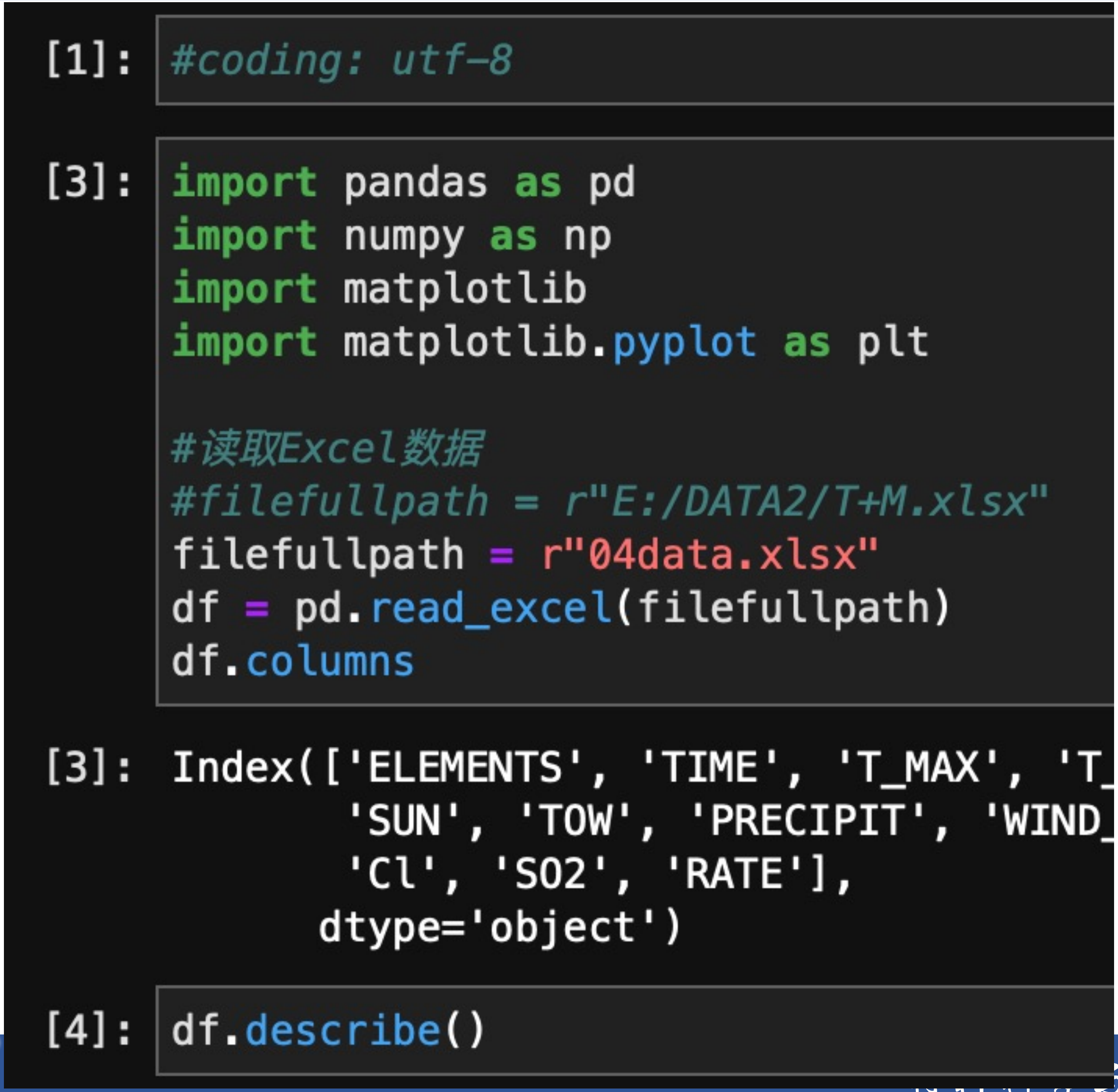
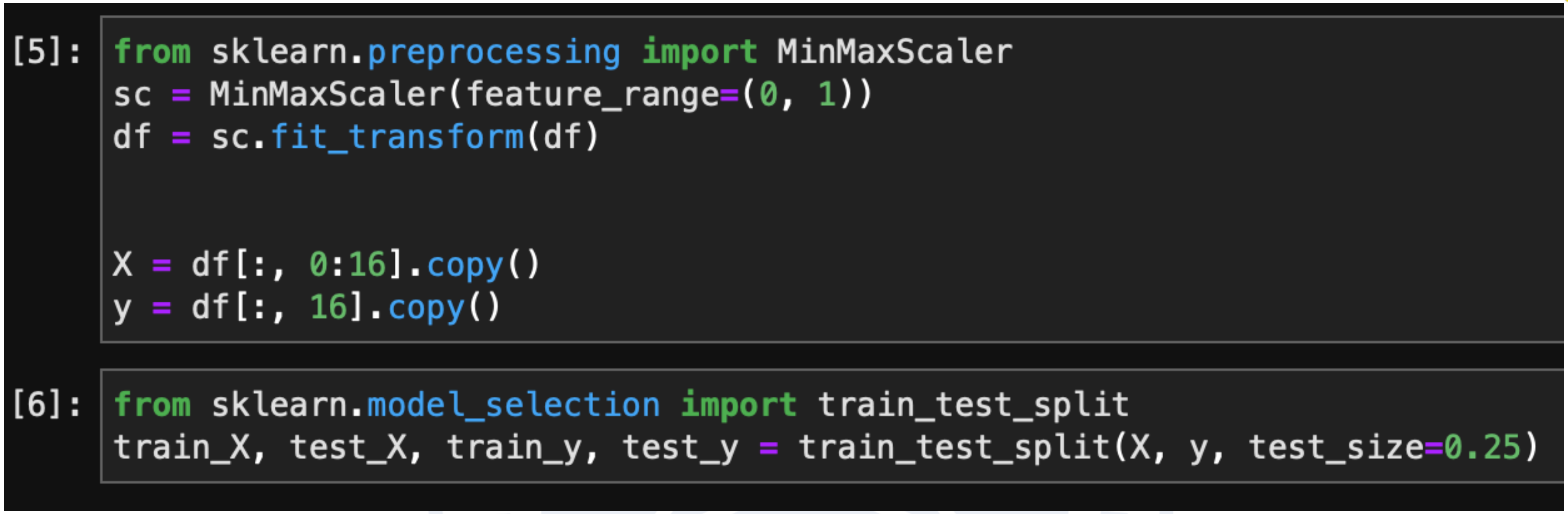
◆上机操作

3. 机器学习算法



```
[1]: #coding: utf-8

[3]: import pandas as pd
     import numpy as np
     import matplotlib
     import matplotlib.pyplot as plt

     #读取Excel数据
     #filefullpath = r"E:/DATA2/T+M.xlsx"
     filefullpath = r"04data.xlsx"
     df = pd.read_excel(filefullpath)
     df.columns

[3]: Index(['ELEMENTS', 'TIME', 'T_MAX', 'T_
            'SUN', 'TOW', 'PRECIPIT', 'WIND_
            'Cl', 'SO2', 'RATE'],
           dtype='object')

[4]: df.describe()
```

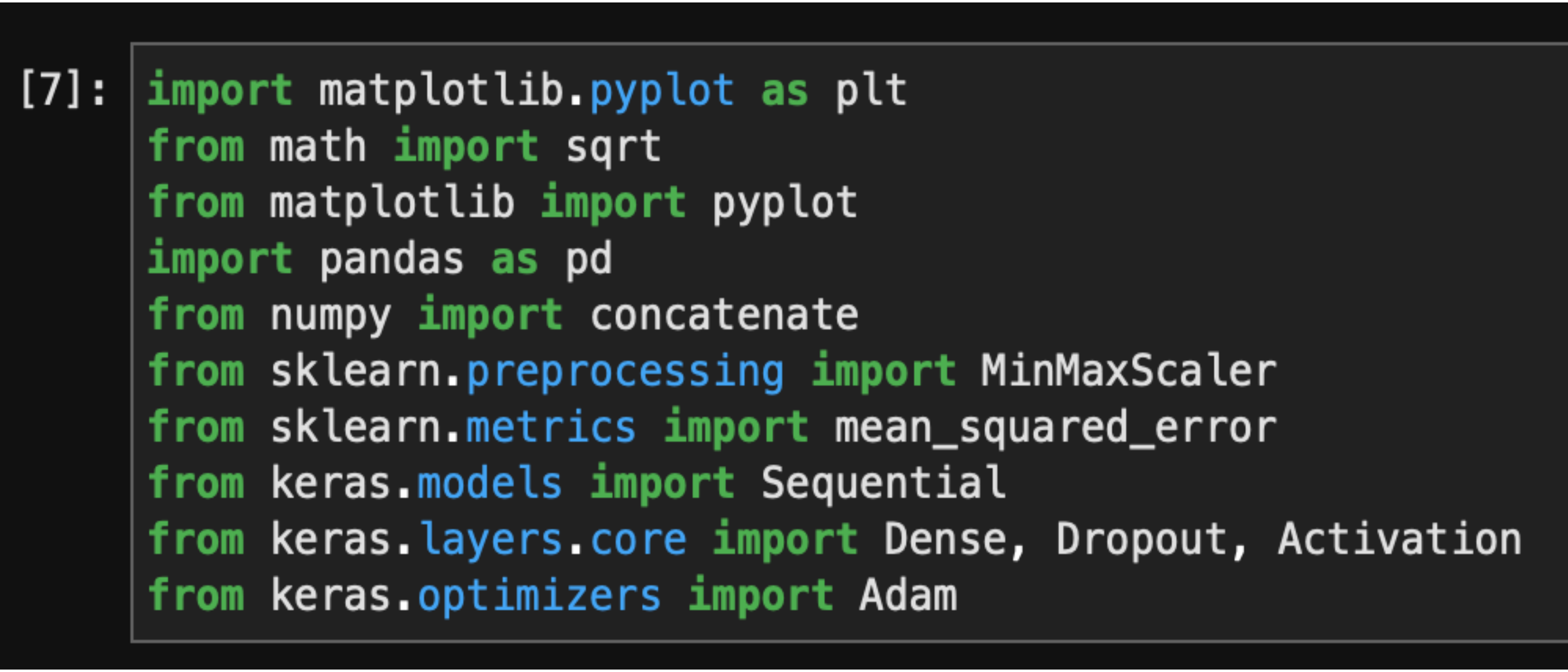
3. 机器学习算法



```
[5]: from sklearn.preprocessing import MinMaxScaler
     sc = MinMaxScaler(feature_range=(0, 1))
     df = sc.fit_transform(df)


     X = df[:, 0:16].copy()
     y = df[:, 16].copy()
```

```
[6]: from sklearn.model_selection import train_test_split
     train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.25)
```

