pdf2ipynb

July 2, 2021

1 Objective

As a good example of extracting information from PDF files, we set out to convert a Jupyter notebook in PDF form to its original `.ipynb` form.

I don't know if this is hard. Let's get started.

[1]:
```
from pathlib import Path
from PyPDF2 import PdfFileReader
```

N.B.

- The `n_pages` assignment should be kept in the `with` context; otherwise, one'd get a `ValueError: seek of closed file`
  - same as `pdf.getPage()`

[2]:
```
path_pdf = Path("pdf2ipynb.pdf")
with open(path_pdf, "rb") as f:
    pdf = PdfFileReader(f)
    n_pages = pdf.getNumPages()
    page00 = pdf.getPage(0)
    print(page00)
pdf
```

```
{'/Resources': IndirectObject(21, 0), '/Type': '/Page', '/Parent':
IndirectObject(69, 0), '/Contents': [IndirectObject(20, 0)], '/MediaBox': [0, 0,
612, 792]}
```

[2]:
```
<PyPDF2.pdf.PdfFileReader at 0x7f0f180c9290>
```

[3]:
```
n_pages
```

[3]:
```
8
```

[4]:
```
info = pdf.getDocumentInfo()
info
```

[4]:
```
{'/Creator': 'LaTeX with hyperref',
 '/Producer': 'xdvipdfmx (20210318)',
 '/CreationDate': "D:20210702211923+07'00'"}
```

```
[5]: [ s for s in dir(pdf) if not s.startswith("_")]
```

```
[5]: ['cacheGetIndirectObject',
      'cacheIndirectObject',
      'decrypt',
      'documentInfo',
      'flattenedPages',
      'getDestinationPageNumber',
      'getDocumentInfo',
      'getFields',
      'getFormTextFields',
      'getIsEncrypted',
      'getNamedDestinations',
      'getNumPages',
      'getObject',
      'getOutlines',
      'getPage',
      'getPageLayout',
      'getPageMode',
      'getPageNumber',
      'getXmpMetadata',
      'isEncrypted',
      'namedDestinations',
      'numPages',
      'outlines',
      'pageLayout',
      'pageMode',
      'pages',
      'read',
      'readNextEndLine',
      'readObjectHeader',
      'resolvedObjects',
      'stream',
      'strict',
      'trailer',
      'xmpMetadata',
      'xref',
      'xrefIndex',
      'xref_objStm']
```

The above few cells came from the real python's tutorial, but only after reading the first few paragraphs of it did I find out that to extract the content of a PDF file, people seems to not recommend `pypdf2`; instead, people suggest using `pdfminer` (or `pdfminer.six`).

I chose to install `pip install pdfminer.six` because it seems to be a fork of `pdfminer` that is being constantly maintained, whereas `pdfminer` itself seems to be free of maintainance.

[6]:

```
import pdfminer
dir(pdfminer)
```

[6]:

```
['__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
 '__version__',
 'sys',
 'warnings']
```

[7]:

```
from pdfminer.high_level import extract_text
text = extract_text(path_pdf)
text
```

