
Feature Importance and Feature Selection With XGBoost in Python

How to Develop Your First XGBoost Model in Python



How to Use XGBoost for Time Series Forecasting

Data Preparation for Gradient Boosting with XGBoost in Python



Avoid Overfitting By Early Stopping With XGBoost in Python

Jason Brownlee August 27, 2019 at 6:46 am #

Sc are relative. Test different cut-off values on your specific dataset.

Shubham Jaiswal August 31, 2019 at 9:28 am #

One good way to not worry about thresholds is to use something like – CalibratedClassifierCV(clf, cv='prefit', method='sigmoid').

It kind of calibrated your classifier to .5 without screwing you base classifier output.

Jason Brownlee September 1, 2019 at 5:34 am #

Nice, thanks for sharing this tip!

I also have a little more on the topic here:
https://machinelearningmastery.com/calibrated-classification-model-in-scikit-learn/

Omogbehin . May 16, 2017 at 10:23 am #

REPLY

Hello sir,

For the XGBoost feature selection, How do i change the Y axis to the names of my attributes. Kind regards sir.

y 17, 2017 at 8:23 am #

REPLY

Great question, I'm not sure off-hand. You may need to use the xgboost API directly.

Franco Arda October 12, 2018 at 8:27 pm #

REPLY

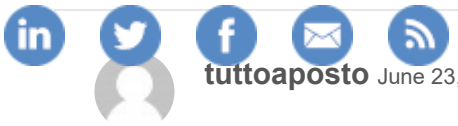
@Omogbehin, to get the Y labels automatically, you need to switch from arrays to Pandas dataframe. By doing so, you get automatically labeled Y and X.

```
column_names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = pd.read_csv("diabetes.csv", names = column_names)
X = data.iloc[:,0:8]
Y = data.iloc[:,8]
model = XGBClassifier()
model.fit(X, Y)

from xgboost import plot_importance
plot_importance(model)
```

plt.show()

Never miss a tutorial:



tuttoaposto June 23, 2020 at 3:56 pm #

REPLY

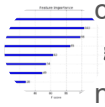
Picked for you: You can plot feature_importance directly as in:

clf = xgb.XGBClassifier(
learning_rate =0.1,
n_estimators=1000,
max_depth=5,
min_child_weight=1,
gamma=0,
subsample=0.8,
colsample_bytree=0.8,
objective= 'multi:softprob',
nthread=4,
# scale_pos_weight=1,
num_class=6,
seed=0,
verbosity=0).fit(X_train, y_train)

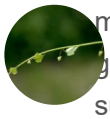
%matplotlib notebook
fig, ax = plt.subplots(figsize=(10,6))
xgb.plot_importance(clf, height = 0.4, grid = False, ax=ax, importance_type='weight')
fig.subplots_adjust(left = 0.35);

2. Or you can also output a list of feature importance based on normalized gain values, i.e. gain/sum of gain:

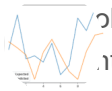
pd.Series(clf.feature_importances_, index=X_train.columns,
name='Feature Importance').sort_values(ascending=False)



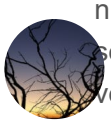
Feature Importance and Feature Selection With XGBoost in Python



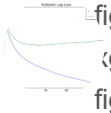
How to Develop Your First XGBoost Model in Python



How to Use XGBoost for Time Series Forecasting



Data Preparation for Gradient Boosting with XGBoost in Python



Avoid Overfitting By Early Stopping With XGBoost In Python

Loving the Tutorials?

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

>> SEE WHAT'S INSIDE

Jason Brownlee June 24, 2020 at 6:22 am #

REPLY

Great tips!

Simone June 21, 2017 at 11:14 pm #

REPLY

Hi Jason,

Is it possible using "feature_importances_" in XGBRegressor() ?

Jason Brownlee June 22, 2017 at 6:06 am #

REPLY

I'm not sure off the cuff, sorry.

REPLY

Simone June 22, 2017 at 7:06 am #

Ok, I will try another method for features selection.

Thanks

REPLY

Richard July 22, 2017 at 8:04 pm #

Hello Jason, I use the XGBRegressor and want to do some feature selection. However, although the 'plot_importance(model)' command works, when I want to retreive the values using el.feature_importances_, it says 'AttributeError: 'XGBRegressor' object has no attribute ure_importances_'. Any hints how to retreive the feature importances for regression?

REPLY

Jason Brownlee July 23, 2017 at 6:23 am #

Sorry to hear that Richard. I'm not sure of the cause.

REPLY

Long.Ye August 23, 2017 at 10:41 am #

Hi Jason,

Do you know some methods to quality variable importance in RNN or LSTM? Could the XGBoost method be used in regression problems of RNN or LSTM? Thanks a lot.

REPLY

Jason Brownlee August 23, 2017 at 4:23 pm #

Perhaps, I have not tried.

REPLY

Edward August 25, 2017 at 3:57 pm #

Can you explain how the decision trees feature importance also works?

REPLY

Biswajit September 9, 2017 at 10:36 pm #

Hi Jason while trying to fir my model in Xgboost object it is showing the below error

OSError: [WinError -529697949] Windows Error 0xe06d7363

i am using 32 bit anaconda

import platform

platform.architecture()

('32bit', 'WindowsPE')

Please suggest how to get over this issue

REPLY

Never miss a tutorial:

Jason Brownlee September 11, 2017 at 12:01 pm #

Sorry, I have not seen this error.

Perhaps you can post to stackoverflow?

REPLY

kim tae in September 23, 2017 at 4:51 pm #

Hi Jason.

SelectFromModel(model, threshold=thresh, prefit=True)

I wonder what prefit = true means in this section. I checked on the sklearn site, but I do not understand.

REPLY

Jason Brownlee September 24, 2017 at 5:15 am #

It specifies not to fit the model again, that we have already fit it prior.

REPLY

at 2:14 am #

Hi, Jason

Can you get feature importance of artificial neural network?

If you can, how?

Thanks very much.

REPLY

Jason Brownlee January 19, 2018 at 6:35 am #

Perhaps, I have not seen this done.

REPLY

Zhang January 25, 2018 at 11:41 pm #

Hi, Jason. I am doing a project with Stochastic gradient boosting. My database is clinical data and I think the ranking of feature importance can feed clinicians back with clinical knowledge, i.e., machine can tell us which clinical features are most important in distinguishing phenotypes of the diseases. What I did is to predict the phenotypes of the diseases with all the variables of the database using SGB in the training set, and then test the performance of the model in testing set. If the testing is good (e.g., high accuracy and kappa), then I would like to say the ranking of the feature importance is reasonable as machine can make good prediction using this ranking information (i.e., the feature importance is the knowledge machine learns from the database) and it is correct because machine uses this knowledge to make good classification). Vice versa, if the prediction is poor I would like to say the ranking of feature importance is bad or even wrong. In this case we cannot trust the 'knowledge' feed back by the machine. In other words, it wastes time to do feature selection in this case because the feature importance is not correct (either because of the poor data quality or the machine learning algorithm is not suitable). May I ask whether my thinking above is reasonable?