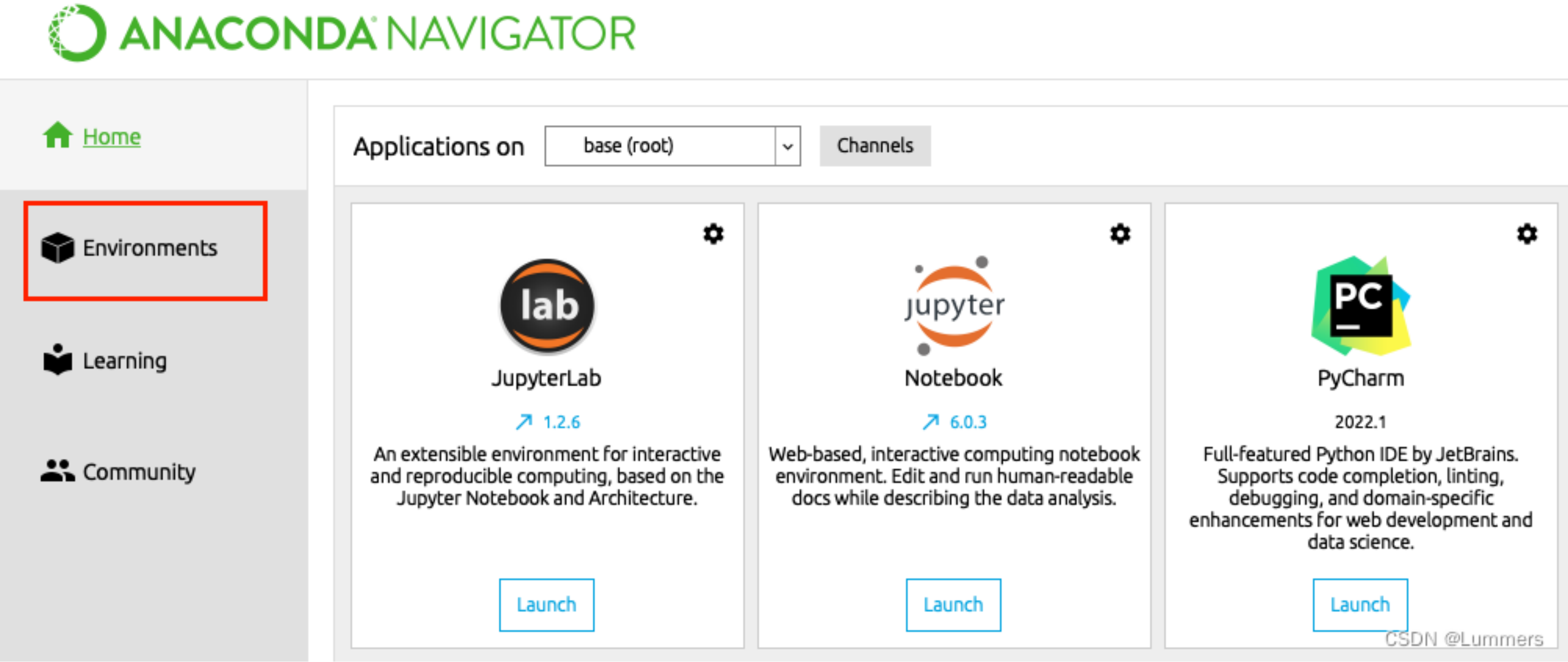
3. 机器学习算法



```
import matplotlib.pyplot as plt
from math import sqrt
from matplotlib import pyplot
import pandas as pd
from numpy import concatenate
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.optimizers import Adam
```

- ◆发生了什么?
- ◆如何解决?

3. 机器学习算法



ANACONDA NAVIGATOR

Applications on base (root) Channels

JupyterLab

↗ 1.2.6

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook

↗ 6.0.3

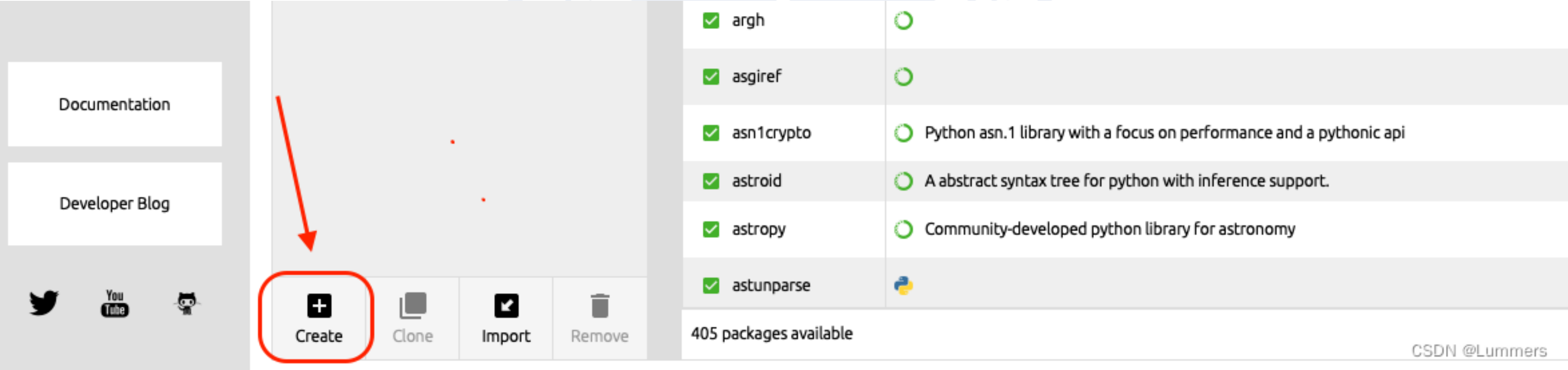
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

PyCharm

2022.1

Full-featured Python IDE by JetBrains. Supports code completion, linting, debugging, and domain-specific enhancements for web development and data science.