[7]:

```
'pdf2ipynb\n\nJuly 2, 2021\n\n1 Objective\n\nAs a good example of extracting
information from PDF files, we set out to convert a Jupyter\nnotebook in PDF
form to its original .ipynb form.\n\n• The n_pages assignment should be kept in
the with context; otherwise, one’d get a\n\n[19]: path_pdf =
Path("JupyterNotebook-LinearRegression-MultipleInput.pdf")\n\nI don’t know if
this is hard. Let’s get started.\n\n[1]: from pathlib import Path\n\nfrom PyPDF2
import PdfFileReader\n\nN.B.\n\nValueError: seek of closed file\n\n- same as
pdf.getPage()\n\nwith open(path_pdf, "rb") as f:\npdf =
PdfFileReader(f)\nn_pages = pdf.getNumPages()\npage00 =
pdf.getPage(0)\nprint(page00)\n\npdf\n\n{\'/Resources\': IndirectObject(19, 0),
\'/Type\': \'/Page\', \'/Parent\':\nIndirectObject(54, 0), \'/Contents\':
[IndirectObject(18, 0)], \'/MediaBox\': [0, 0,\n612, 792]}\n\n[19]:
<PyPDF2.pdf.PdfFileReader at 0x7fecc84ab2d0>\n\n[17]: n_pages\n\n[17]: 5\n\n[6]:
info = pdf.getDocumentInfo()\n\ninfo\n\n[6]: {\'/Creator\': \'LaTeX with
hyperref package\',\n\n\'/Producer\': \'XeTeX 0.99998\',\n\'/CreationDate\':
"D:20210625090544+07\'00\'"}\n\n1\n\n\x0c[4]: [ s for s in dir(pdf) if not
s.startswith("_")]\n\n[4]: [\'cacheGetIndirectObject\',\n\n\'cacheIndirectObject
\',\n\'decrypt\',\n\'documentInfo\',\n\'flattenedPages\',\n\'getDestinationPageN
umber\',\n\'getDocumentInfo\',\n\'getFields\',\n\'getFormTextFields\',\n\'getIsE
ncrypted\',\n\'getNamedDestinations\',\n\'getNumPages\',\n\'getObject\',\n\'getO
utlines\',\n\'getPage\',\n\'getPageLayout\',\n\'getPageMode\',\n\'getPageNumber\
',\n\'getXmpMetadata\',\n\'isEncrypted\',\n\'namedDestinations\',\n\'numPages\',
\n\'outlines\',\n\'pageLayout\',\n\'pageMode\',\n\'pages\',\n\'read\',\n\'readNe
xtEndLine\',\n\'readObjectHeader\',\n\'resolvedObjects\',\n\'stream\',\n\'strict
\',\n\'trailer\',\n\'xmpMetadata\',\n\'xref\',\n\'xrefIndex\',\n\'xref_objStm\']
\n\nThe above few cells came from the real python’s tutorial, but only after
reading the first few\nparagraphs of it did I find out that to extract the
```

```
content of a PDF file, people seems to not\nrecommend pypdf2; instead, people
suggest using pdfminer (or pdfminer.six).\n\nI chose to install pip install
pdfminer.six because it seems to be a fork of pdfminer that is\nbeing constantly
maintained, whereas pdfminer itself seems to be free of
maintainance.\n\n2\n\n\x0c[24]: import pdfminer\n\ndir(pdfminer)\n\n[24]: [\'__b
uiltins__\',\n\n\'__cached__\',\n\'__doc__\',\n\'__file__\',\n\'__loader__\',\n\
'__name__\',\n\'__package__\',\n\'__path__\',\n\'__spec__\',\n\'__version__\',\n
\'sys\',\n\'warnings\']\n\n[26]: from pdfminer.high_level import
extract_text\n\ntext = extract_text(path_pdf)\ntext\n\n[26]: \'LinearRegression-
MultipleInput-TensorBoard\\n\\nJune 25, 2021\\n\\n1 Import and\ncheck TensorFlow
version\\n\\n[14]: import numpy as np\\n\\nimport tensorflow as\ntf\\nimport
matplotlib.pyplot as plt\\nprint(tf.__version__)\\n\\n2.5.0\\n\\n2\nDownload and
check Boston Housing dataset\\n\\n[15]: from\ntensorflow.keras.datasets import
boston_housing\\n\\n(x_train, y_train), (x_test,\ny_test) =
boston_housing.load_data()\\n\\ndef normalize(x,y):\\n\\nmean_y