My second question is that I did not do feature selection to identify a subset of features as you did in your post. I just treat the few features on the top of the ranking list as the most important clinical features and then did classical analysis like t test to confirm these features are statistically different in different phenotypes. Can I still name it as feature selection or feature extraction? I am little bit confused about these terms. Thanks and I am waiting for your reply.

REPLY

Jason Brownlee January 26, 2018 at 5:42 am #

Sorry, I'm not sure I follow. Perhaps you can distil your question into one or two lines?

Yes, you could still call this feature selection.

REPLY

Zhang January 26, 2018 at 8:09 pm #

Thanks for your reply.

As you ent boosting (SGB) is a model with built-in feature selection, which is thought to be more efficient in feature selection than wrapper methods and filter methods. But I doubt whether we can always trust the feature selected by SGB because the importance (relative influence) of the features are still provided by the model when the model has bad performance (e.g., very poor accuracy in testing). In this case, the model may be even wrong, so the selected features may be also wrong. So I would like to hear some comment from you regarding to this issue.

Thanks.

REPLY

Jason Brownlee January 27, 2018 at 5:57 am #

Perhaps compare models fit with different subsets of features to see if it is lifting skill.

Try using an ensemble of models fit on different subsets of features to see if you can lift skill further.

Sa January 29, 2018 at 3:25 pm #

REPLY

Hi, [illegible]n.

Could you please let me know if the feature selection method that you used here, is classified as filter, wrapper or embedded feature selection method?

Regards,

Jason Brownlee January 30, 2018 at 9:47 am #

REPLY

Here we are doing feature importance or feature scoring. It would be a filter.

Youcai Wang February 2, 2018 at 9:37 am #

REPLY

Hi Brownlee, if I have a dataset with 118 variables, but the target variable is in 116, and I want to use 6-115 and 117-118 variables as dependent variables, how can I modify the code X = dataset[:,0:8] Y = dataset[:,8]

to get X and Y?

I could not figure out this simple question. Please help

Thanks,

Nick March 18, 2018 at 9:47 am #

REPLY

Hi Jason,

Thanks

Did you notice that the values of the importances were very different when you used model.get_importances_ versus xgb.plot_importance(model)?

I used these two methods on a model I just trained and it looks like they are completely different. Moreover, the numpy array feature_importances do not directly correspond to the indexes that are returned from the plot_importance function.

In other words, these two methods give me qualitatively different results. Any idea why?

Jason Brownlee March 19, 2018 at 6:03 am #

REPLY

I have not noticed that. Perhaps post a ticket on the xgboost user group or on the project? Sounds like a fault?

Nick March 31, 2018 at 4:01 am # REPLY

There is a typo in my question:

It should be model.feature_importances, not model.get_importances_.

Eran M April 3, 2018 at 9:19 pm # REPLY

Better importance estimation:

…el.feature_importances_ uses the …oster.get_fscore() which uses Booster.get_score(importance_type='weight')

…ch is an estimation to 'gain' (as of how many times all trees represented a certain feature).

…nk it would be better to use Booster.get_score(importance_type='gain') to get a more precise evaluation of how important a feature is.

In general, it describes how good was it to split branches by that feature

Jason Brownlee April 4, 2018 at 6:12 am # REPLY

Thanks for sharing.

John Markson November 22, 2018 at 12:42 am # REPLY

Hi Jason

…e posts. Regarding the feature importance in Xgboost (or more generally gradient boosting trees), how do you feel about the SHAP? I am not sure if you already had any post discussing SHAP, but it is definitely interesting to people who need gradient boosting tree models for feature selections.

Thanks!

Jason Brownlee November 22, 2018 at 6:25 am # REPLY

What is SHAP?

dasgupso May 24, 2018 at 10:01 pm # REPLY

Hi Jason

I need to know the feature importance calculations by different methods like "weight", "gain", or "cover" etc. in Xgboost.

Please let me know how can we do it ? Can it be done using same way as you developed the model here (using Xgbclassifier).

Also, it's the default method which is giving variable importance as per your code (model.feature_importances_).

I need to save importances for very large set of features(around 225) using "weight", "gain", or "cover" etc. in xgboost.

Jason Brownlee May 25, 2018 at 9:25 am #

REPLY

Do not sure, I cannot present this, you might have to implement it yourself.

analysisma October 12, 2018 at 12:17 pm #

REPLY

you can use the following code:

```
model.get_booster().get_score(importance_type='type')
```

type can either 'weight', 'gain' or 'cover'.

jason January 24, 2020 at 8:22 am #

REPLY

default = 'weight'

see: https://xgboost.readthedocs.io/en/latest/python/python_api.html

Camel August 2, 2018 at 12:41 pm #

REPLY

s a problem when modeling with xgboost. Could I ask for your help? I use predict function to get a predict probability, but I get some prob which is below 0 or over 1. I'm wondering what's my problem. Could you help me? Thank you very much.

Jason Brownlee August 2, 2018 at 2:10 pm #

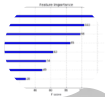
REPLY

What is the problem exactly? Low probabilities?

Rocky September 9, 2018 at 5:20 am #

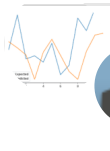
REPLY

Is it necessary to perform a gridsearch when comparing the performance of the model with different numbers of features? E.g if I wanted to see if a model with 8 features performed better than one with 4, would it be good practice to run a gridsearch with both?

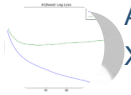


Feature Importance and Feature Selection With XGBoost in Python

How to Develop Your First XGBoost Model in Python



Data Preparation for Gradient Boosting with XGBoost in Python



Avoid Overfitting By Early Stopping With XGBoost In Python

Jason Brownlee September 9, 2018 at 6:01 am # REPLY

...epe...on h... much time and resources you have and the goals of your project.

Perhaps a comparison of the same configuration of model with different input features would be a good first step (w.g. without the grid search).

James September 26, 2018 at 12:33 pm # REPLY

How can we use let's say top 10 features to train the model? I can not find a parameter to do so ...e initiating.

Jason Brownlee September 26, 2018 at 2:24 pm # REPLY

You must use feature selection methods to select the features you want to use. There is no ...best feature selection method, just different perspectives on what might be useful.