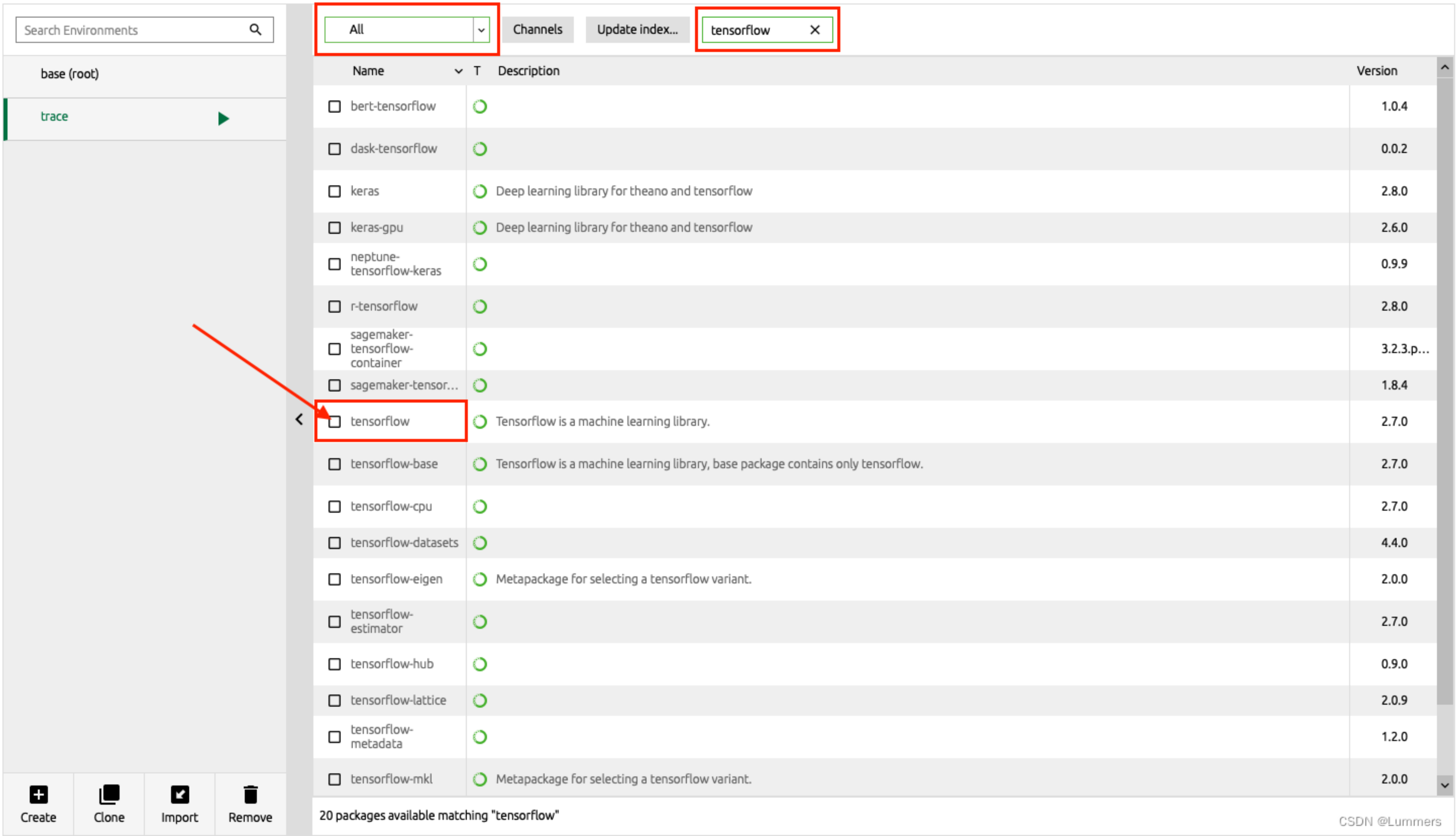
3. 机器学习算法



☑	argh	○
☑	asgiref	○
☑	asn1crypto	○ Python asn.1 library with a focus on performance and a pythonic api
☑	astroid	○ A abstract syntax tree for python with inference support.
☑	astropy	○ Community-developed python library for astronomy
☑	astunparse	

405 packages available

3. 机器学习算法

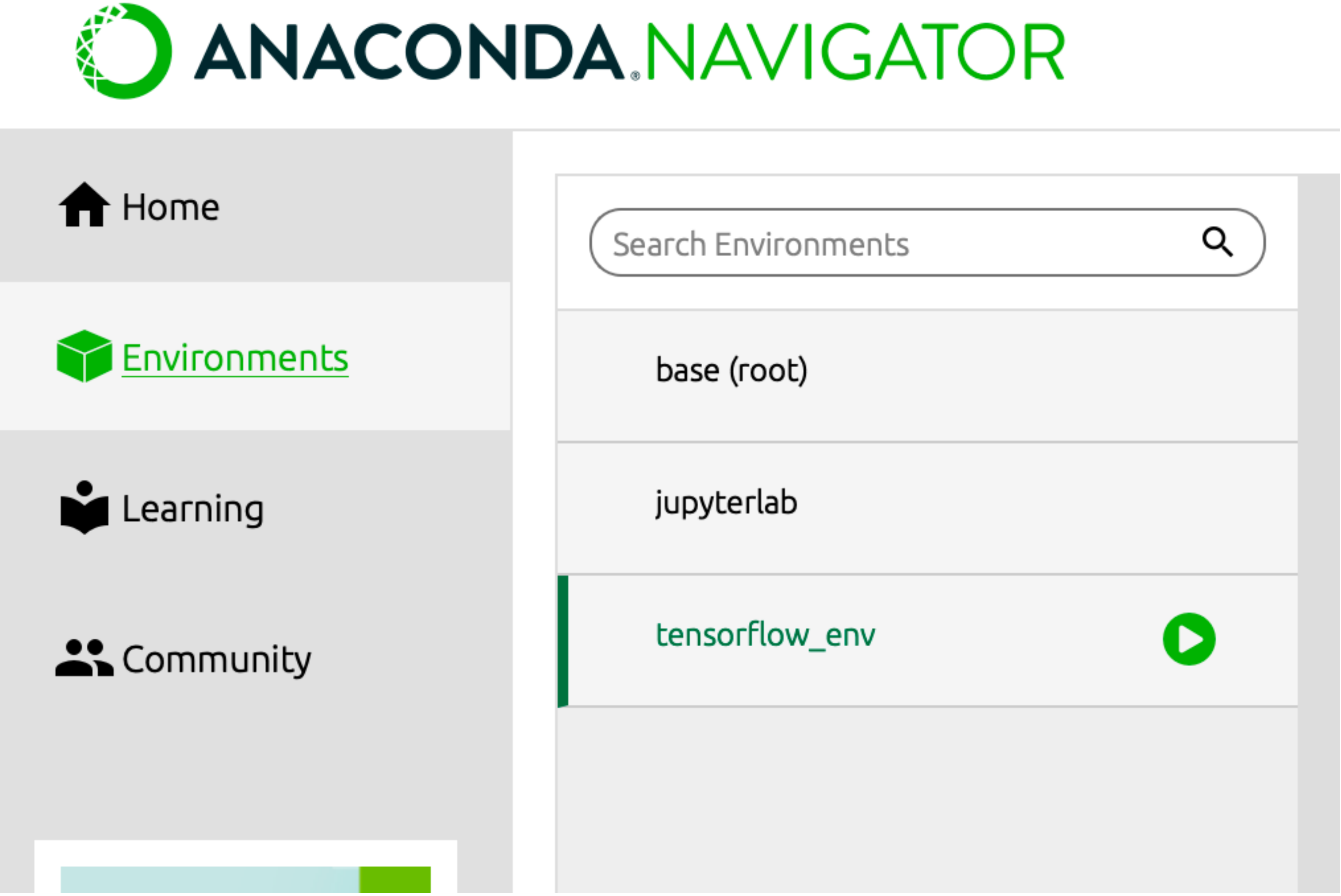


All

tensorflow ✕

	Name	T	Description	Version
□	bert-tensorflow	○		1.0.4
□	dask-tensorflow	○		0.0.2
□	keras	○	Deep learning library for theano and tensorflow	2.8.0
□	keras-gpu	○	Deep learning library for theano and tensorflow	2.6.0
□	neptune-tensorflow-keras	○		0.9.9
□	r-tensorflow	○		2.8.0
□	sagemaker-tensorflow-container	○		3.2.3.p...
□	sagemaker-tensor...	○		1.8.4
□	tensorflow	○	Tensorflow is a machine learning library.	2.7.0
□	tensorflow-base	○	Tensorflow is a machine learning library, base package contains only tensorflow.	2.7.0
□	tensorflow-cpu	○		2.7.0
□	tensorflow-datasets	○		4.4.0
□	tensorflow-eigen	○	Metapackage for selecting a tensorflow variant.	2.0.0
□	tensorflow-estimator	○		2.7.0
□	tensorflow-hub	○		0.9.0
□	tensorflow-lattice	○		2.0.9
□	tensorflow-metadata	○		1.2.0
□	tensorflow-mkl	○	Metapackage for selecting a tensorflow variant.	2.0.0

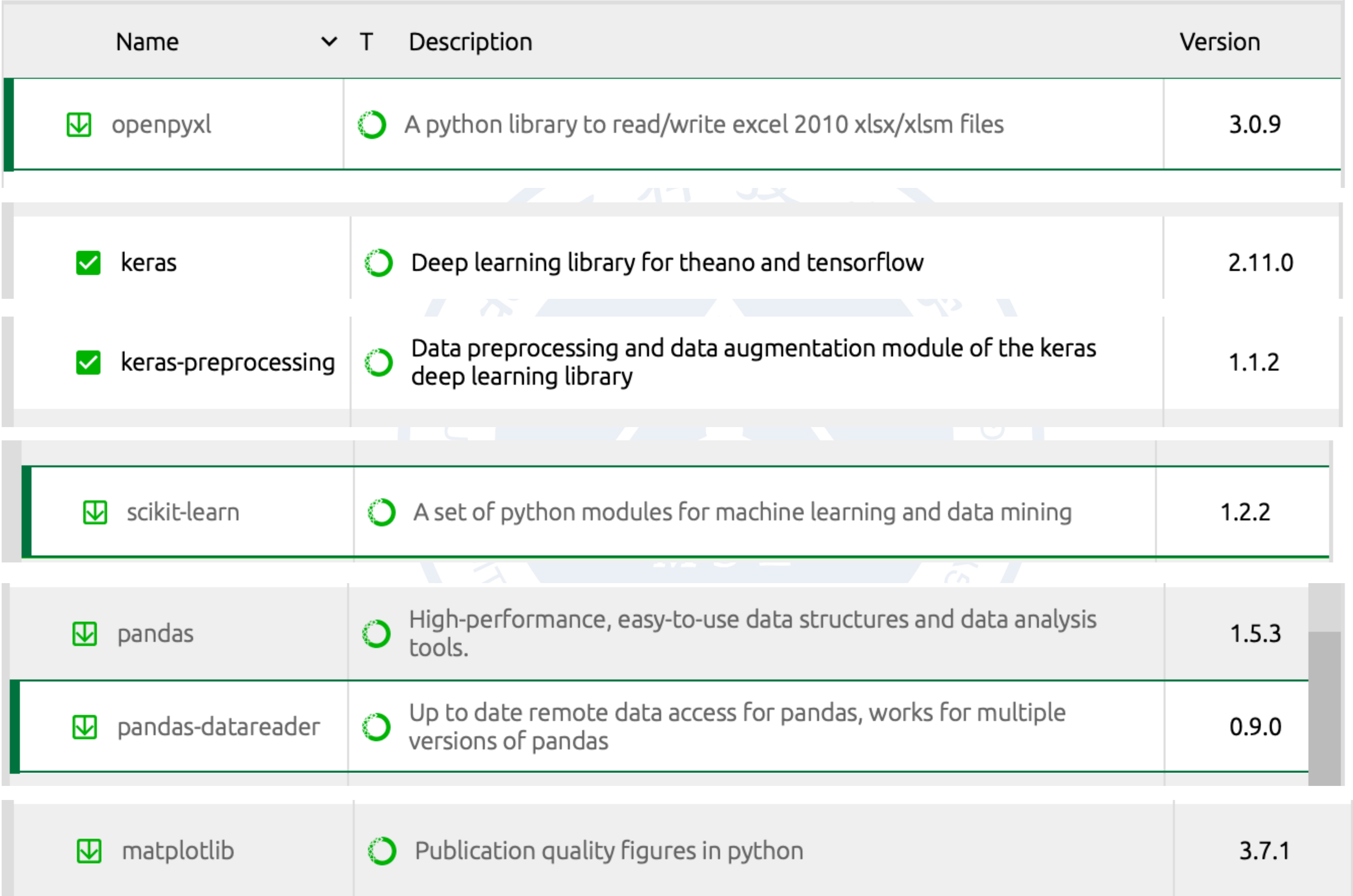
20 packages available matching "tensorflow"