=\ny.mean(axis=0)\\nstd_y = y.std(axis=0)\\n\\nmean_x = x.mean(axis=0)\\nstd_x
=\nx.std(axis=0)\\n\\nx_norm = (x - mean_x)/std_x\\ny_norm = (y
-\nmean_y)/std_y\\n\\nreturn x_norm, y_norm\\n\\nx_train, y_train =
normalize(x_train,\ny_train)\\nx_test, y_test\\ny_test)\\nprint(\\\'Train:\\\',
x_train.shape,\ny_train.shape)\\nprint(\\\'Test:\\\', x_test.shape,
y_test.shape)\\n\\n=\nnormalize(x_test,\\n\\nTrain: (404, 13) (404,)\\nTest:
(102, 13)\n(102,)\\n\\n1\\n\\n\\x0c3 Prepare the model:\\n\\nEquation: y = ax +
b\\n- a,b:\nparameters that we need to find - x,y: observed data from the
reality\\n\\n[16]: a\n= tf.Variable(tf.random.uniform(shape=[13], minval=-1.,
maxval=1.))\\n\\nb =\ntf.Variable(tf.random.uniform(shape=[], minval=0.,
maxval=.5))\\n\\ndef\npredict(x):\\n\\nreturn tf.reduce_sum(a * x, 1) +
b\\n\\ndef mse(groundtruth,\nprediction):\\n\\nreturn
tf.reduce_mean(tf.square(groundtruth-
prediction))\\n\\ndef\ntest_the_model():\\n\\nprint(f\\\'Model:
a={np.around(a.numpy(),3)},\nb={b.numpy()}\\\')\\ny_test_hat =
predict(x_test)\\nerror = mse(y_test,\ny_test_hat)\\nprint(\\\'Test MSE:\\\',\ne
rror.numpy())\\n\\nplt.figure(figsize=[5,5])\\nplt.scatter(y_test,\ny_test_hat)\
\nplt.title("House\\\'s Price
Prediction")\\nplt.xlim([-6,6]);\nplt.ylim([-6,6])\\nplt.xlabel("True
Price")\\nplt.ylabel("Predicted\nPrice")\\nplt.show()\\n\\ndef
train(learning_rate=1e-1, epochs=5):\\n\\nglobal\na\\nglobal b\\n\\nwriter = tf.
summary.create_file_writer("./tmp/mylogs")\\n\\nwith\nwriter.as_default():\\n\\n
for i in range(epochs):\\n\\nwith\n\n3\n\n\x0ctf.GradientTape(persistent=True)
as g:\\n\\nloss = mse(y_train,\npredict(x_train))\\n\\nprint(f"Epoch {i+1},
Loss: ", loss.numpy())\\n\\ngrad_a =\ng.gradient(loss, a)\\ngrad_b =
g.gradient(loss, b)\\n\\na.assign_sub(learning_rate\n*
grad_a)\\nb.assign_sub(learning_rate
*\ngrad_b)\\n\\n2\\n\\n\\x0ctf.summary.scalar("train-loss",
loss,\nstep=i)\\ntf.summary.scalar("train-loss-2",
2*loss,\nstep=i)\\nwriter.flush()\\n\\ntest_the_model()\\n\\nModel: a=[-0.933
-0.083 0.089\n-0.365 -0.718 -0.586
-0.367\\n\\n0.317\\n\\n0.525\\n\\n0.804\\n\\n-0.548 -0.492
```

```
-0.608],\nb=0.10259240865707397\\n\\nTest MSE: 2.6259625\\n\\n4 Train the
model\\n\\n[ ]:\ntrain(1e-1, epochs=50)\\n\\n3\\n\\n\\x0c5 Test the model after
training\\n\\n[20]:\ntest_the_model()\\n\\nModel: a=[-0.118 0.14 -0.012
0.111\n-0.265\\n\\n0.263\\n\\n0.02\\n\\n-0.381\\n\\n0.261 -0.15\\n\\n-0.215
0.088 -0.436],\nb=6.539222940915579e-09\\n\\nTest MSE: 0.24517176\\n\\n6 View
TensorBoard\\n\\nNote:\nYou should select this cell and call to Interrupt the
Kernel after finishing\nchecking Tensor-\\nBoard\\n\\n[ ]: !tensorboard --logdir
./tmp/mylogs\\n\\n4\\n\\n\\x0c7\nHomework\\n\\n1. Apply denormalization for the
testing function\\n2. Apply\nStochastic Gradient Descent for training:\\n\\n•
1-batch\\n• mini-batch\\n\\n3. Try\nother loss functions:\\n\\n• Root Mean
Squared Error,\\n• Mean Absolute Error,\\n•\nCombined
RMSE+MAE+MSE\\n\\n5\\n\\n\\x0c\'\n\n[27]: print(text)\n\nJune 25,
2021\n\nLinearRegression-MultipleInput-TensorBoard\n\n1 Import and check
TensorFlow version\n\n[14]: import numpy as np\n\nimport tensorflow as
