How to Save and Reuse Data Preparation Objects in Scikit-Learn

by **Jason Brownlee** on November 20, 2019 in **Data Preparation**

Last Updated on June 30, 2020

It is critical that any data preparation performed on a training dataset is also performed on a new dataset in the future.

This may include a test dataset when evaluating a model or new data from the domain when using a model to make predictions.

Typically, the model fit on the training dataset is saved for later use. The correct solution to preparing new data for the model in the future is to also save any data preparation objects, like data scaling methods, to file along with the model.

In this tutorial, you will discover how to save a model and data preparation object to file for later use.

After completing this tutorial, you will know:

- The challenge of correctly preparing test data and new data for a machine learning model.
- The solution of saving the model and data preparation objects to file for later use.
- How to save and later load and use a machine learning model and data preparation model on new data.

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

Let's get started.

- **Update Jan/2020**: Updated for changes in scikit-learn v0.22 API.
- **Update May/2020**: Improved code examples and pr

How to Save and Load Models and Data
Photo by Dennis Jarvis

Tutorial Overview

This tutorial is divided into three parts; they are:

1. Challenging of Preparing New Data for a Model
2. Save Data Preparation Objects
3. How to Save and Later Use a Data Preparation Object

Challenging of Preparing New Data for a Model

Each input variable in a dataset may have different units.

For example, one variable may be in inches, another in miles, another in days, and so on.

As such, it is often important to scale data prior to fitting a model.

This is particularly important for models that use a weighted sum of the input or distance measures like logistic regression, neural networks, and k-nearest neighbors. This is because variables with larger values or ranges may dominate or wash out the effects of variables with smaller values or ranges.

Scaling techniques, such as normalization or standardization, have the effect of transforming the distribution of each input variable to be the same, such as the same minimum and maximum in the case of normalization or the same mean and standard deviation in the case of standardization.

A scaling technique must be fit, which just means it needs to calculate coefficients from data, such as the observed min and max, or the observed mean and standard deviation. These values can also be set by domain experts.

The best practice when using scaling techniques for evaluating models is to fit them on the training dataset, then apply them to the training and test datasets.

Or, when working with a final model, to fit the scaling method on the training dataset and apply the transform to the training dataset and any new dataset in the future.

It is critical that any data preparation or transformation applied to the training dataset is also applied to the test or other dataset in the future.

This is straightforward when all of the data and the model are in memory.

This is challenging when a model is saved and used la

What is the best practice to scale data when saving a

Save Data Preparation Objects

The solution is to save the data preparation object to file along with the model.

For example, it is common to use the pickle framework (built-in to Python) for saving machine learning models for later use, such as saving a final model.

This same framework can be used to save the object that was used for data preparation.

Later, the model and the data preparation object can be loaded and used.

It is convenient to save the entire objects to file, such as the model object and the data preparation object. Nevertheless, experts may prefer to save just the model parameters to file, then load them later and set them into a new model object. This approach can also be used with the coefficients used for scaling the data, such as the min and max values for each variable, or the mean and standard deviation for each variable.

The choice of which approach is appropriate for your project is up to you, but I recommend saving the model and data preparation object (or objects) to file directly for later use.

To make the idea of saving the object and data transform object to file concrete, let's look at a worked example.

How to Save and Later Use a Data Preparation Object

In this section, we will demonstrate preparing a dataset, fitting a model on the dataset, saving the model and data transform object to file, and later loading the model and transform and using them on new data.

1. Define a Dataset

First, we need a dataset.

We will use a test dataset from the scikit-learn dataset ... wo input variables created randomly via the make_blobs() ...

The example below creates a test dataset with 100 ex ... (0 and 1). The dataset is then split into training and test s ... are then reported.

Importantly, the *random_state* is set when creating the ... ame dataset is created and the same split of data is performed each time that the code is run.


```
# example of creating a test dataset and splitting it into train and test sets
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
# prepare dataset
X, y = make_blobs(n_samples=100, centers=2, n_features=2, random_state=1)
# 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=1)
# summarize the scale of each input variable
for i in range(X_test.shape[1]):
    print('>%d, train: min=%.3f, max=%.3f, test: min=%.3f, max=%.3f' %
        (i, X_train[:, i].min(), X_train[:, i].max(),
            X_test[:, i].min(), X_test[:, i].max()))
```

Running the example reports the min and max values for each variable in both the train and test datasets.