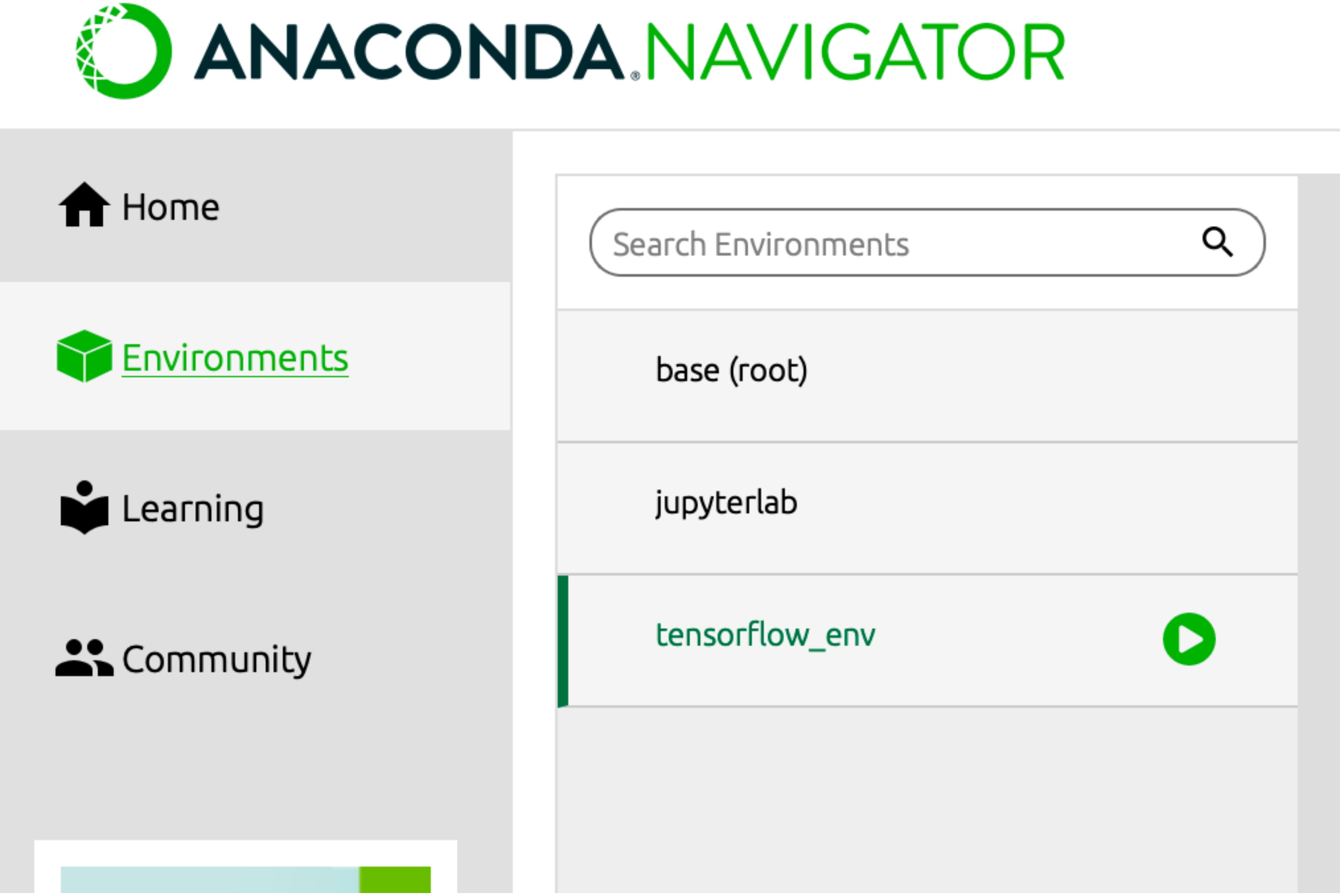
ANACONDA NAVIGATOR

- ◆再试一下
- ◆发生了什么?
- ◆如何解决?

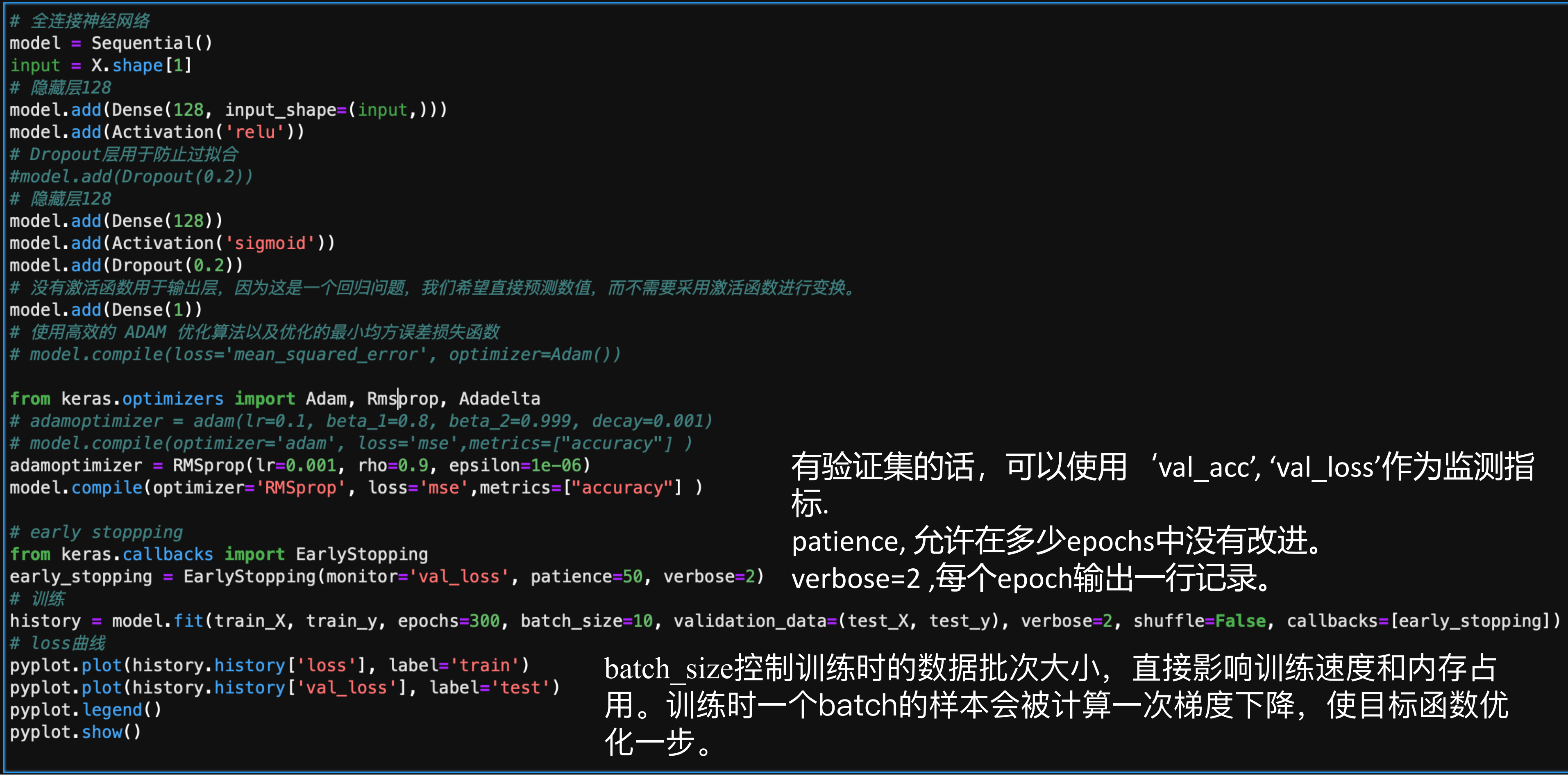
3. 机器学习算法



	Name ⌄	T	Description	Version
⬇	openpyxl	○	A python library to read/write excel 2010 xlsx/xlsm files	3.0.9
☑	keras	○	Deep learning library for theano and tensorflow	2.11.0
☑	keras-preprocessing	○	Data preprocessing and data augmentation module of the keras deep learning library	1.1.2
⬇	scikit-learn	○	A set of python modules for machine learning and data mining	1.2.2
⬇	pandas	○	High-performance, easy-to-use data structures and data analysis tools.	1.5.3
⬇	pandas-datareader	○	Up to date remote data access for pandas, works for multiple versions of pandas	0.9.0
⬇	matplotlib	○	Publication quality figures in python	3.7.1


ANACONDA NAVIGATOR
Home
Environments
Learning
Community
Search Environments
base (root)
jupyterlab
tensorflow_env