tf\nimport matplotlib.pyplot as plt\nprint(tf.__version__)\n\n2.5.0\n\n2
Download and check Boston Housing dataset\n\n[15]: from
tensorflow.keras.datasets import boston_housing\n\n(x_train, y_train), (x_test,
y_test) = boston_housing.load_data()\n\ndef normalize(x,y):\n\nmean_y =
y.mean(axis=0)\nstd_y = y.std(axis=0)\n\nmean_x =
x.mean(axis=0)\n\n4\n\n\x0cstd_x = x.std(axis=0)\n\nx_norm = (x -
mean_x)/std_x\ny_norm = (y - mean_y)/std_y\n\nreturn x_norm, y_norm\n\nx_train,
y_train = normalize(x_train, y_train)\nx_test,
y_test\ny_test)\nprint(\'Train:\', x_train.shape,
y_train.shape)\nprint(\'Test:\', x_test.shape, y_test.shape)\n\n=
normalize(x_test,\n\nTrain: (404, 13) (404,)\nTest: (102, 13) (102,)\n\n1\n\n3
Prepare the model:\n\nEquation: y = ax + b\n- a,b: parameters that we need to
find - x,y: observed data from the reality\n\n[16]: a =
tf.Variable(tf.random.uniform(shape=[13], minval=-1., maxval=1.))\n\nb =
tf.Variable(tf.random.uniform(shape=[], minval=0., maxval=.5))\n\ndef
predict(x):\n\nreturn tf.reduce_sum(a * x, 1) + b\n\ndef mse(groundtruth,
prediction):\n\nreturn tf.reduce_mean(tf.square(groundtruth-prediction))\n\ndef
test_the_model():\n\nprint(f\'Model: a={np.around(a.numpy(),3)},
b={b.numpy()}\')\ny_test_hat = predict(x_test)\nerror = mse(y_test,
y_test_hat)\nprint(\'Test MSE:\',
error.numpy())\n\nplt.figure(figsize=[5,5])\nplt.scatter(y_test,
y_test_hat)\nplt.title("House\'s Price Prediction")\nplt.xlim([-6,6]);
plt.ylim([-6,6])\n\n5\n\n\x0cwriter =
tf.summary.create_file_writer("./tmp/mylogs")\n\nplt.xlabel("True
Price")\nplt.ylabel("Predicted Price")\nplt.show()\n\ndef
train(learning_rate=1e-1, epochs=5):\n\nglobal a\nglobal b\n\nwith
writer.as_default():\n\nfor i in range(epochs):\n\nwith
tf.GradientTape(persistent=True) as g:\n\nloss = mse(y_train,
predict(x_train))\n\nprint(f"Epoch {i+1}, Loss: ", loss.numpy())\n\ngrad_a =
g.gradient(loss, a)\ngrad_b = g.gradient(loss, b)\n\na.assign_sub(learning_rate
* grad_a)\nb.assign_sub(learning_rate * grad_b)\n\ntf.summary.scalar("train-
loss", loss, step=i)\n\ntf.summary.scalar("train-loss-2", 2*loss,
step=i)\nwriter.flush()\n\ntest_the_model()\n\nModel: a=[-0.933 -0.083 0.089
```

```
-0.365 -0.718 -0.586 -0.367\n\n2\n\n0.317\n\n0.525\n\n0.804\n\n-0.548 -0.492
-0.608], b=0.10259240865707397\n\nTest MSE: 2.6259625\n\n4 Train the
model\n\n6\n\n\x0c3\n\n4\n\n5\n\n[ ]: train(1e-1, epochs=50)\n\n5 Test the model
after training\n\n[20]: test_the_model()\n\nModel: a=[-0.118 0.14 -0.012 0.111
-0.265\n\n0.263\n\n0.02\n\n-0.381\n\n0.261 -0.15\n\n-0.215 0.088 -0.436],
b=6.539222940915579e-09\n\nTest MSE: 0.24517176\n\n6 View TensorBoard\n\nNote:
You should select this cell and call to Interrupt the Kernel after\nfinishing
checking Tensor-\nBoard\n\n[ ]: !tensorboard --logdir ./tmp/mylogs\n\n1. Apply
denormalization for the testing function\n2. Apply Stochastic Gradient Descent
for training:\n\n7 Homework\n\n• 1-batch\n• mini-batch\n\n3. Try other loss
functions:\n\n• Root Mean Squared Error,\n• Mean Absolute Error,\n• Combined
RMSE+MAE+MSE\n\n7\n\n\x0c1.1 Ref.\n\n[ ]:\n\n[ ]:\n\n•
https://realpython.com/pdf-python/\n• https://pdfminersix.readthedocs.io/en/late
st/tutorial/highlevel.html\n\n8\n\n\x0c'
```