Sinan Ozdemir September 28, 2018 at 6:56 am # REPLY

Hi Jason,

After reading your book, I was able to implement a model successfully. However, I have a few questions and I will appreciate if you provide feedback:

Q1 – In terms of feature selection, can we apply PCA (Principal Component Analysis), LDA (Linear Discriminant Analysis), or kernel PCA when we use XGBOOST to determine the most important features?

Q2 – Do you think we should apply standard scaling after one hot encoding the categorical values? Again, some people say that this is not necessary in decision tree like models, but I would like to get your opinion.

Q3 – Do we need to be concerned with the dummy variable trap when we use XGBOOST? I couldn't find a good source about how XGBOOST handles the dummy variable trap meaning if it is necessary to drop a column.

As always I really appreciate your feedback.

Thank you.

Jason Brownlee September 28, 2018 at 2:58 pm # REPLY

You can try dimensionality reduction methods, it really depends on the dataset and the configuration of the model as to whether they will be beneficial.

No real need to rescale data for xgboost. Standardizing might be useful for Gaussian variables. Test and see.

Dummy variables can cause problems, especially if they expose a grouping of levels not obvious from the data (e.g. the addition of flag variables)

Sinan Ozdemir October 5, 2018 at 6:57 am

REPLY

As always, thank you so much Jason.

For people who are interested in my experiment:

Dimensionality reduction method didn't really help much. XGBOOST feature selection method was way better in my case.

Standardizing didn't really change neither the accuracy score or the predicting results.

Keeping dummy variable increased the accuracy by about 2%, I used KFold to measure the accuracy.

Thanks

Thanks

Jason Brownlee October 5, 2018 at 2:29 pm

REPLY

Nice work!

Alvie December 13, 2018 at 3:07 am

REPLY

Thanks for your post. It is really helpful.

But I am still confused about "Importance is calculated for a single decision tree by the amount that each attribute split point improves the performance measure".

How to calculate the "amount that each attribute split point improves the performance measure"?

Jason Brownlee December 13, 2018 at 7:41 am

REPLY

This is calculated as part of constructing each individual tree. The final importance scores are an average of these scores.

Yang Song December 27, 2018 at 7:09 pm

REPLY

Hi Jason, I have encountered a problem when I try to reimplement the python trained xgboost model by c++. I built the same decision trees as the python trained(use the 'model.dump_model' function) but I got the different scores. I didn't know why and can't figure that,can you give me several tips? thanks!

Jason Brownlee December 28, 2018 at 5:54 am #

REPLY

Perhaps there was a difference in your implementation? It could be one of a million things – impossible for me to diagnose sorry.

Kamil February 12, 2019 at 7:14 am #

REPLY

Hi,

I'm running this code:

```
plot_importance(model)
pyplot.show()
```

getting an error:

ValueError: tree must be Booster, XGBModel or dict instance

can I deal with that?

Jason Brownlee February 12, 2019 at 8:09 am #

REPLY

Sorry, I have not seen that error, I have some suggestions here:
https://machinelearningmastery.com/faq/single-faq/why-does-the-code-in-the-tutorial-not-work-for-me

AAV March 16, 2019 at 9:16 am #

REPLY

Is there any way to get sign of the features to understand if the impact is positive or negative.

Jason Brownlee March 17, 2019 at 6:14 am #

REPLY

Not as far as I know, sorry.

Charles Brauer March 22, 2019 at 4:17 am #

REPLY

When I click on the link: “names in the problem description” I get a 404 error.
The “f1, f2.. ” names are not useful. I want the real column names.

REPLY

Jason Brownlee March 22, 2019 at 8:37 am #

Thanks, I have updated the link to:
https://github.com/jbrownlee/Datasets/blob/master/pima-indians-diabetes.names

REPLY

Abhinav May 7, 2019 at 7:42 pm #

Hi Jason,

Thanks again for an awesome post. Just like there are some tips which we keep in mind while feature selection using Random Forest.

Like – The categorical variable with high cardinality/ continous variable are given preference over others while splitting the nodes (as correlation is not visible in case of RF feature importance.

Do XGBoost have similar cons similar to Random Forest??

REPLY

Jason Brownlee May 8, 2019 at 6:43 am #

Yes, perhaps this post will help:
https://machinelearningmastery.com/configure-gradient-boosting-algorithm/

REPLY

Constantine May 14, 2019 at 2:10 am #

Hello!

Given feature importance is a very interesting property, I wanted to ask if this is a feature that can be found in other models, like Linear regression (along with its regularized partners), in Support Vector Regressors or Neural Networks, or if it is a concept solely defined solely for tree-based models. I ask because I am not sure whether I can consider eg Linear Regression’s coefficients as the analog for feature importance.

Thanks!

REPLY

Jason Brownlee May 14, 2019 at 7:49 am #

Yes, coefficient size in linear regression can be a sign of importance.

SVM, less so.

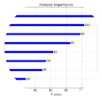
Never miss a tutorial:

Constantine May 16, 2019 at 4:39 am #

REPLY

Many thanks!

Picked for you:

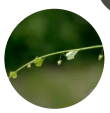


Feature Importance and Feature Selection With XGBoost in Python

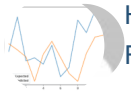
Jason Brownlee May 16, 2019 at 6:36 am #

REPLY

You're welcome.



How to Develop Your First XGBoost Model in Python



How to Use XGBoost for Time Series Forecasting

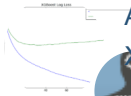
Hiro May 19, 2019 at 1:38 am #

REPLY

Thanks for all of your posts. I use your blog to study a lot.

I have a question. If you had a large number of features, do you want to use all of them? I have a dataset with over 1,000 features but not all of them are meaningful for this classification problem I am working on. Should I reduce the number of features before applying XGBoost? If so, how can I do so?

Data Preparation for Gradient Boosting with XGBoost in Python



Avoid Overfitting By Early Stopping With XGBoost In Python

Jason Brownlee May 19, 2019 at 8:05 am #

REPLY

Try modeling with all features and compare results to models fit on subsets of selected features to see if it improves performance.

Loving the Tutorials?

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

Jonathan May 21, 2019 at 4:08 am #

REPLY

>> SEE WHAT'S INSIDE

Hi Jason,

Does multicollinearity affect feature importance for boosted regression trees? If so, how would you suggest to treat this problem?

Thanks!

Jason Brownlee May 21, 2019 at 6:40 am #

REPLY

Probably not.

Try modeling with an without the colinear features and compare results.