We can see that each variable has a different scale, and that the scales differ between the train and test datasets. This is a realistic scenario that we may encounter with a real dataset.

```
>0, train: min=-11.856, max=0.526, test: min=-11.270, max=0.085
>1, train: min=-6.388, max=6.507, test: min=-5.581, max=5.926
```

2. Scale the Dataset

Next, we can scale the dataset.

We will use the MinMaxScaler to scale each input variable to the range [0, 1]. The best practice use of this scaler is to fit it on the training dataset and then apply the transform to the training dataset, and other datasets: in this case, the test dataset.

The complete example of scaling the data and summarizing the effects is listed below.

```
# example of scaling the dataset
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
# prepare dataset
X, y = make_blobs(n_samples=100, centers=2, n_features=2, random_state=1)
# 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=1)
# define scaler
scaler = MinMaxScaler()
# fit scaler on the training dataset
scaler.fit(X_train)
# transform both datasets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
# summarize the scale of each input variable
for i in range(X_test.shape[1]):
    print('>%d, train: min=%.3f, max=%.3f, te
        (i, X_train_scaled[:, i].min(), X_trai
            X_test_scaled[:, i].min(), X_test_
```


Running the example prints the effect of the scaled da ... able in the train and test datasets.

We can see that all variables in both datasets now have values in the desired range of 0 to 1.

```
>0, train: min=0.000, max=1.000, test: min=0.047, max=0.964
>1, train: min=0.000, max=1.000, test: min=0.063, max=0.955
```

3. Save Model and Data Scaler

Next, we can fit a model on the training dataset and save both the model and the scaler object to file.

We will use a LogisticRegression model because the problem is a simple binary classification task.

The training dataset is scaled as before, and in this case, we will assume the test dataset is currently not available. Once scaled, the dataset is used to fit a logistic regression model.

We will use the pickle framework to save the *LogisticRegression* model to one file, and the *MinMaxScaler* to another file.

The complete example is listed below.

```
# example of fitting a model on the scaled dataset
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from pickle import dump
# prepare dataset
```

```
X, y = make_blobs(n_samples=100, centers=2, n_features=2, random_state=1)
# split data into train and test sets
X_train, _, y_train, _ = train_test_split(X, y, test_size=0.33, random_state=1)
# define scaler
scaler = MinMaxScaler()
# fit scaler on the training dataset
scaler.fit(X_train)
# transform the training dataset
X_train_scaled = scaler.transform(X_train)
# define model
model = LogisticRegression(solver='lbfgs')
model.fit(X_train_scaled, y_train)
# save the model
dump(model, open('model.pkl', 'wb'))
# save the scaler
dump(scaler, open('scaler.pkl', 'wb'))
```

Running the example scales the data, fits the model, a e.

You should have two files in your current working direc

- *model.pkl*
- *scaler.pkl*

4. Load Model and Data Scaler

Finally, we can load the model and the scaler object a

In this case, we will assume that the training dataset is dataset is available.

We will load the model and the scaler, then use the scaler to prepare the new data and use the model to make predictions. Because it is a test dataset, we have the expected target values, so we will compare the predictions to the expected target values and calculate the accuracy of the model.

The complete example is listed below.

```
# load model and scaler and make predictions on new data
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from pickle import load
# prepare dataset
X, y = make_blobs(n_samples=100, centers=2, n_features=2, random_state=1)
# split data into train and test sets
_, X_test, _, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
# load the model
model = load(open('model.pkl', 'rb'))
# load the scaler
scaler = load(open('scaler.pkl', 'rb'))
# check scale of the test set before scaling
print('Raw test set range')
for i in range(X_test.shape[1]):
    print('>%d, min=%.3f, max=%.3f' % (i, X_test[:, i].min(), X_test[:, i].max()))
# transform the test dataset
X_test_scaled = scaler.transform(X_test)
print('Scaled test set range')
for i in range(X_test_scaled.shape[1]):
```

```
    print('>%d, min=%.3f, max=%.3f' % (i, X_test_scaled[:, i].min(), X_test_scaled[:, i].max()))
# make predictions on the test set
yhat = model.predict(X_test_scaled)
# evaluate accuracy
acc = accuracy_score(y_test, yhat)
print('Test Accuracy:', acc)
```

Running the example loads the model and scaler, then uses the scaler to prepare the test dataset correctly for the model, meeting the expectations of the model when it was trained.

To confirm the scaler is having the desired effect, we report the min and max value for each input feature both before and after applying the scaling. The model then makes a prediction for the examples in the test set and the classification accuracy is calculated.

In this case, as expected, the data set correctly norma n the test set because the test problem is trivial.

```
Raw test set range
>0, min=-11.270, max=0.085
>1, min=-5.581, max=5.926

Scaled test set range
>0, min=0.047, max=0.964
>1, min=0.063, max=0.955

Test Accuracy: 1.0
```


This provides a template that you can use to save bot on your own projects.

Further Reading

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

Posts

- Save and Load Machine Learning Models in Python with scikit-learn
- How to Train a Final Machine Learning Model

APIs

- sklearn.datasets.make_blobs API.
- sklearn.model_selection.train_test_split API.
- sklearn.preprocessing.MinMaxScaler API.
- sklearn.metrics.accuracy_score API.
- sklearn.linear_model.LogisticRegression API.
- pickle API.

Summary

In this tutorial, you discovered how to save a model and data preparation object to file for later use.

Specifically, you learned:

- The challenge of correctly preparing test data and new data for a machine learning model.
- The solution of saving the model and data preparation objects to file for later use.
- How to save and later load and use a machine learning model and data preparation model on new data.

Do you have any questions?
Ask your questions in the comments below and I will do my best to answer.

About Jason Brownlee

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 →

‹ What Does Stochastic Mean in Machine Learning?   3 Ways to Encode Categorical Variables for Deep Learning ›

27 Responses to *How to Save and Reuse Data Preparation Objects in Scikit-Learn*

Venkat November 21, 2019 at 3:59 pm #

REPLY

Thank You for providing valuable information about Data Preparation and explanation about data set.

Jason Brownlee November 22, 2019 at 5:58 a

You're welcome.

Elie Kawerk November 21, 2019 at 8:42 pm #

Hi Jason,

Wouldn't it be more suitable to wrap scaler and model i

Best,
Elie

Jason Brownlee November 22, 2019 at 6:02 am #

REPLY

Yes, agreed!

In this case, I was trying to get across the idea of saving the scaler objects. A pipeline would be a simpler implementation.

Alberto Morales November 26, 2019 at 6:40 pm #

REPLY

Hi, guys.

It's an interesting post, but… How about running this processes into production?

load(open.. load? dump(model, open…? to "local" filesystems? It's not enough. Do you agree?

A lot or organizations fail during the last step (going onto production), it's necessary to apply software engineering techniques to success.

Just pickling a bunch of objects doesn't help you.

Kind regards.

Jason Brownlee November 27, 2019 at 6:02 am #

REPLY

Nice point.

Yes, saving/loading is good as first step. A better approach might be to store the coefficients used for scaling in a config and load them into the app and apply them to new data as needed, either with custom data prep code or plugging the coefficients into sklearn objects.

Does that help?

Nadia Chaabouni May 8, 2020 at 1:59 a

Thank you Jason for this interesting p
Could you please give a small example or poin
Thank you in advance

Jason Brownlee May 8, 2020 at 6:3

Yes, start with the above tutorial.

Cristian Lazaro February 21, 2020 at 4:30 pm #

REPLY

Thank you what a great post much appreciated helped me quite a bit. Very Informative.

Jason Brownlee February 22, 2020 at 6:18 am #

REPLY

You're welcome.

Emily May 11, 2020 at 4:11 pm #

REPLY

Hi Jason,

I built a clustering model to segment customer accounts. It had the following steps:

– Min Max scalar
– PCA for dimensionality reduction and then

– K means model to get ~ 5 clusters

I saved the entire pipeline using pickle and now I have to refresh the model with data for latest month. When I refreshed the model, the clusters drastically changed when compared to last month. I know they are stochastic in nature but didn't expect ~40% movement of accounts.

Is it because of the choice of algo, I mean K means? What could be the best model refresh strategy given that I'll have to refresh every month?

I know for sure that the Business owner won't be happy if there's that much movement.

Jason Brownlee May 12, 2020 at 6:36 am # REPLY

Good question!

You may have to re-fit the model with all of the old
support incremental updates.

VLADIMIR KIM May 25, 2020 at 4:41 pm #

Thank you very very much! Exactly what i nee

Jason Brownlee May 26, 2020 at 6:15 am # REPLY

You're welcome.

Mukul Verma May 27, 2020 at 5:13 am # REPLY

Hi there,Thank you so much for such a great Tutorial.You saved lot of my time.I have been worrying about how to give scaled input to a model (loaded from .h5 file of a model trained earlier by me) to get the same scaled input data.
Thank you so much.

Jason Brownlee May 27, 2020 at 8:03 am # REPLY

Load the data, scale it, load the model, pass the data to the loaded model.

sandip pani July 7, 2020 at 7:00 pm #

HI Jason,
This is helpful. Thanks for this article.

Can you also explain suppose I have preprocessing steps where I use custom logic to fill missing value. Ex. If Age is missing I find Mode of age for each group and assign accordingly if age is missing for any. So prediction as user enters all-details except Age in the form , so Would I call custom preprocessing here.

Jason Brownlee July 8, 2020 at 6:29 am #

REPLY

Thank you.

Yes, you would need to save any statistics used by custom data prep

dan August 4, 2020 at 10:21 pm #

Thanks for your post, that's very useful. I have already made a cluster analysis in order to use the labe classification task. So I have trained my model and sav data, I need have a supervised data? in other words, th labels?

Thanks in advance

Jason Brownlee August 5, 2020 at 6:13 am #

REPLY

A supervised learning model can be used to make predictions on new data, input only.

This usage of the model is the goal of supervised learning – to predict labels for new data.

John October 8, 2020 at 11:59 pm #

REPLY

Thank you, this is very helpful!

There may be a problem if the new (or test) dataset includes features with min/max values lower/higher than in the training dataset. I assume the scaler could still handle it, but the transformed high values would be above 1? It could be worse in the case of categorical features which are not represented in the training set. Do you know a solution for this case?

Thank you in advance

Jason Brownlee October 9, 2020 at 6:47 am #

You might need to manually clip new values to the known range first.

For categorical, you can set an argument to ignore new labels not seen during training, e.g. map to all zeros.

John October 9, 2020 at 5:59 pm #

REPLY

Great, thank you for the advice! I didn't know that there is an argument to ignore new labels.

Jason Brownlee October 10, 2020

Yes, set"

```
... handle_unknown="ignore"
```


LMannik November 12, 2020 at 5:36 am #

Hi, thanks for this blog. It's my go-to for machi
about assigning the fit for the model in these steps from your code:

# define model
model = LogisticRegression(solver='lbfgs')
model.fit(X_train_scaled, y_train)

For example, in the last line, model.fit(X_train_scaled, y_train), I expected to see something like fitted = model.fit(X_train_scaled, y_train), and then you'd pickle "fitted". How are you saving the trained model without assigning it to a new attribute? Or are you really just pickling the untrained model from the previous line of code: model = LogisticRegression(solver='lbfgs')?

Jason Brownlee November 12, 2020 at 6:43 am #

REPLY

You're welcome.

We fit the model by a call to fit() then save it. Recall model is an object it contains the coefficients required by the model.

Hesham Abdelghany January 1, 2021 at 4:38 pm #

REPLY

Hi Jason,

Thanks for the great post.
I have a question regarding scaling the output target variable:

1) Does the scaling method for target variable needs to be the same as the input features?

2) Do I need save 2 scaling objects, one for features different from the one for target variable?

3) If I pass scaling object for target variable from training stage to prediction stage, wouldn't that be a cause for data leaking from training to production?

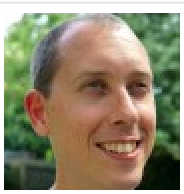
Leave a Reply

Name (required)

Email (will not be published) (required)

Website

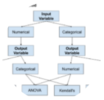
SUBMIT COMMENT



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

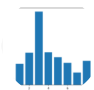
Never miss a tutorial:

Picked for you:

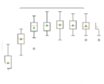


How to Choose a Feature Selection Method For Machine Learning

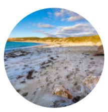
Data Preparation for Machine Learning (7-Day Mini-Course)



How to Calculate Feature Importance With Python



Recursive Feature Elimination (RFE) for Feature Se



How to Remove Outliers for Machine Learning

Loving the

The Data Prepa
where you'll find the

>> SEE WHAT'S INSIDE

Address: PO Box 206, Vermont Victoria 3133, Australia. | ACN: 626 223 336.