[8]:
```python
print(text)
```

```
pdf2ipynb

July 2, 2021

1 Objective

As a good example of extracting information from PDF files, we set out to
convert a Jupyter
notebook in PDF form to its original .ipynb form.

• The n_pages assignment should be kept in the with context; otherwise, one'd
get a

[19]: path_pdf = Path("JupyterNotebook-LinearRegression-MultipleInput.pdf")

I don't know if this is hard. Let's get started.

[1]: from pathlib import Path

from PyPDF2 import PdfFileReader

N.B.

ValueError: seek of closed file

- same as pdf.getPage()

with open(path_pdf, "rb") as f:
pdf = PdfFileReader(f)
n_pages = pdf.getNumPages()
page00 = pdf.getPage(0)
```

```
print(page00)

pdf

{'/Resources': IndirectObject(19, 0), '/Type': '/Page', '/Parent':
IndirectObject(54, 0), '/Contents': [IndirectObject(18, 0)], '/MediaBox': [0, 0,
612, 792]}

[19]: <PyPDF2.pdf.PdfFileReader at 0x7fecc84ab2d0>

[17]: n_pages

[17]: 5

[6]: info = pdf.getDocumentInfo()

info

[6]: {'/Creator': 'LaTeX with hyperref package',

'/Producer': 'XeTeX 0.99998',
'/CreationDate': "D:20210625090544+07'00'"}

1

 [4]: [ s for s in dir(pdf) if not s.startswith("_")]

[4]: ['cacheGetIndirectObject',

'cacheIndirectObject',
'decrypt',
'documentInfo',
'flattenedPages',
'getDestinationPageNumber',
'getDocumentInfo',
'getFields',
'getFormTextFields',
'getIsEncrypted',
'getNamedDestinations',
'getNumPages',
'getObject',
'getOutlines',
'getPage',
'getPageLayout',
'getPageMode',
'getPageNumber',
'getXmpMetadata',
'isEncrypted',
'namedDestinations',
'numPages',
'outlines',
'pageLayout',
'pageMode',
'pages',
'read',
'readNextEndLine',
'readObjectHeader',
'resolvedObjects',
'stream',
'strict',
'trailer',
'xmpMetadata',
'xref',
'xrefIndex',
'xref_objStm']
```