Grzegorz Kępisty July 16, 2019 at 5:03 pm #

REPLY

Hello Jason,

Concerning default feature importance in similar method from sklearn (Random Forest) I recommend beautiful article :

https://explained.ai/rf-importance/

The authors show that the default feature importance implementation using Gini is biased.

I observed this kind of bias several times, that is overestimation of importance of artificial random variable added to data sets. For this issue – so called – permutation importance was a solution at a cost of longer computation.

However, there are other methods like “drop-col importance” (described in same source). Interestingly, while working with production data, I observed that some variables occur in head of sorted distribution or tail – depending which method of 2 above I applied.

This is somehow confusing and now I am cautious in using RF for feature selection.

Do you have some experience in this field or some best practices to share?

regards!

Jason Brownlee July 17, 2019 at 8:20 am #

REPLY

Thanks for sharing.

My best advice is to use importance as a suggestion but remain skeptical. Test many methods, many subsets, make features earn the use in the model with hard evidence.

Loving the Tutorials?

new_to_modelling July 17, 2019 at 1:08 am #

REPLY

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

My data has 5 columns, where i want to predict one of those columns so remaining 4. Out of which 2 are categorical variable and 3 are numerical variable. So, i used https://scikit-learn.org/stable/auto_examples/compose/plot_column_transformer_mixed_types.html to workout a mixed data type issues. But when i the feature_importance size does not match with the original number of columns? The size of feature_importances_ array is 918

I mean its generating extra feature or is it creating a feature value for one_hot_encoding of the categorical variable.

I checked my data has 1665 unique brand values. So, its not the same as feature_importances_ array size

Jason Brownlee July 17, 2019 at 8:28 am #

REPLY

Performing feature selection on the categorical data might be confusing as it is probably one hot encoded.

Perhaps create a subset of the data with just the numerical features and perform feature selection on that?

REPLY

new_to_modelling July 17, 2019 at 6:45 pm #

Thanks, but i found it was working once i tried dummies in place of the above mentioned column transformer approach seems like during transformation there is some loss of information when the xgboost booster picks up the feature names

REPLY

Jason Brownlee July 18, 2019 at 8:24 am #

Glad to hear it.

REPLY

Mike Sishi July 28, 2019 at 4:22 am #

Interesting article, thanks a lot!!

Ho can I reverse-engineer a Decision Tree? That is, change the target variable and consequently have ture variables adjust themselves.

Logically, I want to set a target variable value and get all possible values of feature variables that can yield the target variable value.

REPLY

Jason Brownlee July 28, 2019 at 6:49 am #

Reverse ML/predictive modeling is very hard if not entirely intractable.

You could run one model values and do this and give many “results”.

It would might not make sense for an ensemble of trees.

REPLY

Mike Sishi July 28, 2019 at 6:27 pm #

Hi Jason,

Thanks for your prompt response. I will will try to work on the solution and let you know how it goes.

Kind Regards

REPLY

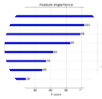
Jason Brownlee July 29, 2019 at 6:11 am #

Good luck!

Abdoul August 17, 2019 at 3:00 am #

REPLY

How to extract the n best attributs at the end?

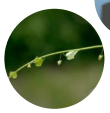


Feature Importance and Feature Selection With XGBoost in Python

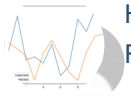
Jason Brownlee August 17, 2019 at 5:58 am #

REPLY

I give an example in the above tutorial.



How to Develop Your First XGBoost Model in Python



How to Use XGBoost for Time Series Forecasting

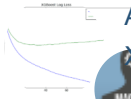
Robert Feyerharm August 28, 2019 at 11:49 pm #

REPLY

Thanks Jason, very helpful!

Data Preparation for Gradient Boosting with XGBoost in Python

Is there a way to determine if a feature has a net positive or negative correlation with the outcome variable?



Avoid Overfitting By Early Stopping With XGBoost In Python

Jason Brownlee August 29, 2019 at 6:12 am #

REPLY

Yes, you can calculate the correlation between them.

Roger September 2, 2019 at 1:16 am #

REPLY

Thresh=0.030, n=10, precision: 46.81%
Thresh=0.031, n=9, precision: 50.00%
Thresh=0.032, n=8, precision: 47.83%
Thresh=0.033, n=7, precision: 51.11%
Thresh=0.035, n=6, precision: 48.78%
Thresh=0.041, n=5, precision: 41.86%
Thresh=0.042, n=4, precision: 58.62%
Thresh=0.043, n=3, precision: 68.97%
Thresh=0.045, n=2, precision: 62.96%
Thresh=0.059, n=1, precision: 0.00%

Hi Jason, Thank you for your post, and I am so happy to read this kind of useful ML articles. I have a question: the above output is from my example. As you can see, when thresh = 0.043 and n = 3, the precision dramatically goes up. So, I want to take a closer look at that thresh and wants to find out the names and corresponding feature importances of those 3 features. How can I achieve this goal?

Jason Brownlee September 6, 2019 at 1:57 pm # REPLY

n fe e ha unique index of the column in the dataset from 0 to n. If you know the names of the columns, you can map the column index to names.

You can then do this in Python to automate it.

I hope that helps.

Ralph September 21, 2019 at 5:45 am # REPLY

Hi! I am using instead the xgb.train command instead of XGBClassifier because this is much faster. By the way you have any idea why, and if it possible to obtain the same performance with BClassifier (might be related to the number of threads)?

way, you have any idea of how to get importance feature with xgb.train?

Many thanks

Jason Brownlee September 22, 2019 at 9:27 am # REPLY

Are you sure it is faster? It should be identical in speed.

abstract September 25, 2019 at 1:56 pm # REPLY

Any reason why the Accuracy has increased from 76.38 at n=7 to 77.56 at n=6 ?

Jason Brownlee September 26, 2019 at 6:27 am # REPLY

Perhaps the change in inputs or perhaps the stochastic nature of the learning algorithm.

A fair comparison would use repeated k-fold cross validation and perhaps a significance test.

Maria September 27, 2019 at 11:50 pm # REPLY

Hello Jason,

I work on an imbalanced dataset for annomaly detection in machines. I have 590 features and 1567 observations. I tried this approach for reducing the number of features since I noticed there was multicollinearity, however, there is no important shift in the results for my precision and recall and sometimes the results get really weird. I was wondering what could that be an indication of?
Here are the results of the features selection

Thresh=0.000, n=211, f1_score: 5.71%
precision_score: 50.00%
recall_score: 3.03%
accuracy_score: 91.22%
Thresh=0.000, n=210, f1_score: 5.71%
precision_score: 50.00%
recall_score: 3.03%
accuracy_score: 91.22%
Thresh=0.000, n=209, f1_score: 5.71%
precision_score: 50.00%
recall_score: 3.03%
accuracy_score: 91.22%
Thresh=0.000, n=208, f1_score: 5.71%
precision_score: 50.00%
recall_score: 3.03%
accuracy_score: 91.22%
Thresh=0.000, n=207, f1_score: 5.71%
precision_score: 50.00%
recall_score: 3.03%
accuracy_score: 91.22%

.

Thresh=0.006, n=55, f1_score: 11.11%
precision_score: 66.67%
recall_score: 6.06%
accuracy_score: 91.49%
Thresh=0.006, n=54, f1_score: 5.88%
precision_score: 100.00%
recall_score: 3.03%
accuracy_score: 91.49%
Thresh=0.007, n=53, f1_score: 5.88%
precision_score: 100.00%
recall_score: 3.03%
accuracy_score: 91.49%
Thresh=0.007, n=52, f1_score: 5.88%
precision_score: 100.00%
recall_score: 3.03%
accuracy_score: 91.49%

.

.

.

Thresh=0.007, n=47, f1_score: 0.00%
precision_score: 0.00%
recall_score: 0.00%
accuracy_score: 91.22%

UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 due to no predicted samples.
'precision', 'predicted', average, warn_for)

Precision is ill-defined and being set to 0.0 due to no predicted samples.
'precision', 'predicted', average, warn_for)

Jason Brownlee September 28, 2019 at 6:20 am #

REPLY

Intersting, I'm not sure.

You may need to dig into the specifics of the data to what is going on. If you're using CV, then perhaps some folds don't have examples of the target class – use stratified CV.

Maria September 28, 2019 at 8:41 pm #

REPLY

You're right. I have 104 exemples of the minority class and 1463 of the other one. Would you suggest oversampling in that case?

Jason Brownlee September 29, 2019 at 6:11 am #

REPLY

Perhaps test it and see.

More ideas here:
https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

Divya Maheshwari November 1, 2019 at 6:51 pm #

REPLY

Hi

I have a doubt as to how can we know the names of the features that are selected in: model using each importance as a threshold.

Jason Brownlee November 2, 2019 at 6:41 am #

REPLY

Each column in the array of loaded data will map to the column in your raw data.

If you know column names in the raw data, you can figure out the names of columns in your loaded data, model, or visualization.

krs reddy November 22, 2019 at 12:46 am #

REPLY

Can you try plotting model interpretation using shap library for tree based algorithms??

Jason Brownlee November 22, 2019 at 6:05 am #

REPLY

nks the gestion.

Kamal December 16, 2019 at 6:39 pm #

REPLY

Variable of Importance in Xgboost for multilinear features –

I am using 60 obseravation*90features data (all continuous variables) and the response variable is also continuous. These 90 features are highly correlated and some of them might be redundant. I am using feature importance in python(xgb.feature_importances_), that sumps up 1. I run xgboost 100 times and select features based on the rank of mean variable importance in 100 runs. Let's say I choose 10 factors and then, again run xgboost with the same hyperparameters on these 10 features, surprisingly most important feature becomes least important in these 10 variables.Any feasible explanation for ?

Jason Brownlee December 17, 2019 at 6:31 am #

REPLY

Not off hand, does it matter though?

Choose a subset of features that gives the best results/most skillful model – any importance scores are a "suggestion" at best.

Sathya Bhat January 25, 2020 at 3:48 am #

REPLY

In a XGBoost model, the top features we derive shows which feature is more influential than the rest.

For example if the top feature is tenure days, how do i determine if "more tenure days" or "less tenure days" increase the rating in the output..

How do I determine if it is a positive influence or negative influence?

Jason Brownlee January 25, 2020 at 8:40 am #

REPLY

Not sure I follow. Importance is not positive or negative.

If you're in doubt: build a model with and without it and compare the performance of the model.

Sahil Basera February 11, 2020 at 7:28 pm #

REPLY

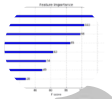
I don't understand the F -score in the feature importance plot, who can the value be 100+. Also, if this is not the traditional F-score, could you point to the definition/explanation of it? (can't find it in the

xgb documentation)

Jason Brownlee February 12, 2020 at 5:44 am #

REPLY

The scores are relative.



Feature Importance and Feature Selection With XGBoost in Python

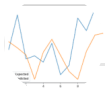
Shreya February 26, 2020 at 7:34 am #

REPLY

How to Develop Your First XGBoost Model in Python

Is it possible to plot important features on model ensembled using Voting Classifier ?
t then what could be the alternative to plot important features in an ensembled technique ?

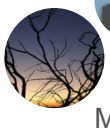
Thank You



How to Use XGBoost for Time Series Forecasting

Jason Brownlee February 26, 2020 at 8:30 am #

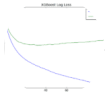
REPLY



Data Preparation for Gradient Boosting with XGBoost in Python

Not really.

Most ensembles of decision trees can give you feature importance.



Avoid Overfitting By Early Stopping With XGBoost In Python

Shreya February 27, 2020 at 12:57 am #

REPLY

Thank You !

But what about ensemble using Voting Classifier consisting of Random Forest, Decision Tree, XGBoost and Logistic Regression ?

Jason Brownlee February 27, 2020 at 5:55 am #

REPLY

Voting ensemble does not offer a way to get importance scores (as far as I know), regardless of what is being combined.

Shreya March 1, 2020 at 2:29 am #

ok.

Thanks a lot !!

Jason Brownlee March 1, 2020 at 5:25 am #

You're welcome.

Never miss a tutorial:

Picked for you:

Loving the Tutorials?

it works in xgboost 0.90, but not 1.0.2

REPLY

Jason Brownlee April 7, 2020 at 5:51 am #

Thanks, I will investigate!

REPLY

Jason Brownlee April 8, 2020 at 10:12 am #

I have added a work around.

REPLY

d8veone April 9, 2020 at 12:28 am #

Awesome! you're a true master. Thank you.

REPLY

Jason Brownlee April 9, 2020 at 8:04 am #

Thanks for your kind words.

REPLY

Lovisa April 13, 2020 at 1:19 am #

Please, remove my last post… xgboost 0.90 worked

REPLY

Jason Brownlee April 7, 2020 at 1:30 pm #

Done.

REPLY

John April 14, 2020 at 8:43 am #

Hi,
I would like to use the "Feature Selection with XGBoost Feature Importance Scores" approach with model selection in my reserach. How can I cite it in paper/thesis?
Thank you

REPLY

Jason Brownlee April 14, 2020 at 10:39 am #

This will help:
https://machinelearningmastery.com/faq/single-faq/how-do-i-reference-or-cite-a-book-or-blog-post

REPLY

Mohie April 26, 2020 at 2:22 pm #

my xgb model is taking too long for one fit and i want to try many thresholds so can i use ner simple model to know the best threshold and is yes what do you recommend ?