3. 机器学习算法



```
# 全连接神经网络
model = Sequential()
input = X.shape[1]
# 隐藏层128
model.add(Dense(128, input_shape=(input,)))
model.add(Activation('relu'))
# Dropout层用于防止过拟合
#model.add(Dropout(0.2))
# 隐藏层128
model.add(Dense(128))
model.add(Activation('sigmoid'))
model.add(Dropout(0.2))
# 没有激活函数用于输出层，因为这是一个回归问题，我们希望直接预测数值，而不需要采用激活函数进行变换。
model.add(Dense(1))
# 使用高效的 ADAM 优化算法以及优化的最小均方误差损失函数
# model.compile(loss='mean_squared_error', optimizer=Adam())

from keras.optimizers import Adam, Rmsprop, Adadelta
# adamoptimizer = adam(lr=0.1, beta_1=0.8, beta_2=0.999, decay=0.001)
# model.compile(optimizer='adam', loss='mse',metrics=["accuracy"] )
adamoptimizer = RMSprop(lr=0.001, rho=0.9, epsilon=1e-06)
model.compile(optimizer='RMSprop', loss='mse',metrics=["accuracy"] )

# early stoppping
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=50, verbose=2)
# 训练
history = model.fit(train_X, train_y, epochs=300, batch_size=10, validation_data=(test_X, test_y), verbose=2, shuffle=False, callbacks=[early_stopping])
# loss曲线
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```

有验证集的话，可以使用 ‘val_acc’, ‘val_loss’作为监测指标.

patience, 允许在多少epochs中没有改进。

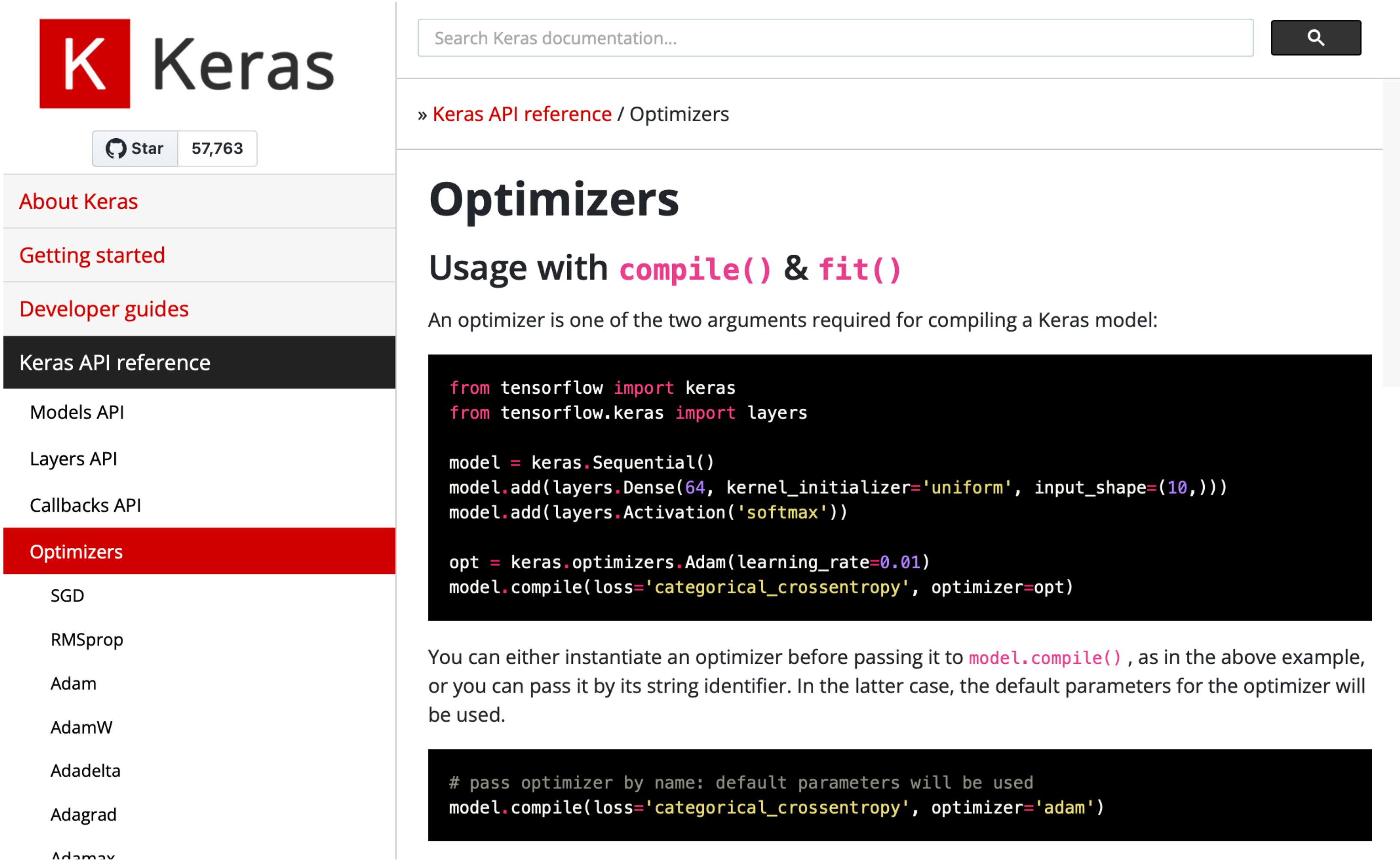
verbose=2 ,每个epoch输出一行记录。

batch_size控制训练时的数据批次大小，直接影响训练速度和内存占用。训练时一个batch的样本会被计算一次梯度下降，使目标函数优化一步。

dropout会在每次训练迭代中随机关闭一些神经元。被关闭的神经元将不参与前向传播和反向传播。这样，每个神经元都需要学习如何与其他神经元共同工作，而不是依赖某些特定的神经元。这种随机性有助于防止网络对训练数据的过度拟合。

- ◆发生了什么?
- ◆如何解决?

3. 机器学习算法



» Keras API reference / Optimizers

Optimizers

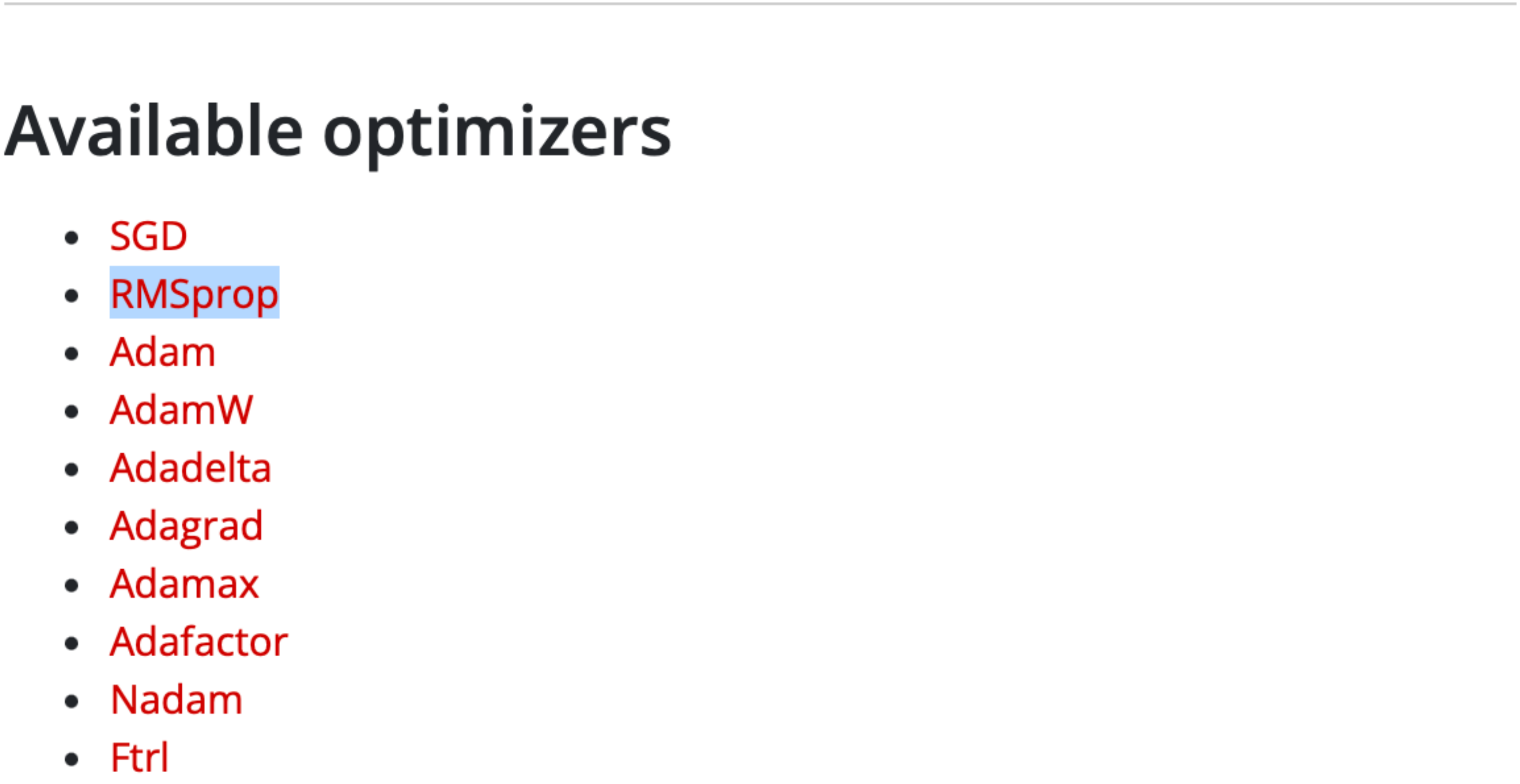
Usage with `compile()` & `fit()`

An optimizer is one of the two arguments required for compiling a Keras model:

```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential()
model.add(layers.Dense(64, kernel_initializer='uniform', input_shape=(10,)))
model.add(layers.Activation('softmax'))

opt = keras.optimizers.Adam(learning_rate=0.01)
model.compile(loss='categorical_crossentropy', optimizer=opt)
```

You can either instantiate an optimizer before passing it to `model.compile()` , as in the above example, or you can pass it by its string identifier. In the latter case, the default parameters for the optimizer will be used.

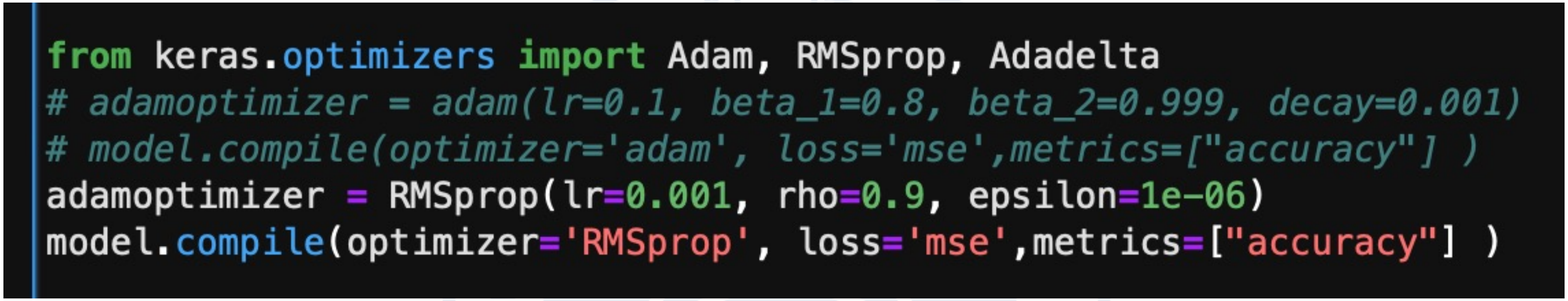
```
# pass optimizer by name: default parameters will be used
model.compile(loss='categorical_crossentropy', optimizer='adam')
```



Available optimizers

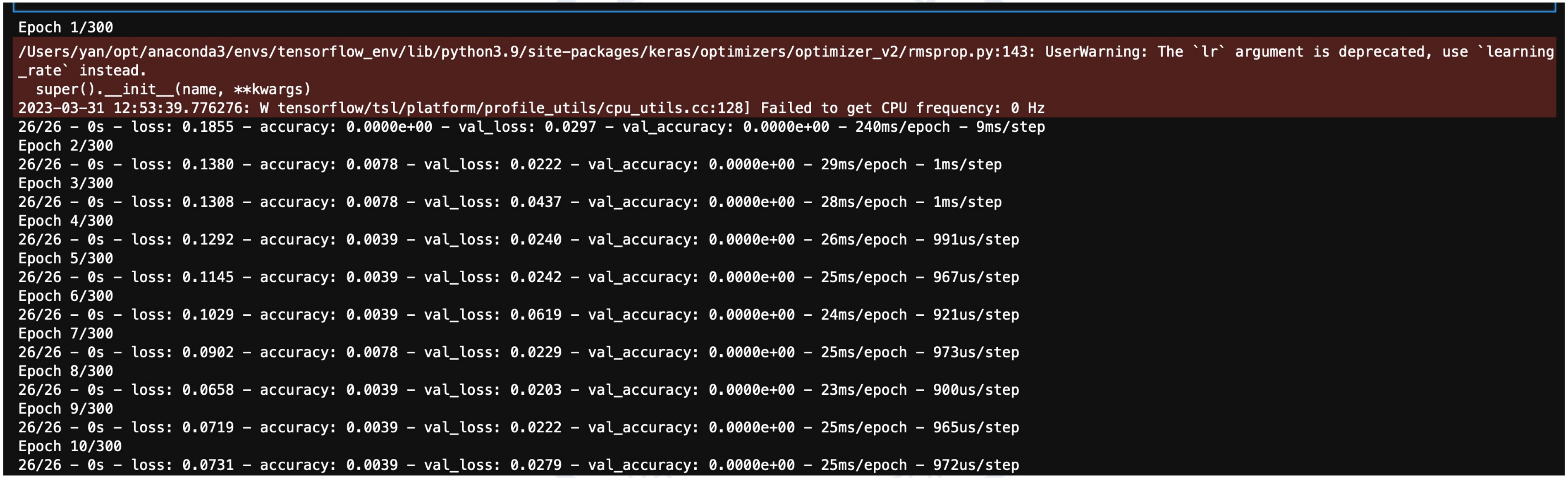
- SGD
- RMSprop
- Adam
- AdamW
- Adadelta
- Adagrad
- Adamax
- Adafactor
- Nadam
- Ftrl

3. 机器学习算法



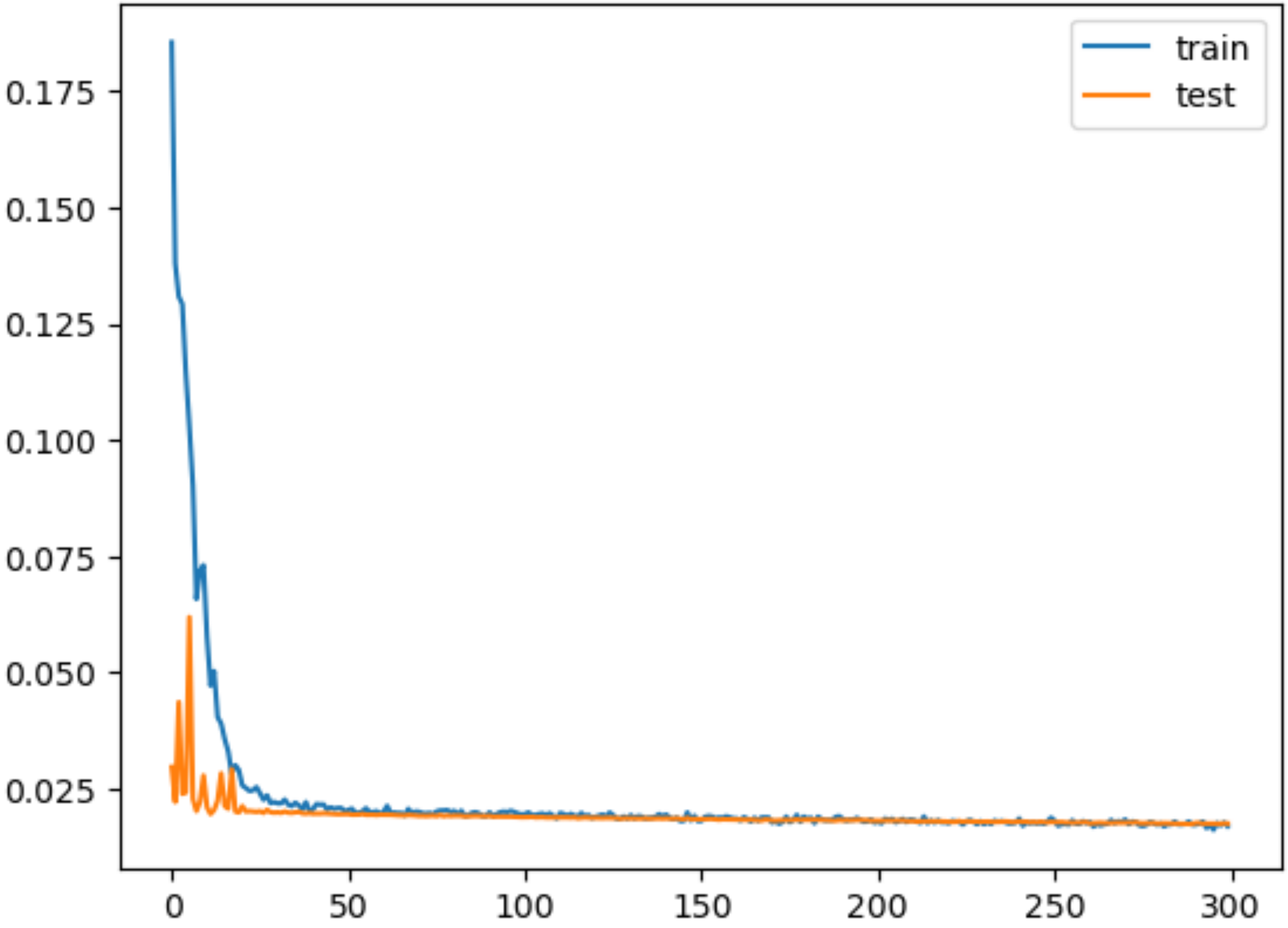
```
from keras.optimizers import Adam, RMSprop, Adadelta
# adamoptimizer = adam(lr=0.1, beta_1=0.8, beta_2=0.999, decay=0.001)
# model.compile(optimizer='adam', loss='mse',metrics=["accuracy"] )
adamoptimizer = RMSprop(lr=0.001, rho=0.9, epsilon=1e-06)
model.compile(optimizer='RMSprop', loss='mse',metrics=["accuracy"] )
```

3. 机器学习算法

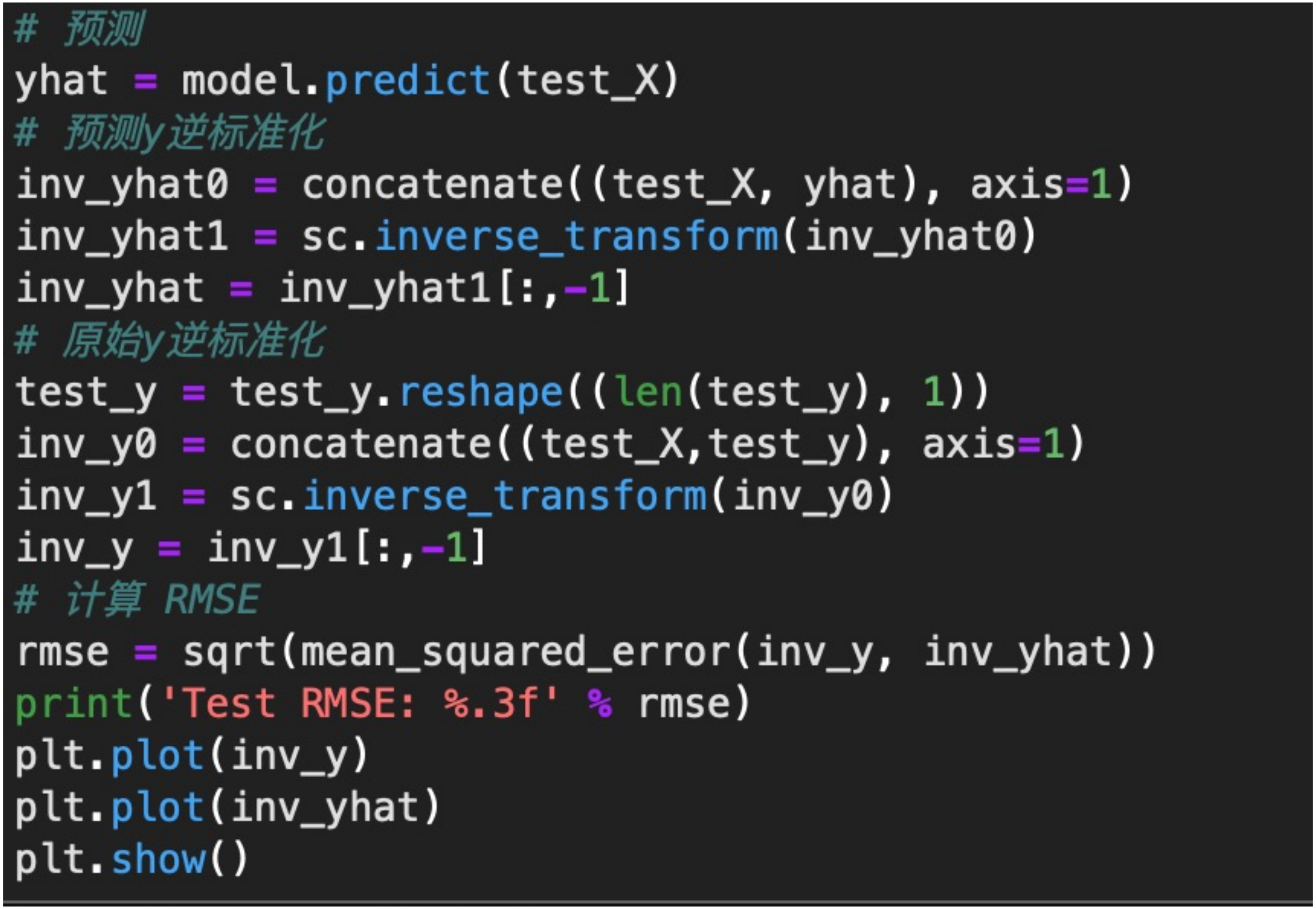