The above few cells came from the real python’s tutorial, but only after reading the first few paragraphs of it did I find out that to extract the content of a PDF file, people seems to not recommend pypdf2; instead, people suggest using pdfminer (or pdfminer.six).

I chose to install pip install pdfminer.six because it seems to be a fork of pdfminer that is being constantly maintained, whereas pdfminer itself seems to be free of maintainance.

2

```
 [24]: import pdfminer

dir(pdfminer)
```

```
[24]: ['__builtins__',

'__cached__',
'__doc__',
'__file__',
'__loader__',
'__name__',
'__package__',
'__path__',
'__spec__',
'__version__',
'sys',
'warnings']
```

```
[26]: from pdfminer.high_level import extract_text

text = extract_text(path_pdf)
text
```

```
[26]: 'LinearRegression-MultipleInput-TensorBoard\n\nJune 25, 2021\n\n1 Import
and
check TensorFlow version\n\n[14]: import numpy as np\n\nimport tensorflow as
tf\nimport matplotlib.pyplot as plt\nprint(tf.__version__)\n\n2.5.0\n\n2
Download and check Boston Housing dataset\n\n[15]: from
tensorflow.keras.datasets import boston_housing\n\n(x_train, y_train), (x_test,
y_test) = boston_housing.load_data()\n\ndef normalize(x,y):\n\nmean_y =
y.mean(axis=0)\nstd_y = y.std(axis=0)\n\nmean_x = x.mean(axis=0)\nstd_x =
x.std(axis=0)\n\nx_norm = (x - mean_x)/std_x\ny_norm = (y -
mean_y)/std_y\n\nreturn x_norm, y_norm\n\nx_train, y_train = normalize(x_train,
y_train)\nx_test, y_test\ny_test)\nprint(\'Train:\', x_train.shape,
y_train.shape)\nprint(\'Test:\', x_test.shape, y_test.shape)\n\n=
normalize(x_test,\n\nTrain: (404, 13) (404,)\nTest: (102, 13)
(102,)\n\n1\n\n\x0c3 Prepare the model:\n\nEquation: y = ax + b\n- a,b:
parameters that we need to find - x,y: observed data from the reality\n\n[16]: a
= tf.Variable(tf.random.uniform(shape=[13], minval=-1., maxval=1.))\n\nb =
tf.Variable(tf.random.uniform(shape=[], minval=0., maxval=.5))\n\ndef
predict(x):\n\nreturn tf.reduce_sum(a * x, 1) + b\n\ndef mse(groundtruth,
prediction):\n\nreturn tf.reduce_mean(tf.square(groundtruth-prediction))\n\ndef
test_the_model():\n\nprint(f\'Model: a={np.around(a.numpy(),3)},
b={b.numpy()}\')\ny_test_hat = predict(x_test)\nerror = mse(y_test,
y_test_hat)\nprint(\'Test MSE:\',
error.numpy())\n\nplt.figure(figsize=[5,5])\nplt.scatter(y_test,
y_test_hat)\nplt.title("House\'s Price Prediction")\nplt.xlim([-6,6]);
plt.ylim([-6,6])\nplt.xlabel("True Price")\nplt.ylabel("Predicted
Price")\nplt.show()\n\ndef train(learning_rate=1e-1, epochs=5):\n\nglobal
a\nglobal b\n\nwriter = tf.summary.create_file_writer("./tmp/mylogs")\n\nwith
writer.as_default():\n\nfor i in range(epochs):\n\nwith

3

 tf.GradientTape(persistent=True) as g:\n\nloss = mse(y_train,
predict(x_train))\n\nprint(f"Epoch {i+1}, Loss: ", loss.numpy())\n\ngrad_a =
g.gradient(loss, a)\ngrad_b = g.gradient(loss, b)\n\na.assign_sub(learning_rate
* grad_a)\nb.assign_sub(learning_rate *
grad_b)\n\n2\n\n\x0ctf.summary.scalar("train-loss", loss,
step=i)\ntf.summary.scalar("train-loss-2", 2*loss,
step=i)\nwriter.flush()\n\ntest_the_model()\n\nModel: a=[-0.933 -0.083 0.089
-0.365 -0.718 -0.586 -0.367\n\n0.317\n\n0.525\n\n0.804\n\n-0.548 -0.492 -0.608],
b=0.10259240865707397\n\nTest MSE: 2.6259625\n\n4 Train the model\n\n[ ]:
train(1e-1, epochs=50)\n\n3\n\n\x0c5 Test the model after training\n\n[20]:
test_the_model()\n\nModel: a=[-0.118 0.14 -0.012 0.111
-0.265\n\n0.263\n\n0.02\n\n-0.381\n\n0.261 -0.15\n\n-0.215 0.088 -0.436],
b=6.539222940915579e-09\n\nTest MSE: 0.24517176\n\n6 View TensorBoard\n\nNote:
You should select this cell and call to Interrupt the Kernel after finishing
checking Tensor-\nBoard\n\n[ ]: !tensorboard --logdir ./tmp/mylogs\n\n4\n\n\x0c7
Homework\n\n1. Apply denormalization for the testing function\n2. Apply
Stochastic Gradient Descent for training:\n\n• 1-batch\n• mini-batch\n\n3. Try
other loss functions:\n\n• Root Mean Squared Error,\n• Mean Absolute Error,\n•
Combined RMSE+MAE+MSE\n\n5\n\n\x0c'
```

```
[27]: print(text)
```

```
June 25, 2021

LinearRegression-MultipleInput-TensorBoard

1 Import and check TensorFlow version

[14]: import numpy as np

import tensorflow as tf
import matplotlib.pyplot as plt
print(tf.__version__)

2.5.0

2 Download and check Boston Housing dataset

[15]: from tensorflow.keras.datasets import boston_housing

(x_train, y_train), (x_test, y_test) = boston_housing.load_data()

def normalize(x,y):

mean_y = y.mean(axis=0)
std_y = y.std(axis=0)

mean_x = x.mean(axis=0)

4

 std_x = x.std(axis=0)

x_norm = (x - mean_x)/std_x
y_norm = (y - mean_y)/std_y

return x_norm, y_norm

x_train, y_train = normalize(x_train, y_train)
```

```
x_test, y_test
y_test)
print('Train:', x_train.shape, y_train.shape)
print('Test:', x_test.shape, y_test.shape)

= normalize(x_test,
```

```
Train: (404, 13) (404,)
Test: (102, 13) (102,)
```