REPLY

Jason Brownlee April 27, 2020 at 5:28 am #

You can try, but the threshold should be calculated for the specific model.

REPLY

kim May 6, 2020 at 12:35 pm #

Hello, Sir.
ried to run print(model.feature_importances_)
but it give an array with all ‘nan’ like [nan nan nan … nan nan nan]
also, when i tried to plot the model with plot_importance(model), it return Booster.get_score() results npty

do you have any advice? thank you very much

REPLY

Jason Brownlee May 6, 2020 at 1:38 pm #

check that you fit the model?

REPLY

kim May 6, 2020 at 2:11 pm #

yes it return like this

XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1, importance_type='gain', interaction_constraints=None, learning_rate=0.300000012, max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan, monotone_constraints=None, n_estimators=100, n_jobs=0, num_parallel_tree=1, objective='binary:logistic', random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method=None, validate_parameters=False, verbosity=None)

Jason Brownlee May 7, 2020 at 6:37 am #

REPLY

cing.

I'm not sure off the cuff, you might have to try varying the training data and review the effects.

Apo May 12, 2020 at 3:56 am #

REPLY

Hello Mr. Brownlee,

I testing your idea with feature importance of XGB and thresholds in a problem that I survey these days. I'm dealing with some weird results and I wonder if you could help.

ly, run a part of code similar to yours to see different metrics results on each threshold (beginning all features to end up with 1). After that I check these metrics and note the best outcomes and the number of features resulting in these (best) metrics. Finally, I'm taking these features and use XGB algorithm with only these features but this time the results are different with results I got in the previous Any good explanation of this side effect?

Thanks for your time

Jason Brownlee May 12, 2020 at 6:51 am #

REPLY

Perhaps the difference in results is due to the stochastic nature of the learning algorithm or test harness.

Perhaps design a robust test harness and perform feature selection within the modeling pipeline.

babak June 6, 2020 at 5:33 pm #

REPLY

dear Jason
thank you for your program
I have 2 questions
1)if my target data are not categorical or binary for example so as Boston housing price has many price target so I encoding the price first before feature selection?
2) does the feature selection and correlation must have the same results?
thank first for your time

Jason Brownlee June 7, 2020 at 6:20 am #

REPLY

No, that is a regression problem:
https://machinelearningmastery.com/faq/single-faq/what-is-the-difference-between-classification-and-regression

No, each technique will give you a different idea of what features may be important. You can fit a model from each suggestion and discover what actually results in a skillful model.

REPLY

babak June 7, 2020 at 8:57 pm #

thank you for your answer

REPLY

Jason Brownlee June 8, 2020 at 6:10 am #

You're welcome.

REPLY

alireza July 1, 2020 at 4:30 am #

hi.
get Effect (percentage) of the input variables on the output variable

REPLY

Jason Brownlee July 1, 2020 at 5:56 am #

One approach would be to covert each score to a ratio of the sum of the scores.

REPLY

James Hutton July 12, 2020 at 8:43 am #

Hi Jason,

Thank rial on this – I learn a lot.

I have one question, in the Feature Selection with XGBoost Feature Importance Scores section, you used

thresholds = sort(model.feature_importances_)

and do the for loop along these threshold values to evaluate the possible models.

Is there anyway how to do similar by using the values from plot_importance() results as the thresholds?

I looked at the data type from plot_importance() return, it is a matplotlib object instead of an array like the ones from model.feature_importances_

Can you please guide me on how to implement this?

Thank you

REPLY

Jason Brownlee July 12, 2020 at 11:28 am #

Good question James, yes there must be, but I'm not sure off hand. Perhaps check the xgboost library API for the appropriate function?

REPLY

James Hutton July 12, 2020 at 8:27 pm #

Alright

One more thing, in the results of different thresholds and respective different n number of features, how to pull in which features are in each scenario of threshold or in this n number of features? Means, which features are they? Can you show perhaps?

REPLY

Jason Brownlee July 13, 2020 at 6:01 am #

Sorry, I don't follow your questions. Can you please restate or elaborate?

REPLY

Julie July 30, 2020 at 1:27 am #

Jason,

way of explaining is very simple and straiprint(classification_report(y_test, predicted_xgb))ght ward. Please keep doing this!!!

As for this subject, I've done both manual feature importance and xgboost buit-in one but got different rankings. I'm not sure why ??

REPLY

Jason Brownlee July 30, 2020 at 6:24 am #

I believe they use a different evaluation function for the plot vs automatic.

REPLY

Antonio September 29, 2020 at 1:12 am #

Amazing job Jason, Very helpful…!

If I may ask about the difference between the two ways of calculating feature importance, as I'm having contradictory results and non-matching numbers.

Exp: first way is giving output in [0,1], and the second way is giving results >1, can you explain the difference please

In addition to that, if we take feature importance as ranking and setting apart the different scale issue between the two approaches, I encountered contradictory results where the number 1 important feature in the first method isn't the number 1 in the second method.

Jason Brownlee September 29, 2020 at 5:42 am #

REPLY

nks

I believe the built-in method uses a different scoring system, you can change it to be consistent with an argument to the function.

Jacob November 17, 2020 at 5:45 am #

REPLY

Hi,

ooks like the feature importance results from the model.feature_importances_ and the built in xgboost.plot_importance are different if your sort the importance weight for model.feature_importances_.

ink you'd rather use model.get_fscore() to determine the importance as xgboost use fs score to mine and generate feature importance plots.

-Jacob

Jason Brownlee November 17, 2020 at 6:34 am #

REPLY

Thanks for sharing.

Sarah November 18, 2020 at 2:20 am #

REPLY

What is the difference between feature importance and feature selection methods?

Jason Brownlee November 18, 2020 at 6:45 am #

REPLY

Good question, I answer it here:
https://machinelearningmastery.com/faq/single-faq/what-is-the-difference-between-feature-selection-and-feature-importance

Sarah November 19, 2020 at 2:49 am #

REPLY

Great explanation, thanks. So, when we run feature selection should we expect the most important variables to be selected?

Jason Brownlee November 19, 2020 at 7:47 am #

REPLY

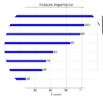
Thanks!

XGBoost performs feature selection automatically as part of fitting the model.

Saran November 19, 2020 at 10:30 am #

REPLY

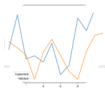
Jason, thank you so much for the clarification about the XG-Boost. In the XGBoost, I used xgb.plot_importance which plots all the features by their F score. How I can plot the selected features which are used as part of fitting the model.?

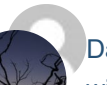


Jason Brownlee November 19, 2020 at 1:38 pm #

REPLY

The importance score itself is a reflection of the degree to which the features were used to fit the model.





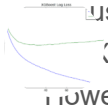
Diana November 27, 2020 at 3:38 am #

REPLY

Hi,

Thank you for your wisdom.

I am using python and the recursive feature elimination (RFE). I'm trying different types of models such as XGBClassifier, Decision Trees, or KNN.



However, the RFE gives me the following error when the model is XGBClassifier or KNN. The KNN does not provide logic to do feature selection, but the XGBClassifier does.

I'm doing something wrong or is there an explanation for this error with XGBClassifier?

Error: The classifier does not expose "coef_" or "feature_importances_" attributes

Jason Brownlee November 27, 2020 at 6:44 am #

REPLY

You're welcome.

Interesting. Perhaps check of your xgboost library is up to date?

Otherwise, perhaps xgboost cannot be used in this way – which is a shame.

Jean-Marc December 7, 2020 at 10:28 pm #

REPLY

Hello,

I am getting an empty select_X_train when using the smallest threshold (So normally I will get the same for all other thresholds). Can someone please help me find out why?

Regards!

REPLY

mhr007 December 11, 2020 at 7:26 pm #

Why are you using 'SelectFromModel' here ?

can't we just do something like this ?

```
ession_model2 = xgb.XGBRegressor(**tuned_params)
ession_model2.fit(X_imp_train,y_train,eval_set = [(X_imp_train,y_train),
(X_imp_test,y_test)],verbose=False)

_importance_dict2temp = regression_model2.get_booster().get_score(importance_type='gain')
_importance_dict2temp = sorted(gain_importance_dict2temp.items(), key=lambda x: x[1],
reverse=True)

ture selection
re_importance_len = len(gain_importance_dict2temp)

temmae = 10000.0
mpfeature_list = []
n range(1,feature_importance_len):

list_of_feature = [x for x,y in gain_importance_dict2temp[:feature_importance_len-i]]
print(list_of_feature)

X_imp_train3 = X_imp_train[list_of_feature]
X_imp_test3 = X_imp_test[list_of_feature]
regression_model = xgb.XGBRegressor(**tuned_params)
regression_model.fit(X_imp_train3,y_train,eval_set = [(X_imp_train3,y_train),
(X_imp_test3,y_test)],verbose=False)

ypred= regression_model.predict(X_imp_test3)
```

REPLY

Jason Brownlee December 12, 2020 at 6:24 am #

We are using select from model because the xgboost model has feature importance scores.

You have implemented essentially what the select from model does automatically.

REPLY

mhr007 December 14, 2020 at 8:41 pm #

Thanks, you are so great, I didn't expect an answer from you for small things like this.

Thanks.

Jason Brownlee December 15, 2020 at 6:19 am #

REPLY

Eric January 22, 2021 at 4:11 am #

REPLY

Followed exact same code but got “ValueError: X has a different shape than during fitting.” in line “select_x_train = selection.transform(x_train)” after projecting the first few lines of results of the features selection

se help, many thanks

Jason Brownlee January 22, 2021 at 7:25 am #

REPLY

Sorry to hear that, perhaps these tips will help:

https://machinelearningmastery.com/faq/single-faq/why-does-the-code-in-the-tutorial-not-work-for-me

Eric January 22, 2021 at 8:06 pm #

REPLY

thanks for the reply. I add the np.sort of the threshold and problem solved

threshold = np.sort(xgb.feature_importances_)

nlee January 23, 2021 at 7:02 am #

REPLY

Happy to hear that Eric!

manuela April 3, 2021 at 8:54 pm #

REPLY

Hi jason, I have used a standard version of Algorithm A which has features x, y, and z then I had used feature engineering to add for algorithm A new features (10 new features)

I would like to use the feature importance method to select the most important features between only the 10 features without removing any of the (x, y, z features)
can I identify first the list of features on which I would like to apply the feature importance method??

Jason Brownlee April 4, 2021 at 6:51 am #

REPLY

Sure. You can use any features you like, e.g. a combination of those selected by an algorithm and those you select.

REPLY

manuela April 4, 2021 at 3:43 pm #

thank you for your replay

REPLY

Jason Brownlee April 5, 2021 at 6:09 am #

You're welcome.

REPLY

Amr J April 29, 2021 at 5:21 pm #

Thank you for the article

I decided to read in the pima Indian data using DF and put inthe feature names so that I can see those when plottng the feature importance.

I have used the following code to add the feature names to the scores of model.feature_importances_ and sort them to put in a plot:

==========

```
from pandas import DataFrame
cols=X.columns
new_df = DataFrame (cols)
importance = model.feature_importances_*100
importance = importance.round(2)
new_df2 = DataFrame (importance)
fi=pd.c                                    s=1)
fi.columns=[ Feature , score ]
fi.set_index('Feature',inplace=True)
fi.sort_values(by='score', ascending=False, inplace=True)
```

==========

When comparing this plot to the one produced by plot_importance(model), you will notice the two do not rank the features in the same order. Any idea why?

I have tried the same thing with the famous wine data and again the two plots gave different orders to the feature importance.

One interesting thing to note is that when using catboost (as compared to xgboost) and then using SHAP to understand the impact of the features on the model, the graph is very similar to the (model.feature_importances_) method.

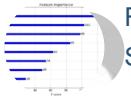
REPLY

Jason Brownlee April 30, 2021 at 6:02 am #

I believe that the plot_importance() uses a different metric for importance scores than feature_importances_.

I believe you can configure the plot function to use the same score to make the scores equivilient. I recommend checking the API.

Picked for you:



Feature Importance and Feature Selection With XGBoost in Python

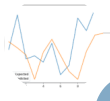
Vasiliki Voukelatou September 4, 2021 at 12:36 am #

REPLY

Hi Jason,

Thanks for the tutorial. Which is the default type for the feature_importances_ , i.e. weight, gain, etc? It is not clear in the documentation.
Thank you in advance.

How to Develop Your First XGBoost Model in Python

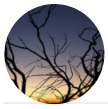


How to Use XGBoost for Time Series Forecasting

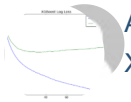
Jason Brownlee September 4, 2021 at 5:23 am #

REPLY

I don't recall, sorry. If the docs are not clear, I recommend dipping into the code.



Data Preparation for Gradient Boosting with XGBoost in Python



Avoid Overfitting By Early Stopping With XGBoost In Python

Shenyn September 28, 2021 at 10:33 pm #

REPLY

Hi Jason,

Thank you for the tutorial, it's really useful! However, I have been encountering this error (ValueError: Shape of passed values is (59372, 40), indices imply (59372, 41)) with the transform part, by any chance do you know how can I solve it? I also posted my question on Stack Overflow, but no luck 😟
https://stackoverflow.com/questions/69362344/valueerror-shape-of-passed-values-is-59372-40-indices-imply-59372-41

Adrian Tam September 29, 2021 at 11:58 pm #

REPLY

Seems an off-by-one error. Check how you preprocess your data.

Hadi.94 December 7, 2021 at 11:35 pm #

REPLY

Hey Mr. Jason .. thank you so much for your amazing article.

I'm using Feature Selection with XGBoost Feature Importance Scores with KNN based module and until now it has shown me great results.

I have one question, when I run the loop responsible of Feature Selection, I want to see the fueaturs that are involved in each iteration. Is there a simple way to do so ?

Adrian Tam December 8, 2021 at 8:07 am #

REPLY

simpler way. But the best would be to drill into the XGBoost code to add a line or two to print that out.

Swappy June 15, 2022 at 2:48 am #

REPLY

UserWarning: X has feature names, but SelectFromModel was fitted without feature names
warnings.warn(

I am getting above mentioned error while I am trying to find the feature importance scores. DF has features with names in it. Below is the code I have used. Could you please mention a solution. Thank you.

```
model = XGBClassifier()
model.fit(X_train, y_train)
# make predictions for test data and evaluate
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
# fit model using each importance as a threshold
thresholds = sort(model.feature_importances_)
for thresh in thresholds:
# select features using threshold
selection = SelectFromModel(model, threshold=thresh, prefit=True)
select_X_train = selection.transform(X_train)
# train model
selection_model = XGBClassifier()
selection_model.fit(select_X_train, y_train)
# eval
select_X_test = selection.transform(X_test)
predictions = selection_model.predict(select_X_test)
accuracy = accuracy_score(y_test, predictions)
print("Thresh=%.3f, n=%d, Accuracy: %.2f%%" % (thresh, select_X_train.shape[1], accuracy*100.0))
```


James Carmichael June 15, 2022 at 7:19 am #

REPLY

Hi Swappy…It looks like you are just using a code sample and not a full program listing.

Romy Opeña July 31, 2022 at 7:00 pm #

REPLY

I am currently applying the XGBoost Classifier on the Kaggle mushroom classification data, replicating your codes in this article. When using XGBClassifier, the number of important features could

be reduced from original 22 variables down to 6-8 with still a high accuracy scores.

I then tried to use the XGBRFClassifier on the same data and this further cut down another variable from the best feature set. However, it seems to have met a 'bump' somewhere where the accuracy went down from 100 to lower values for the next 2 reductions and then it went back up to 100 from which it resumed the downward trend.

It seems unusual but Is this normal or something is wrong with the module?

Thanks for any feedback.

Romy

REPLY

James Carmichael August 1, 2022 at 8:44 am #

Hi Romy…The following may be of interest to you:

https://machinelearningmastery.com/why-does-the-loss-accuracy-fluctuate-during-the-training/

REPLY

Romy Openia August 1, 2022 at 10:38 am #

Thanks, I will check on it. Meanwhile, I have decided to stick with XGBClassifier because I am getting some weird results when I apply XGBRFClassifier.

REPLY

Joe Salter October 4, 2022 at 10:40 pm #

Hi Jason

Thanks for your great content!

Imagin [illegible] one target (y).

After building the model, I want to see what happens if I only change one of these predictors and keep the rest constant. In other words, I want to see only the effect of that specific predictor on the target.

Is there a specific way to do that?

I was thinking about making a mock dataset with all other predictors kept the same and just changing the one that I am interested in. Then predict y and plot changes in that specific predictor and changes in y. Does that make sense? Because when I do it, then the predicted values of the mock data are the same…

REPLY

James Carmichael October 5, 2022 at 7:32 am #

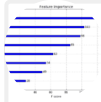
Hi Joe…You are very welcome! In general, your suggestion is a valid one for small feature sets. The following may be of interest:

https://towardsdatascience.com/the-art-of-finding-the-best-features-for-machine-learning-a9074e2ca60d

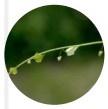
Never miss a tutorial:

Leave a Reply

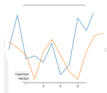
Picked for you:



Feature Importance and Feature Selection With XGBoost in Python

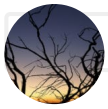


How to Develop Your First XGBoost Model in Python



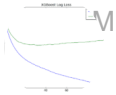
How to Use XGBoost for Time Series Forecasting

Name (required)



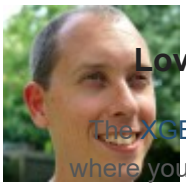
Data Preparation for Gradient Boosting with XGBoost in Python

Email (will not be published) (required)



Avoid Overfitting By Early Stopping With XGBoost In Python

MIT COMMENT



Loving the Tutorials?

Welcome!

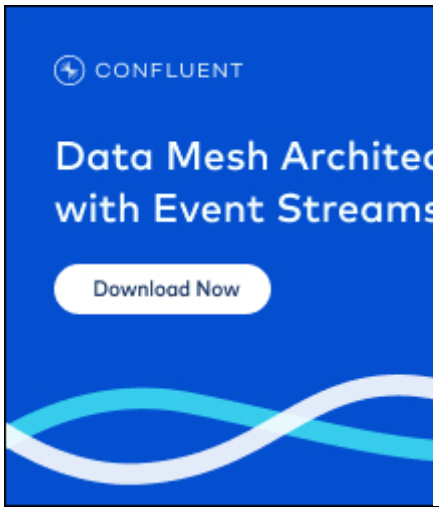
I'm *Jason Brownlee* PhD and I **help developers** get results with **machine learning**.

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

Read more

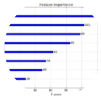
>> SEE WHAT'S INSIDE

AD



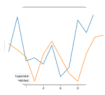
Never miss a tutorial:

Picked for you:



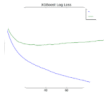
Feature Importance and Feature Selection With XGBoost in Python

How to Develop Your First XGBoost Model in Python



How to Use XGBoost for Time Series Forecasting

Data Preparation for Gradient Boosting with XGBoost in Python



Avoid Overfitting By Early Stopping With XGBoost In Python

Loving the Tutorials?

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

LinkedIn | Twitter >> SEE WHAT'S INSIDE Facebook | RSS

Privacy | Disclaimer | Terms | Contact | Sitemap | Search