```
Epoch 1/300
/Users/yan/opt/anaconda3/envs/tensorflow_env/lib/python3.9/site-packages/keras/optimizers/optimizer_v2/rmsprop.py:143: UserWarning: The `lr` argument is deprecated, use `learning
_rate` instead.
  super().__init__(name, **kwargs)
2023-03-31 12:53:39.776276: W tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU frequency: 0 Hz
26/26 - 0s - loss: 0.1855 - accuracy: 0.0000e+00 - val_loss: 0.0297 - val_accuracy: 0.0000e+00 - 240ms/epoch - 9ms/step
Epoch 2/300
26/26 - 0s - loss: 0.1380 - accuracy: 0.0078 - val_loss: 0.0222 - val_accuracy: 0.0000e+00 - 29ms/epoch - 1ms/step
Epoch 3/300
26/26 - 0s - loss: 0.1308 - accuracy: 0.0078 - val_loss: 0.0437 - val_accuracy: 0.0000e+00 - 28ms/epoch - 1ms/step
Epoch 4/300
26/26 - 0s - loss: 0.1292 - accuracy: 0.0039 - val_loss: 0.0240 - val_accuracy: 0.0000e+00 - 26ms/epoch - 991us/step
Epoch 5/300
26/26 - 0s - loss: 0.1145 - accuracy: 0.0039 - val_loss: 0.0242 - val_accuracy: 0.0000e+00 - 25ms/epoch - 967us/step
Epoch 6/300
26/26 - 0s - loss: 0.1029 - accuracy: 0.0039 - val_loss: 0.0619 - val_accuracy: 0.0000e+00 - 24ms/epoch - 921us/step
Epoch 7/300
26/26 - 0s - loss: 0.0902 - accuracy: 0.0078 - val_loss: 0.0229 - val_accuracy: 0.0000e+00 - 25ms/epoch - 973us/step
Epoch 8/300
26/26 - 0s - loss: 0.0658 - accuracy: 0.0039 - val_loss: 0.0203 - val_accuracy: 0.0000e+00 - 23ms/epoch - 900us/step
Epoch 9/300
26/26 - 0s - loss: 0.0719 - accuracy: 0.0039 - val_loss: 0.0222 - val_accuracy: 0.0000e+00 - 25ms/epoch - 965us/step
Epoch 10/300
26/26 - 0s - loss: 0.0731 - accuracy: 0.0039 - val_loss: 0.0279 - val_accuracy: 0.0000e+00 - 25ms/epoch - 972us/step
```

3. 机器学习算法

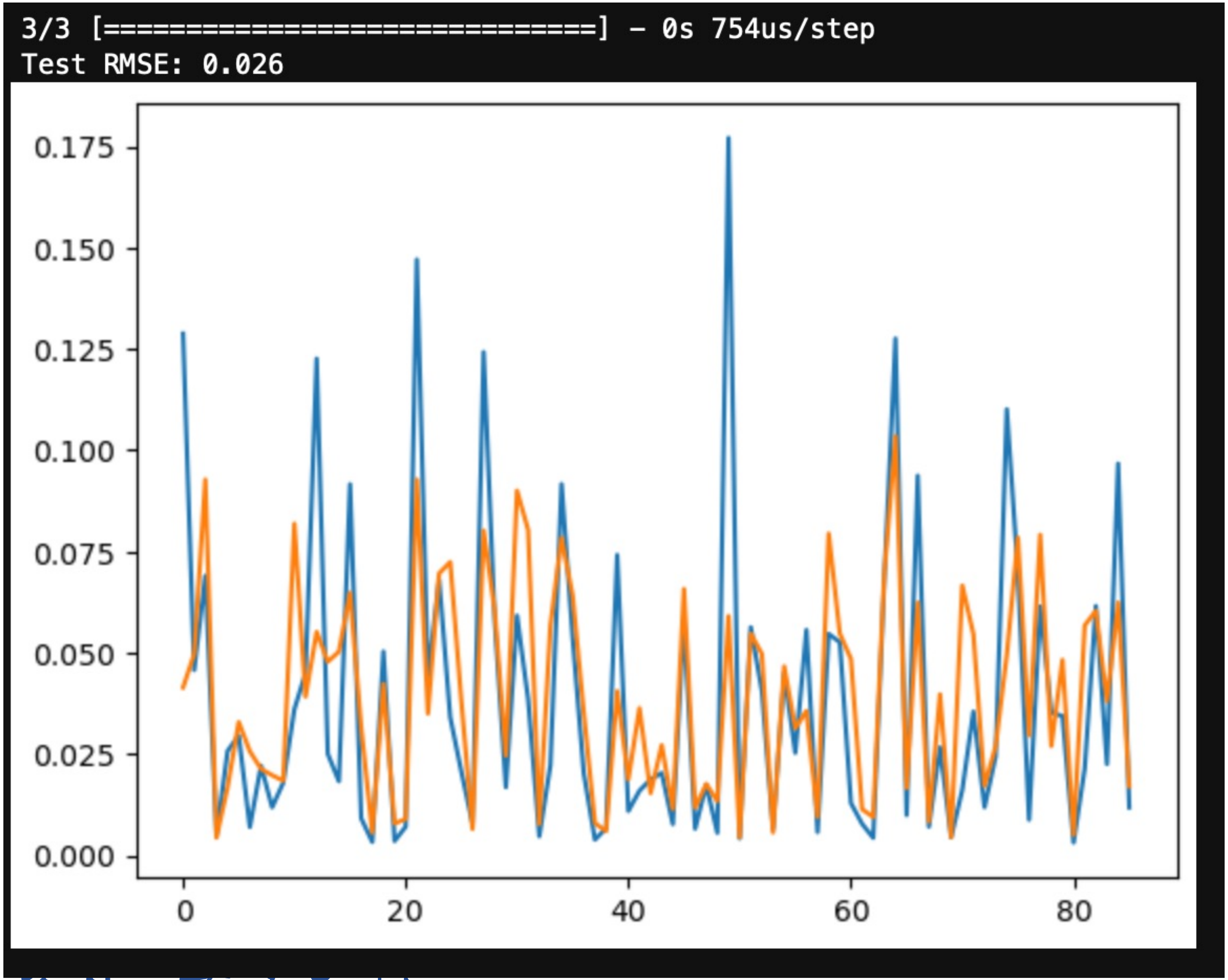

0.175
0.150
0.125
0.100
0.075
0.050
0.025
0
50
100
150
200
250
300
train
test

3. 机器学习算法



```
# 预测
yhat = model.predict(test_X)
# 预测y逆标准化
inv_yhat0 = concatenate((test_X, yhat), axis=1)
inv_yhat1 = sc.inverse_transform(inv_yhat0)
inv_yhat = inv_yhat1[:,-1]
# 原始y逆标准化
test_y = test_y.reshape((len(test_y), 1))
inv_y0 = concatenate((test_X,test_y), axis=1)
inv_y1 = sc.inverse_transform(inv_y0)
inv_y = inv_y1[:,-1]
# 计算 RMSE
rmse = sqrt(mean_squared_error(inv_y, inv_yhat))
print('Test RMSE: %.3f' % rmse)
plt.plot(inv_y)
plt.plot(inv_yhat)
plt.show()
```

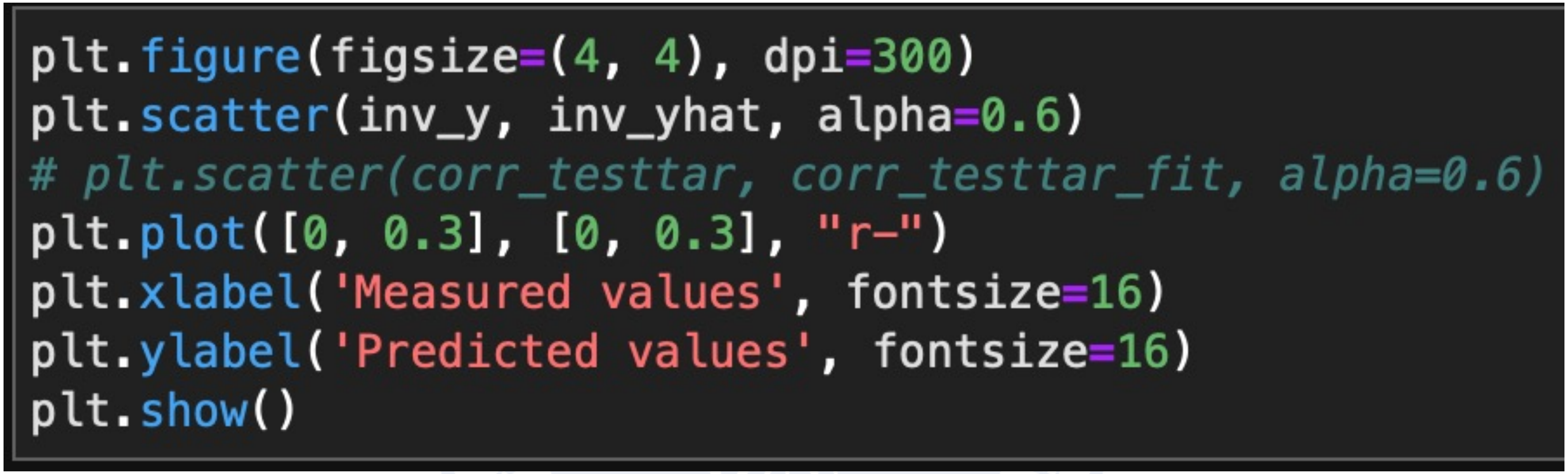
3. 机器学习算法



```
3/3 [==============================] - 0s 754us/step
Test RMSE: 0.026
```

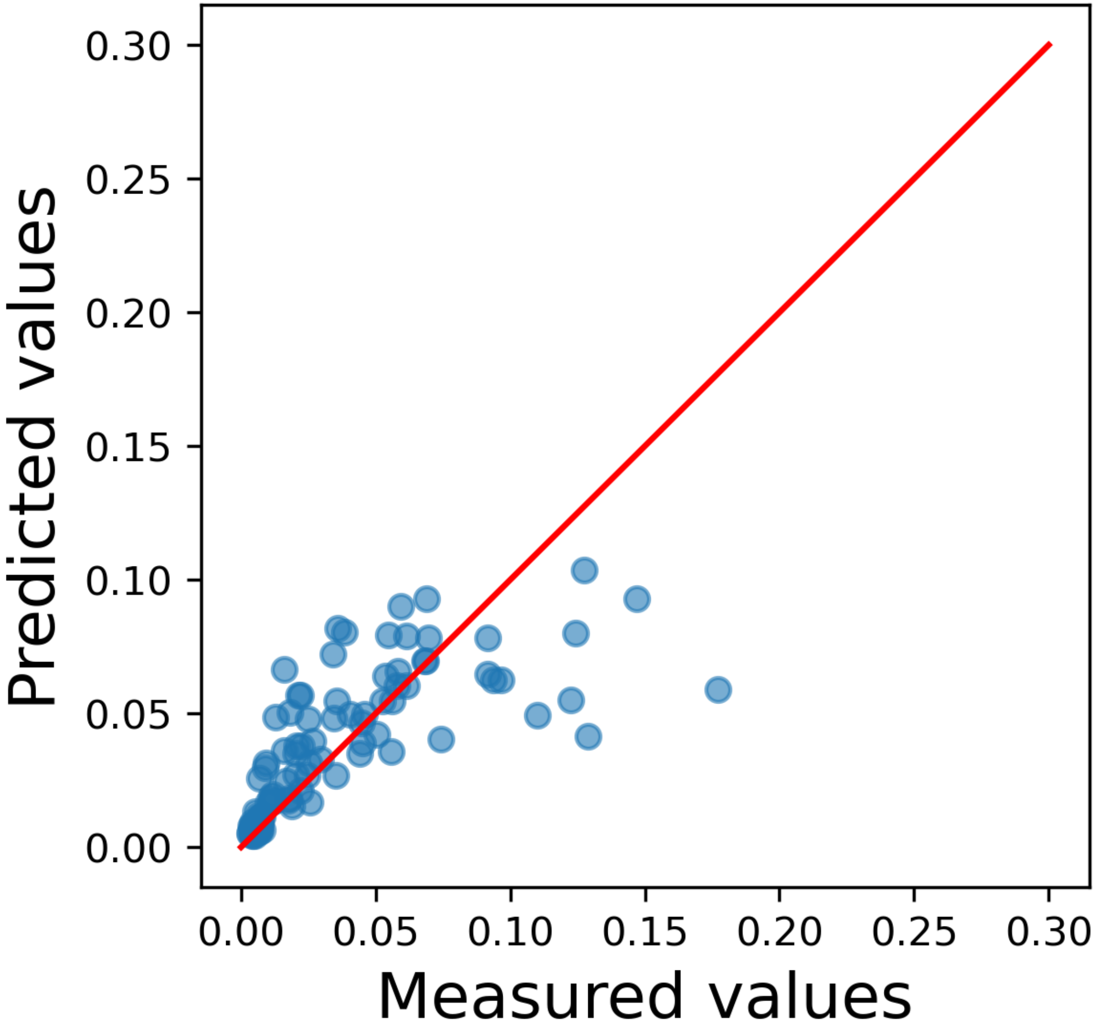
学厚质朴 百炼成材

3. 机器学习算法

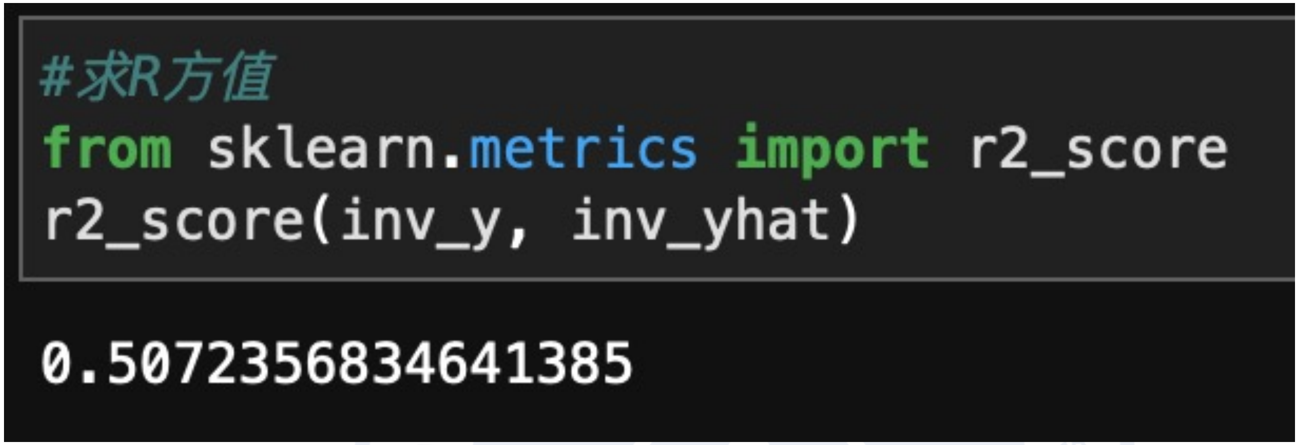


```
plt.figure(figsize=(4, 4), dpi=300)
plt.scatter(inv_y, inv_yhat, alpha=0.6)
# plt.scatter(corr_testtar, corr_testtar_fit, alpha=0.6)
plt.plot([0, 0.3], [0, 0.3], "r-")
plt.xlabel('Measured values', fontsize=16)
plt.ylabel('Predicted values', fontsize=16)
plt.show()
```

3. 机器学习算法


Predicted values
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Measured values
0.00
0.05
0.10
0.15
0.20
0.25
0.30

3. 机器学习算法



```
#求R方值
from sklearn.metrics import r2_score
r2_score(inv_y, inv_yhat)

0.5072356834641385
```

学厚质朴　百炼成材