1

3 Prepare the model:

Equation: y = ax + b
- a,b: parameters that we need to find - x,y: observed data from the reality

```
[16]: a = tf.Variable(tf.random.uniform(shape=[13], minval=-1., maxval=1.))

b = tf.Variable(tf.random.uniform(shape=[], minval=0., maxval=.5))

def predict(x):

return tf.reduce_sum(a * x, 1) + b

def mse(groundtruth, prediction):

return tf.reduce_mean(tf.square(groundtruth-prediction))

def test_the_model():

print(f'Model: a={np.around(a.numpy(),3)}, b={b.numpy()}')
y_test_hat = predict(x_test)
error = mse(y_test, y_test_hat)
print('Test MSE:', error.numpy())

plt.figure(figsize=[5,5])
plt.scatter(y_test, y_test_hat)
plt.title("House's Price Prediction")
plt.xlim([-6,6]); plt.ylim([-6,6])
```

5

```
 writer = tf.summary.create_file_writer("./tmp/mylogs")

plt.xlabel("True Price")
plt.ylabel("Predicted Price")
plt.show()
```

```
def train(learning_rate=1e-1, epochs=5):

global a
global b

with writer.as_default():

for i in range(epochs):

with tf.GradientTape(persistent=True) as g:

loss = mse(y_train, predict(x_train))

print(f"Epoch {i+1}, Loss: ", loss.numpy())

grad_a = g.gradient(loss, a)
grad_b = g.gradient(loss, b)

a.assign_sub(learning_rate * grad_a)
b.assign_sub(learning_rate * grad_b)

tf.summary.scalar("train-loss", loss, step=i)

tf.summary.scalar("train-loss-2", 2*loss, step=i)
writer.flush()

test_the_model()

Model: a=[-0.933 -0.083 0.089 -0.365 -0.718 -0.586 -0.367

2

0.317

0.525

0.804

-0.548 -0.492 -0.608], b=0.10259240865707397

Test MSE: 2.6259625

4 Train the model

6

 3
```

4

5

```
[ ]: train(1e-1, epochs=50)
```

5 Test the model after training

```
[20]: test_the_model()
```

```
Model: a=[-0.118 0.14 -0.012 0.111 -0.265

0.263

0.02

-0.381

0.261 -0.15

-0.215 0.088 -0.436], b=6.539222940915579e-09

Test MSE: 0.24517176
```

6 View TensorBoard

Note: You should select this cell and call to Interrupt the Kernel after finishing checking TensorBoard

```
[ ]: !tensorboard --logdir ./tmp/mylogs
```

1. Apply denormalization for the testing function
2. Apply Stochastic Gradient Descent for training:

7 Homework

- 1-batch
- mini-batch

3. Try other loss functions:

- Root Mean Squared Error,
- Mean Absolute Error,
- Combined RMSE+MAE+MSE

7

```
 1.1 Ref.

[ ]:

[ ]:

• https://realpython.com/pdf-python/
• https://pdfminersix.readthedocs.io/en/latest/tutorial/highlevel.html

8
```

[ ]:

1.1 Ref.

- https://realpython.com/pdf-python/
- https://pdfminersix.readthedocs.io/en/latest/tutorial/highlevel.html

[ ]: