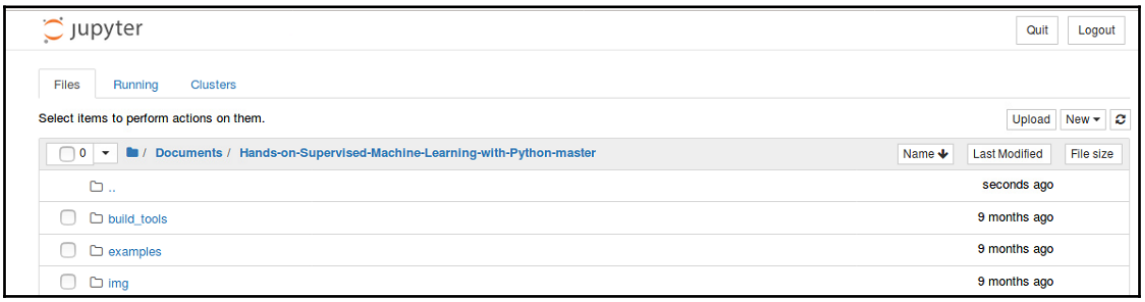
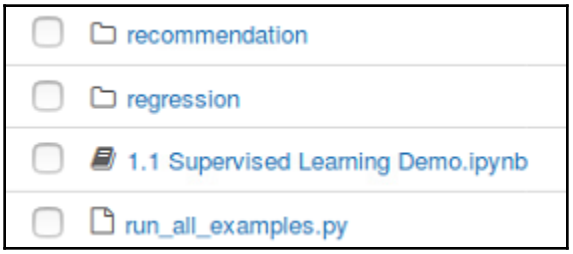
Chapter 1: First Step Towards Supervised Learning



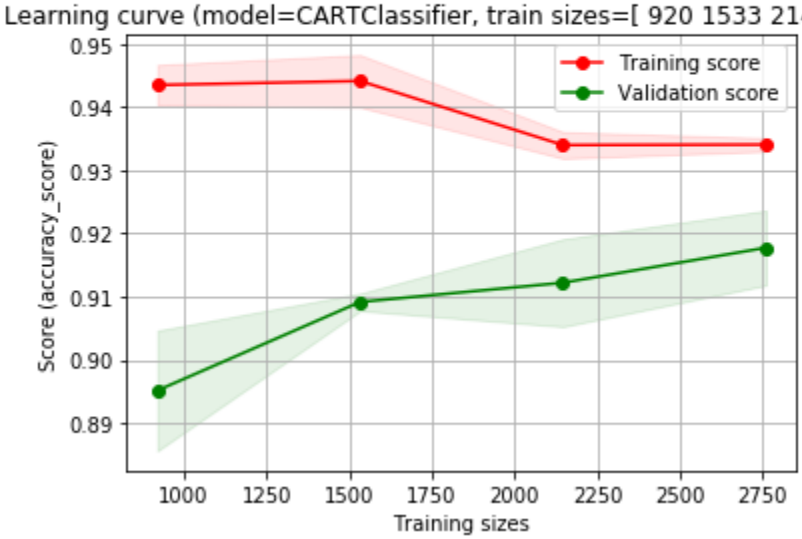


Out[2]:

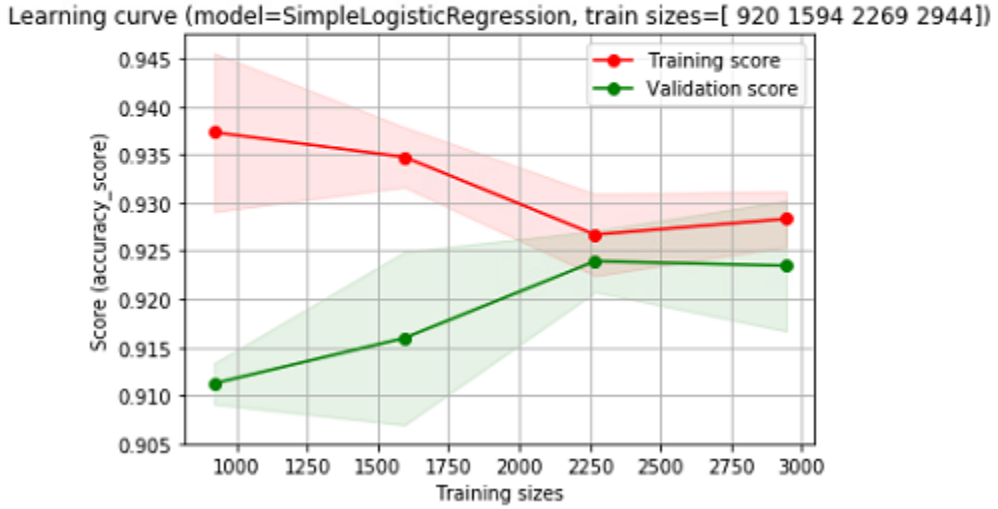
	word_freq_make	word_freq_address	word_freq_all	word_freq_3d	word_freq_our	word_freq_over	word_freq_remove	word_freq_internet
0	0.00	0.64	0.64	0.0	0.32	0.00	0.00	0.00
1	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07
2	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63
4	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63

5 rows × 57 columns

```
Completed fitting 3 folds for train size=920
Completed fitting 3 folds for train size=1533
Completed fitting 3 folds for train size=2146
Completed fitting 3 folds for train size=2760
```



```
Completed fitting 3 folds for train size=920
Completed fitting 3 folds for train size=1594
Completed fitting 3 folds for train size=2269
Completed fitting 3 folds for train size=2944
```



```
| SPAM E-MAIL DATABASE ATTRIBUTES (in .names format)
|
| 48 continuous real [0,100] attributes of type word_freq_WORD
| = percentage of words in the e-mail that match WORD,
| i.e. 100 * (number of times the WORD appears in the e-mail) /
| total number of words in e-mail.  A "word" in this case is any
| string of alphanumeric characters bounded by non-alphanumeric
| characters or end-of-string.
|
| 6 continuous real [0,100] attributes of type char_freq_CHAR
| = percentage of characters in the e-mail that match CHAR,
| i.e. 100 * (number of CHAR occurences) / total characters in e-mail
|
| 1 continuous real [1,...] attribute of type capital_run_length_average
| = average length of uninterrupted sequences of capital letters
|
| 1 continuous integer [1,...] attribute of type capital_run_length_longest
| = length of longest uninterrupted sequence of capital letters
|
| 1 continuous integer [1,...] attribute of type capital_run_length_total
| = sum of length of uninterrupted sequences of capital letters
| = total number of capital letters in the e-mail
|
| 1 nominal {0,1} class attribute of type spam
| = denotes whether the e-mail was considered spam (1) or not (0),
```

```
Hey George,

Make sure to be careful when checking your HP email. There has been a
phishing attempt recently advertising a credit report. This is a known scam,
and should be ignored. Please feel free to let me know if you have any
questions or concerns, but the IT guys told me to warn everyone.
```

In [2]:

```
import pandas as pd

names = ["word_freq_make", "word_freq_address", "word_freq_all",
         "word_freq_3d", "word_freq_our", "word_freq_over",
         "word_freq_remove", "word_freq_internet", "word_freq_order",
         "word_freq_mail", "word_freq_receive", "word_freq_will",
         "word_freq_people", "word_freq_report", "word_freq_addresses",
         "word_freq_free", "word_freq_business", "word_freq_email",
         "word_freq_you", "word_freq_credit", "word_freq_your",
         "word_freq_font", "word_freq_000", "word_freq_money",
         "word_freq_hp", "word_freq_hpl", "word_freq_george",
         "word_freq_650", "word_freq_lab", "word_freq_labs",
         "word_freq_telnet", "word_freq_857", "word_freq_data",
         "word_freq_415", "word_freq_85", "word_freq_technology",
         "word_freq_1999", "word_freq_parts", "word_freq_pm",
         "word_freq_direct", "word_freq_cs", "word_freq_meeting",
         "word_freq_original", "word_freq_project", "word_freq_re",
         "word_freq_edu", "word_freq_table", "word_freq_conference",
         "char_freq_;", "char_freq_(", "char_freq_[", "char_freq_!",
         "char_freq_$", "char_freq_#", "capital_run_length_average",
         "capital_run_length_longest", "capital_run_length_total",
         "is_spam"]

df = pd.read_csv(os.path.join("data", "spam.csv"), header=None, names=names)

# pop off the target
y = df.pop("is_spam")
df.head()
```

```
Spam email:
[  0.          0.          0.          0.          0.          0.          0.
   0.          0.          0.          2.85714286  0.          0.          0.
   0.          2.85714286  2.85714286  2.85714286  5.71428571
   5.71428571  5.71428571  0.          0.          2.85714286  0.          0.
   0.          0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   2.85714286  0.          0.          0.          0.          0.          0.
   0.          0.          0.61728395  0.          0.          2.4691358
   0.61728395  0.          7.2         17.         36.        ]

Real email:
[  1.81818182  0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.          0.
   0.          1.81818182  0.          0.          1.81818182
   1.81818182  1.81818182  0.          0.          0.          1.81818182
   0.          0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.          0.
   0.          1.25        2.          10.        ]
```

```
Decision tree predictions:
Spam email prediction: 'SPAM!'
Real email prediction: 'Not spam'

Logistic regression predictions:
Spam email prediction: 'SPAM!'
Real email prediction: 'Not spam'
```

```
test@test-Veriton-Series:~/Downloads/Hands-on-Supervised-Machine-Learning-with-Python-master$ conda env create -f environment.yml

CondaValueError: prefix already exists: /home/test/anaconda3/envs/packt-sml

test@test-Veriton-Series:~/Downloads/Hands-on-Supervised-Machine-Learning-with-Python-master$ source activate packt-sml
(packt-sml) test@test-Veriton-Series:~/Downloads/Hands-on-Supervised-Machine-Learning-with-Python-master$
```

```
(packt-sml) test@test-Veriton-Series:~/Documents/Hands-on-Supervised-Machine-Learning-with-Python-master$ cat setup.py
# -*- coding: utf-8 -*-

from __future__ import absolute_import

import sys
import setuptools

with open("packtml/VERSION", 'r') as vsn:
    VERSION = vsn.read().strip()

# Permitted args: "install" only, basically.
UNSUPPORTED_COMMANDS = {  # this is a set literal, not a dict
    'develop', 'release', 'bdist_egg', 'bdist_rpm',
    'bdist_wininst', 'install_egg_info', 'build_sphinx',
    'egg_info', 'easy_install', 'upload', 'bdist_wheel',
    '--single-version-externally-managed', 'test', 'build_ext'
}

intersect = UNSUPPORTED_COMMANDS.intersection(set(sys.argv))
if intersect:
    msg = "The following arguments are unsupported: %s. " \
          "To install, please use `python setup.py install`." \
          % str(list(intersect))

    # if "test" is in the arguments, make sure the user knows how to test.
    if "test" in intersect:
        msg += " To test, make sure pytest is installed, and after " \
               "installation run `pytest packtml`"

    raise ValueError(msg)

# get requirements
with open("requirements.txt") as req:
    REQUIREMENTS = req.read().strip().split("\n")

py_version_tag = '-%s.%s'.format(sys.version_info[:2])
setuptools.setup(name="packtml",
                 description="Hands-on Supervised Learning - teach a machine "
                             "to think for itself!",
                 author="Taylor G Smith",
                 author_email="taylor.smith@alkaline-ml.com",
                 packages=['packtml',
                           'packtml/clustering',
                           'packtml/decision_tree',
                           'packtml/metrics',
                           'packtml/neural_net',
                           'packtml/recommendation',
                           'packtml/regression',
                           'packtml/utils'],
                 zip_safe=False,
                 include_package_data=True,
```

```
(packt-sml) test@test-Veriton-Series:~/Desktop/Code/Hands-on-Supervised-Machine-Learning-with-Python-master$ python setup.py install
running install
running bdist_egg
running egg_info
creating packtml.egg-info
writing packtml.egg-info/PKG-INFO
writing dependency_links to packtml.egg-info/dependency_links.txt
writing requirements to packtml.egg-info/requires.txt
writing top-level names to packtml.egg-info/top_level.txt
writing manifest file 'packtml.egg-info/SOURCES.txt'
reading manifest file 'packtml.egg-info/SOURCES.txt'
reading manifest template 'MANIFEST.in'
warning: manifest_maker: MANIFEST.in, line 1: unknown action 'recursive'

writing manifest file 'packtml.egg-info/SOURCES.txt'
installing library code to build/bdist.linux-x86_64/egg
running install_lib
running build_py
creating build
creating build/lib
creating build/lib/packtml
copying packtml/__init__.py -> build/lib/packtml
copying packtml/base.py -> build/lib/packtml
creating build/lib/packtml/clustering
copying packtml/clustering/knn.py -> build/lib/packtml/clustering
copying packtml/clustering/__init__.py -> build/lib/packtml/clustering
creating build/lib/packtml/decision_tree
copying packtml/decision_tree/cart.py -> build/lib/packtml/decision_tree
copying packtml/decision_tree/__init__.py -> build/lib/packtml/decision_tree
copying packtml/decision_tree/metrics.py -> build/lib/packtml/decision_tree
creating build/lib/packtml/metrics
copying packtml/metrics/__init__.py -> build/lib/packtml/metrics
copying packtml/metrics/ranking.py -> build/lib/packtml/metrics
creating build/lib/packtml/neural_net
copying packtml/neural_net/__init__.py -> build/lib/packtml/neural_net
copying packtml/neural_net/base.py -> build/lib/packtml/neural_net
copying packtml/neural_net/transfer.py -> build/lib/packtml/neural_net
copying packtml/neural_net/mlp.py -> build/lib/packtml/neural_net
creating build/lib/packtml/recommendation
copying packtml/recommendation/__init__.py -> build/lib/packtml/recommendation
copying packtml/recommendation/base.py -> build/lib/packtml/recommendation
copying packtml/recommendation/als.py -> build/lib/packtml/recommendation
copying packtml/recommendation/itemitem.py -> build/lib/packtml/recommendation
copying packtml/recommendation/data.py -> build/lib/packtml/recommendation
creating build/lib/packtml/regression
copying packtml/regression/__init__.py -> build/lib/packtml/regression
copying packtml/regression/simple_logistic.py -> build/lib/packtml/regression
copying packtml/regression/simple_regression.py -> build/lib/packtml/regression
creating build/lib/packtml/utils
copying packtml/utils/linalg.py -> build/lib/packtml/utils
copying packtml/utils/extmath.py -> build/lib/packtml/utils
copying packtml/utils/__init__.py -> build/lib/packtml/utils
copying packtml/utils/validation.py -> build/lib/packtml/utils
copying packtml/utils/plotting.py -> build/lib/packtml/utils
copying packtml/VERSION -> build/lib/packtml
creating build/bdist.linux-x86_64
creating build/bdist.linux-x86_64/egg
```

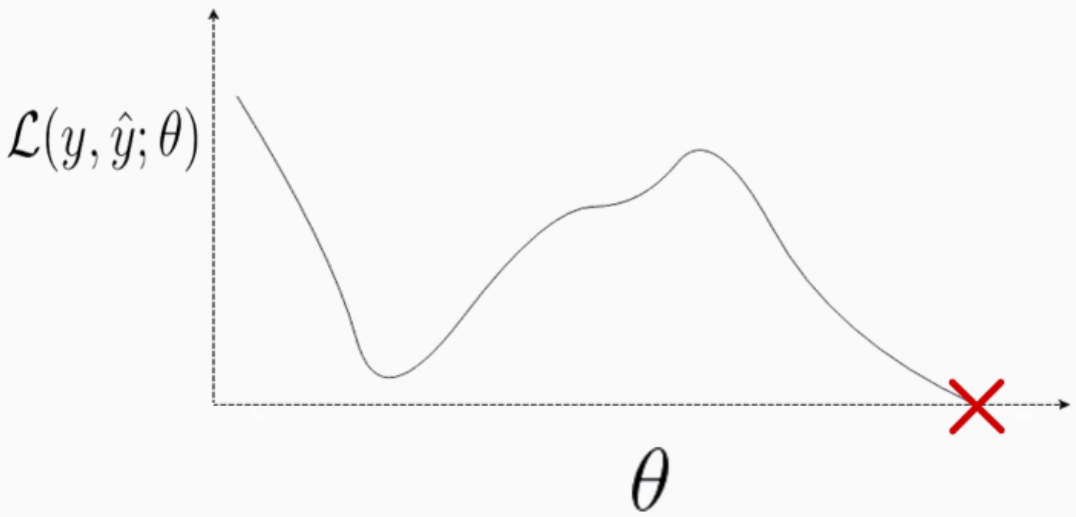
ML: a simple example

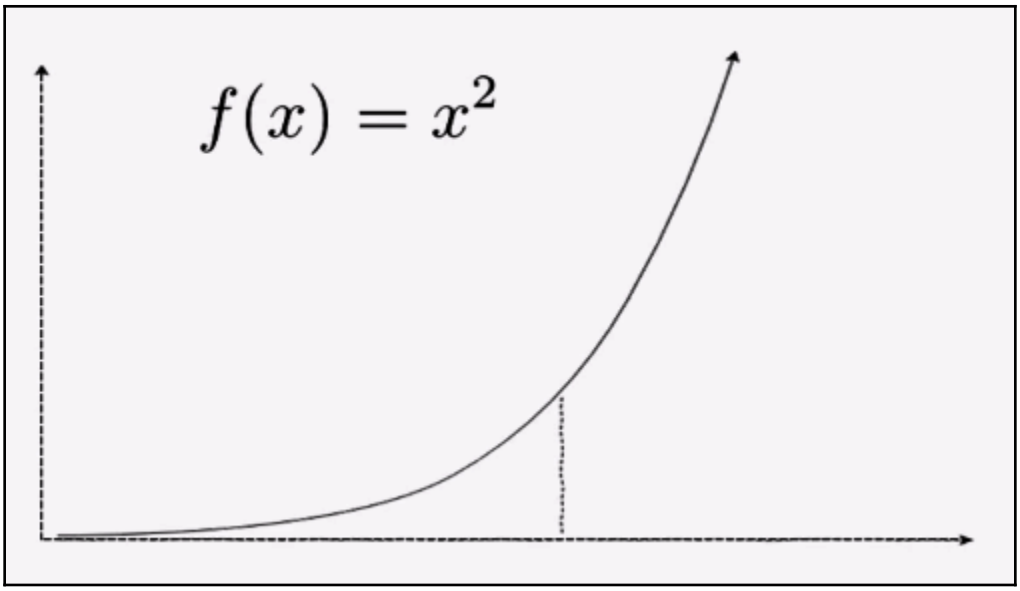
```
from sklearn.linear_model import LogisticRegression

def learn_life_lesson(X, y):
    """Learn a lesson and apply it in a future situation"""
    model = LogisticRegression().fit(X, y)
    return (lambda x: model.predict(x))

# learn a lesson and THEN make a decision
educated_decision = learn_life_lesson(X, y)(X)
educated_decision
```

```
array([1, 1, 0, 0, 0, 1, 1, 0, 1, 0])
```

Learn our function *f*

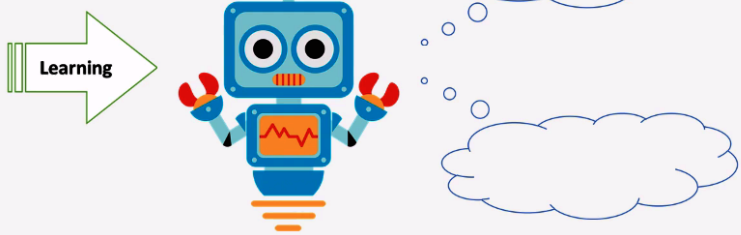
Apply *f* to the data to produce predictions



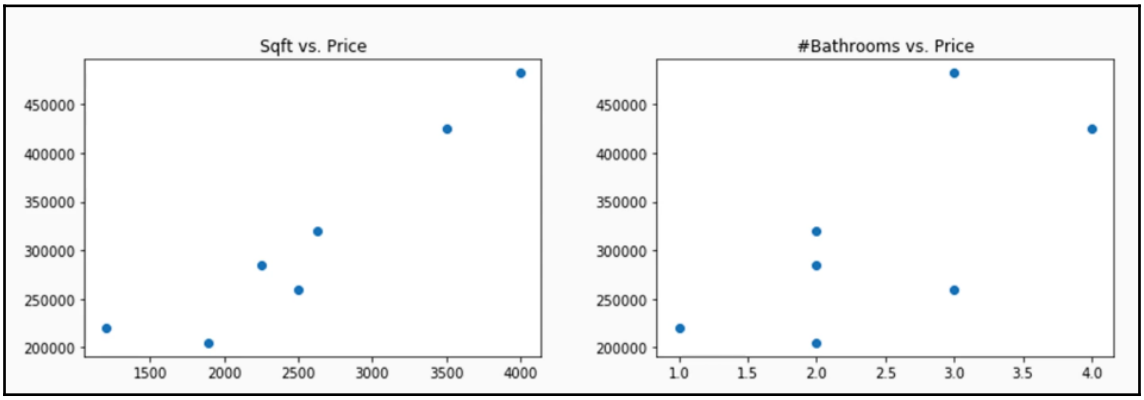


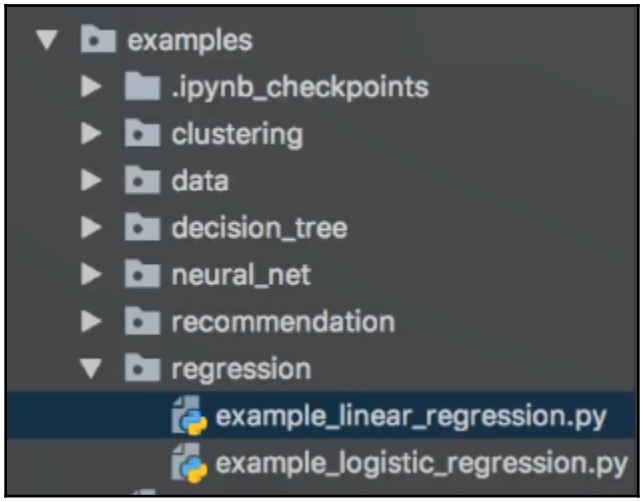
- In-sample: Assesses a model's ability to memorize (bad)
- Out-of-sample: Assesses a model's ability to generalize (good)

$$X = \begin{bmatrix} 21 & 3 \\ 4 & 2 \\ 37 & 2 \end{bmatrix}, y = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

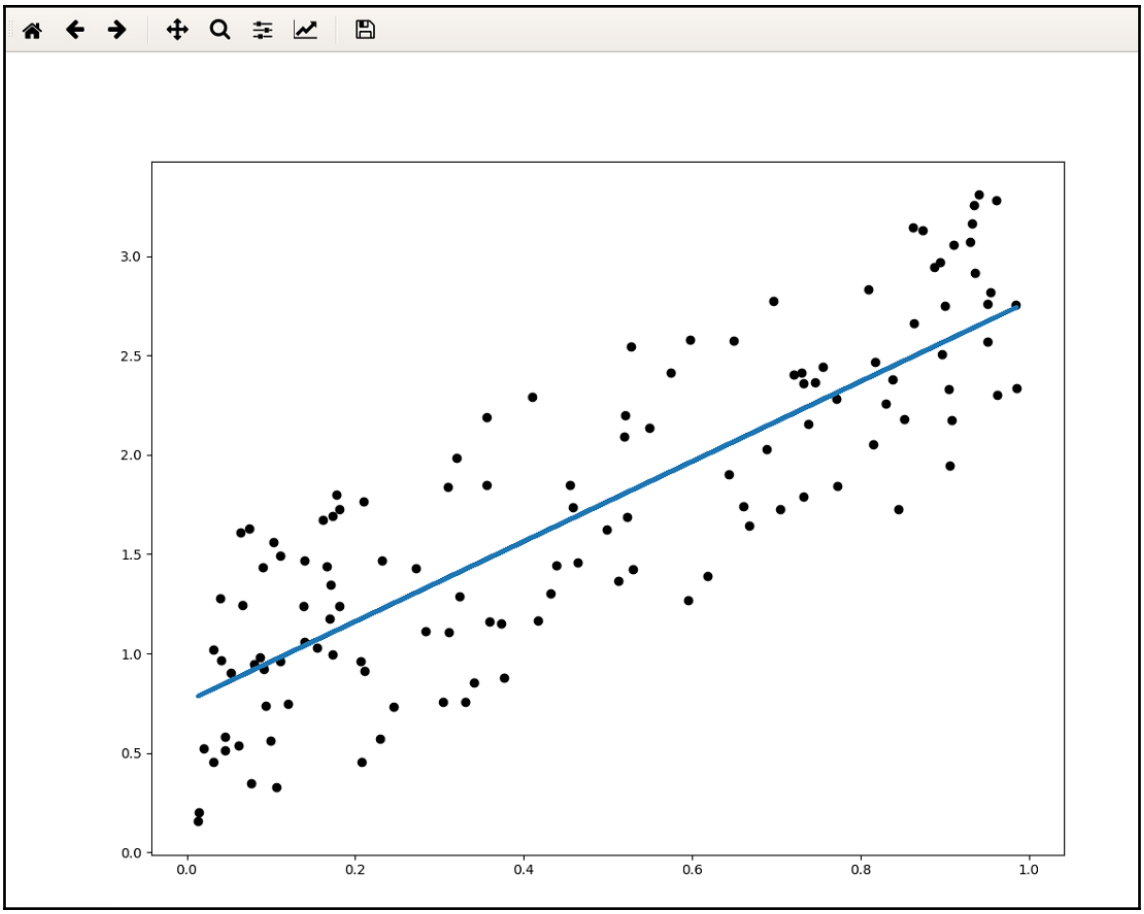


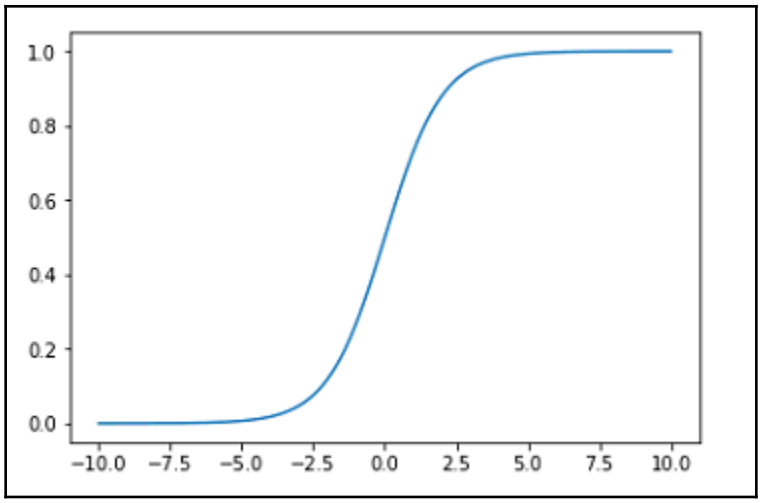
Chapter 2: Implementing Parametric Models



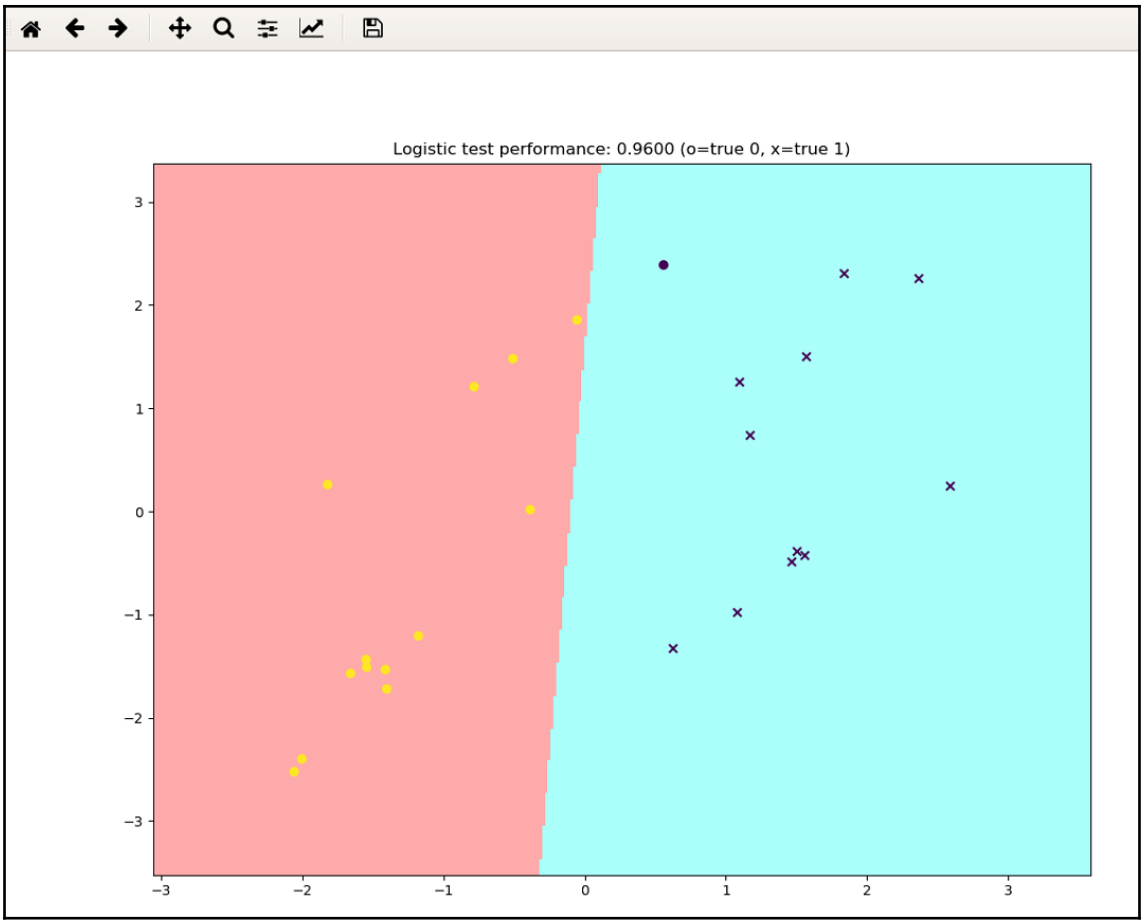


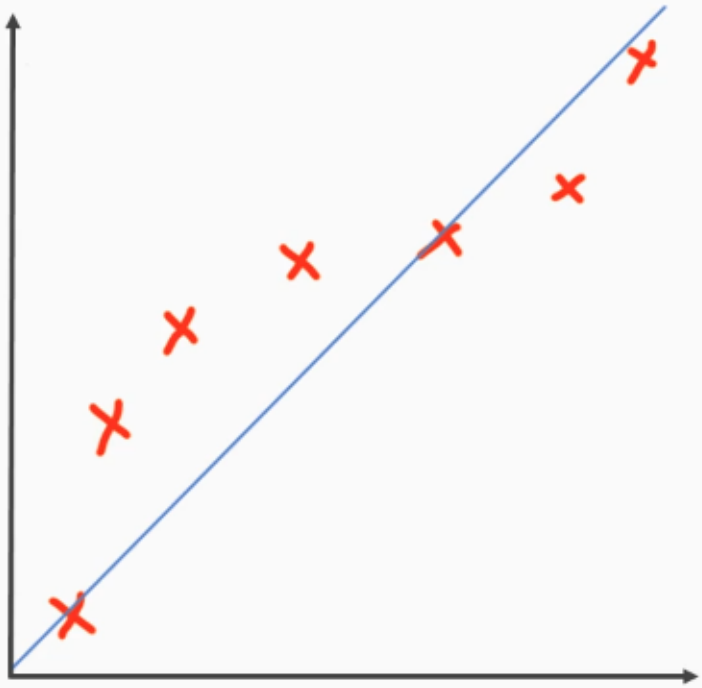

```
test@test-Veriton-Series:~/Documents/Hands-on-Supervised-Machine-Learning-with-Python-master$ source activate packt-sml
(packt-sml) test@test-Veriton-Series:~/Documents/Hands-on-Supervised-Machine-Learning-with-Python-master$ python examples/regression
example_linear_regression.p
y
/home/test/anaconda3/envs/packt-sml/lib/python3.6/site-packages/packtml-1.0.3-py3.6.egg/packtml/regression/simple_regression.py:73: F
utureWarning: `rcond` parameter will change to the default of machine precision times ``max(M, N)`` where M and N are the input matri
x dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old, explicitly pass `rcond=-1`.
  theta, _, rank, singular_values = lstsq(X, y)
Test sum of residuals: -0.000
```


3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0



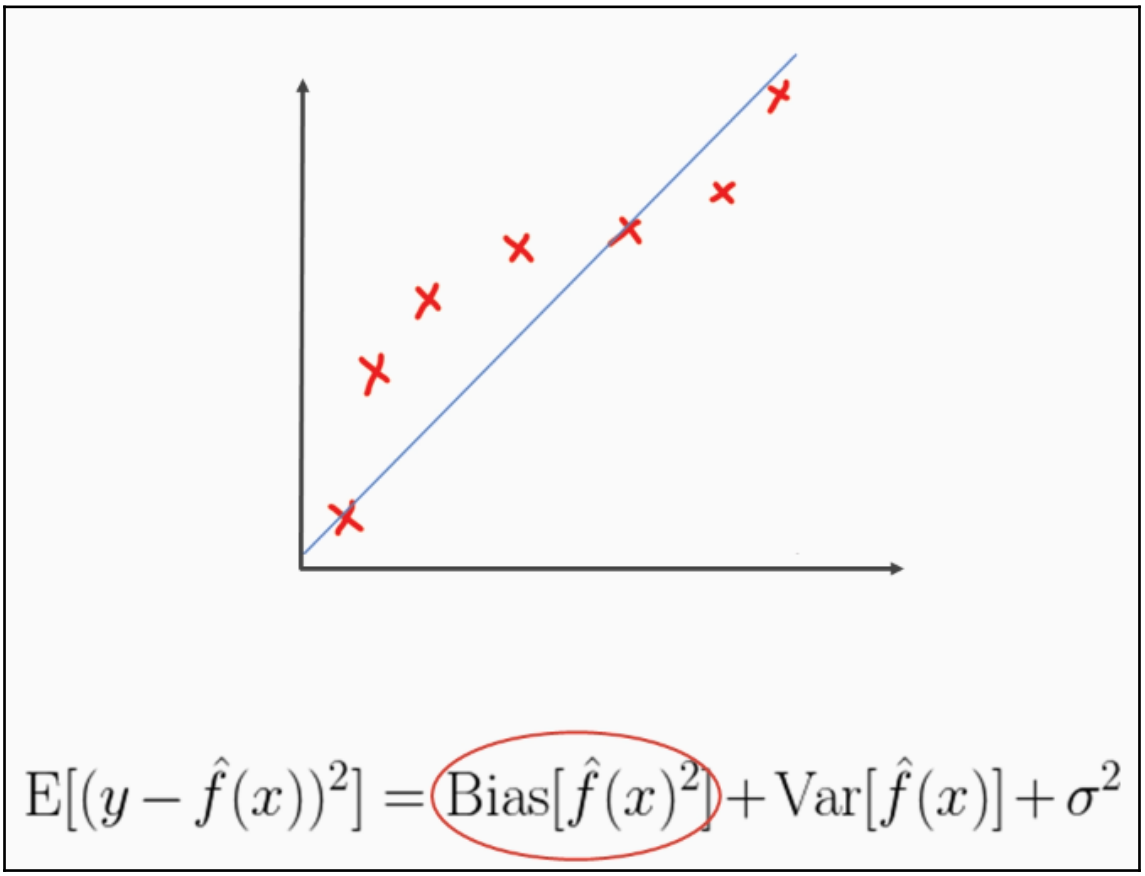

```
(packt-sml) test@test-Veriton-Series:~/Documents/Hands-on-Supervised-Machine-Learning-with-Python-master$ python examples/regression/
example_logistic_regression.py
Test accuracy: 0.960
/home/test/anaconda3/envs/packt-sml/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver w
ill be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
Sklearn test accuracy: 1.000
```


Logistic test performance: 0.9600 (o=true 0, x=true 1)
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

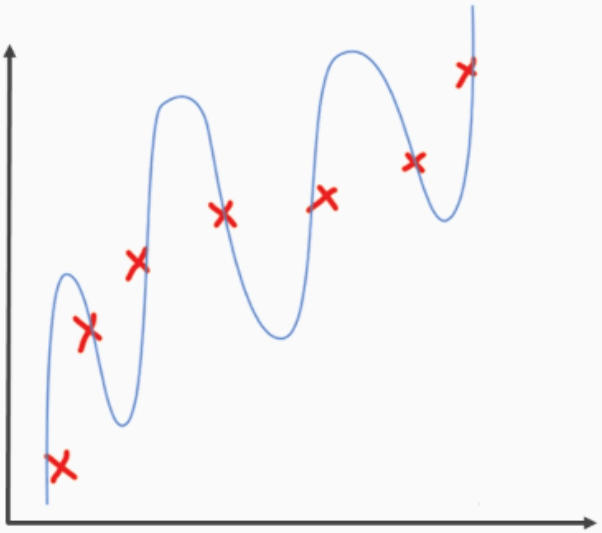


$$\mathrm{E}[(y - \hat{f}(x))^2] = \mathrm{Bias}[\hat{f}(x)^2] + \mathrm{Var}[\hat{f}(x)] + \sigma^2$$

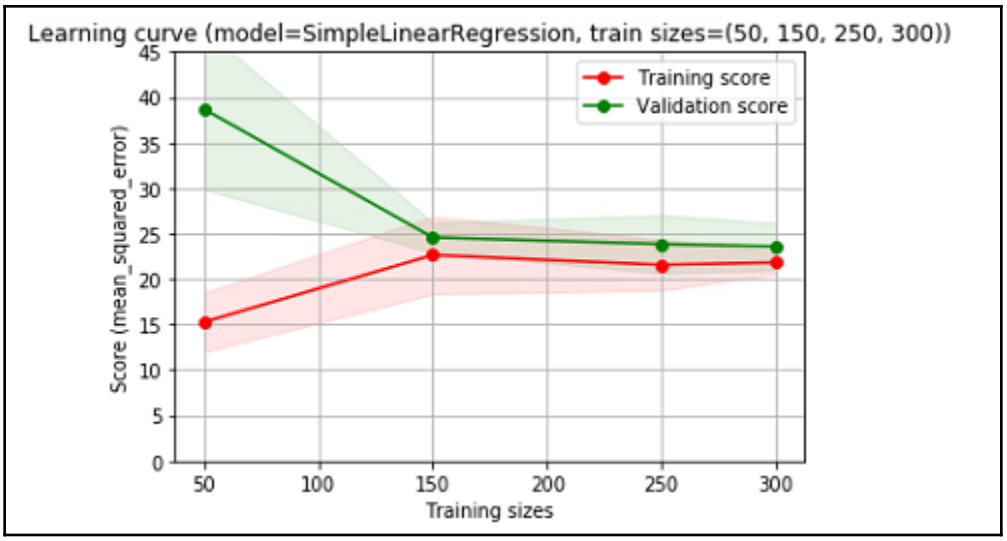
Chapter 3: Working with Non-Parametric Models

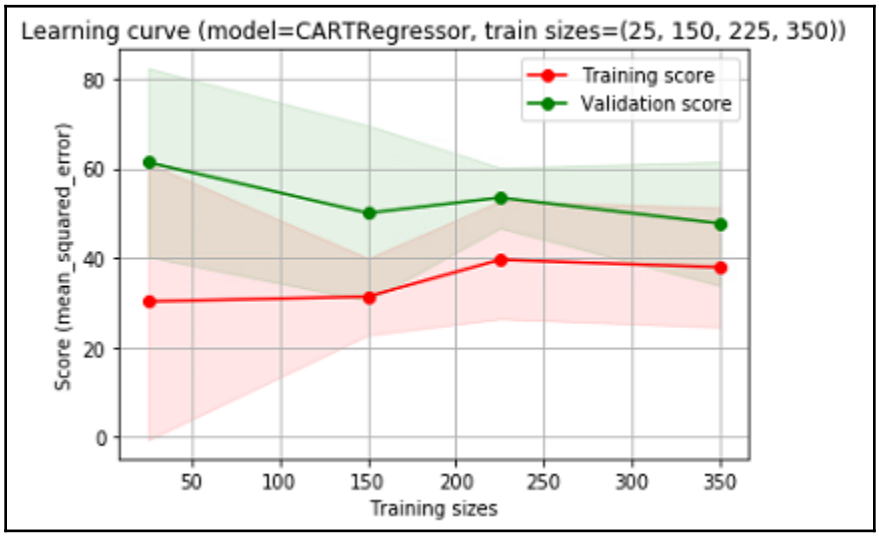


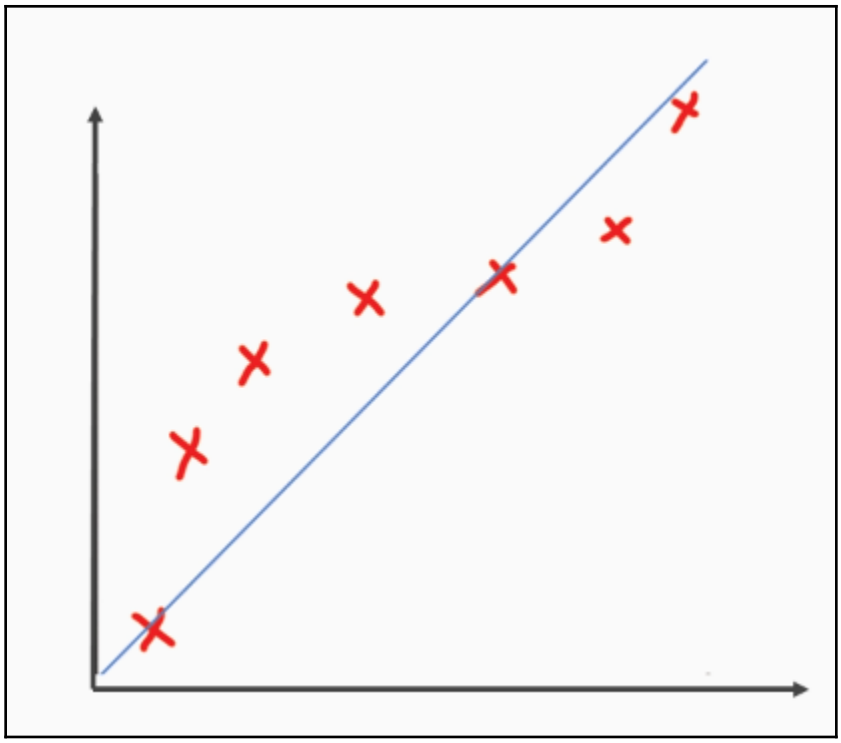
$$\mathrm{E}[(y-\hat{f}(x))^2] = \mathrm{Bias}[\hat{f}(x)^2] + \mathrm{Var}[\hat{f}(x)] + \sigma^2$$

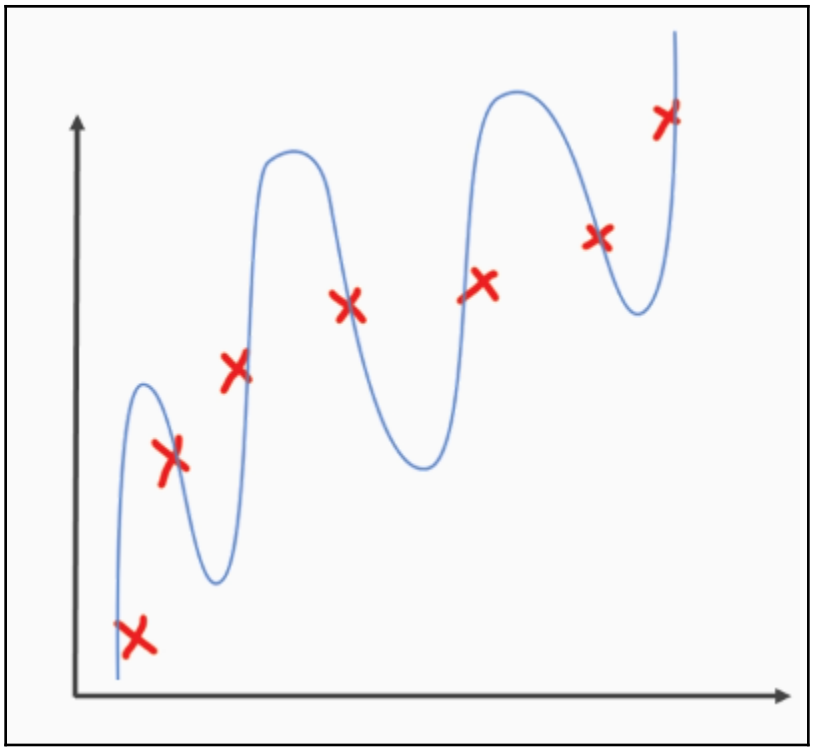


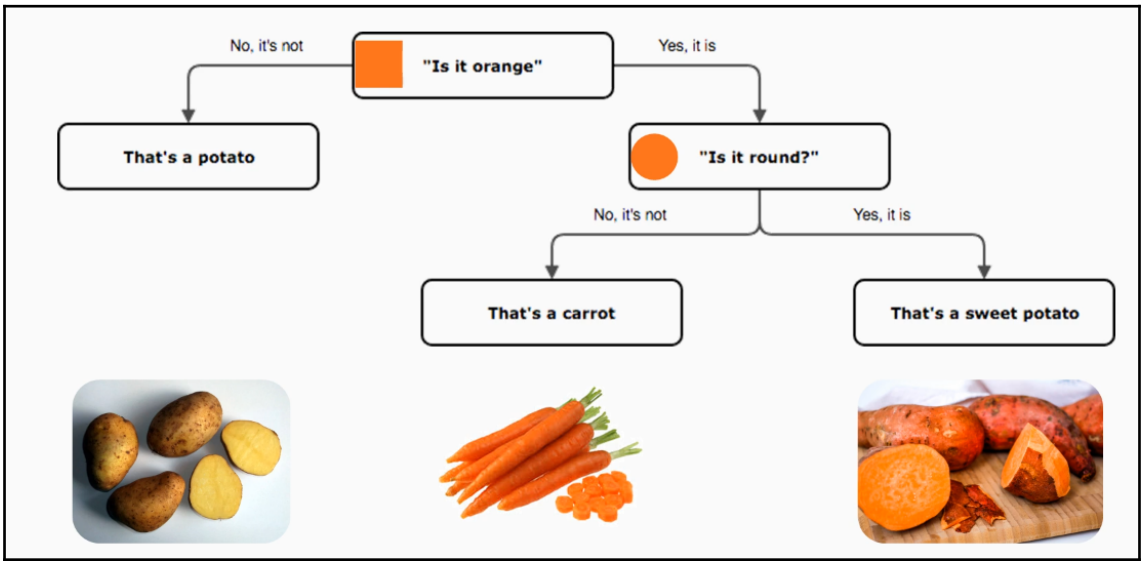
$$\mathrm{E}[(y - \hat{f}(x))^2] = \mathrm{Bias}[\hat{f}(x)^2] + \mathrm{Var}[\hat{f}(x)] + \sigma^2$$


Learning curve (model=SimpleLinearRegression, train sizes=(50, 150, 250, 300))
Training score
Validation score
Score (mean_squared_error)
Training sizes
0
5
10
15
20
25
30
35
40
45
50
100
150
200
250
300

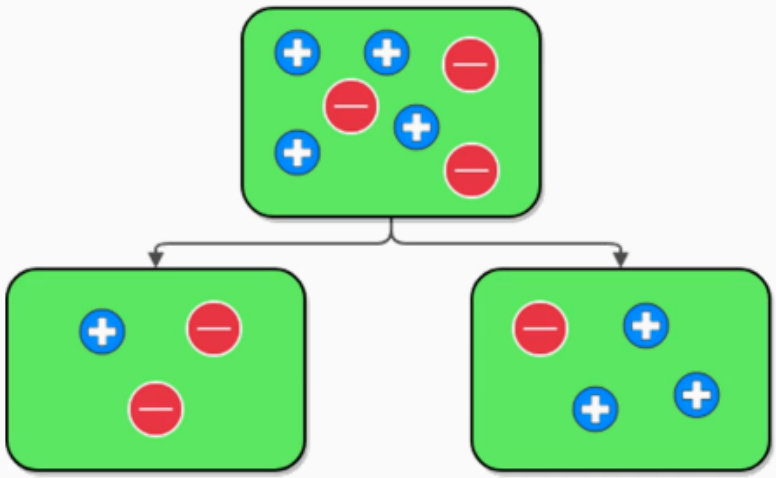

Learning curve (model=CARTRegressor, train sizes=(25, 150, 225, 350))
Training score
Validation score
Score (mean_squared_error)
Training sizes
0
20
40
60
80
50
100
150
200
250
300
350






No, it's not
"Is it orange"
Yes, it is
That's a potato
"Is it round?"
No, it's not
Yes, it is
That's a carrot
That's a sweet potato

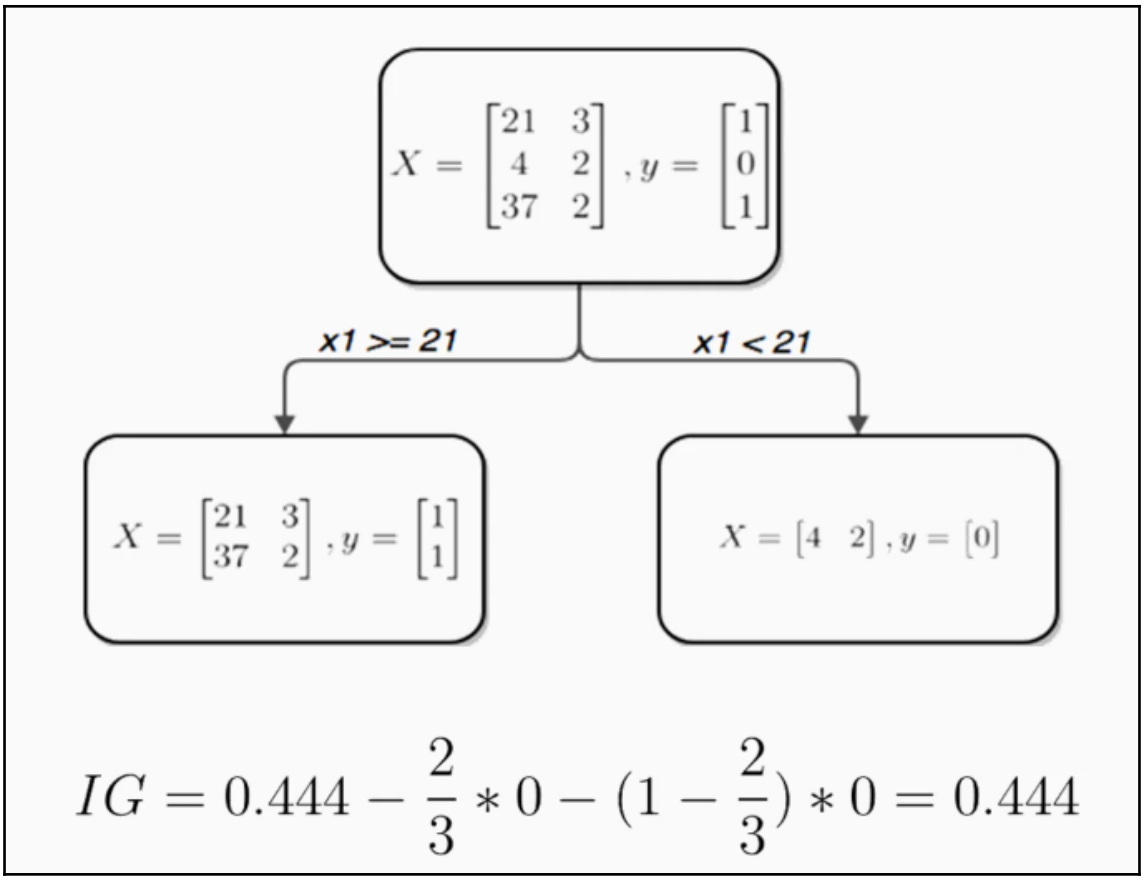
$$I_G = 1 - (\frac{4}{7}^2 + \frac{3}{7}^2) \approx 0.49$$



$$I_G = 1 - (\frac{1}{3}^2 + \frac{2}{3}^2) \approx 0.444 \qquad I_G = 1 - (\frac{1}{4}^2 + \frac{3}{4}^2) = 0.375$$

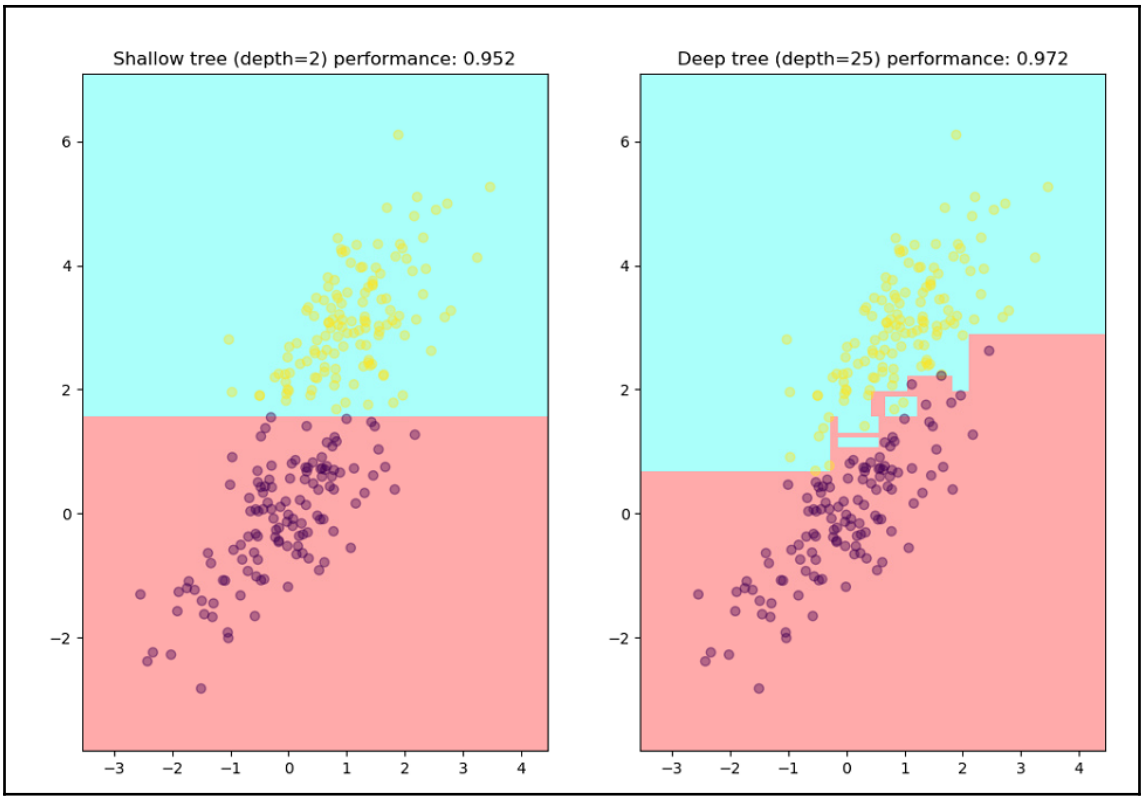
$$IG = 0.49 - \frac{3}{7} * 0.444 - (1 - \frac{3}{7}) * 0.375 \approx 0.085$$

```
(packt-sml) test@test-Veriton-Series:~/Documents/Hands-on-Supervised-Machine-Learning-with-Python-master/examples/decision_tree$ pyth
on example_information_gain.py
Initial gini impurity: 0.4898
Information gain from the split we created: 0.0850
```

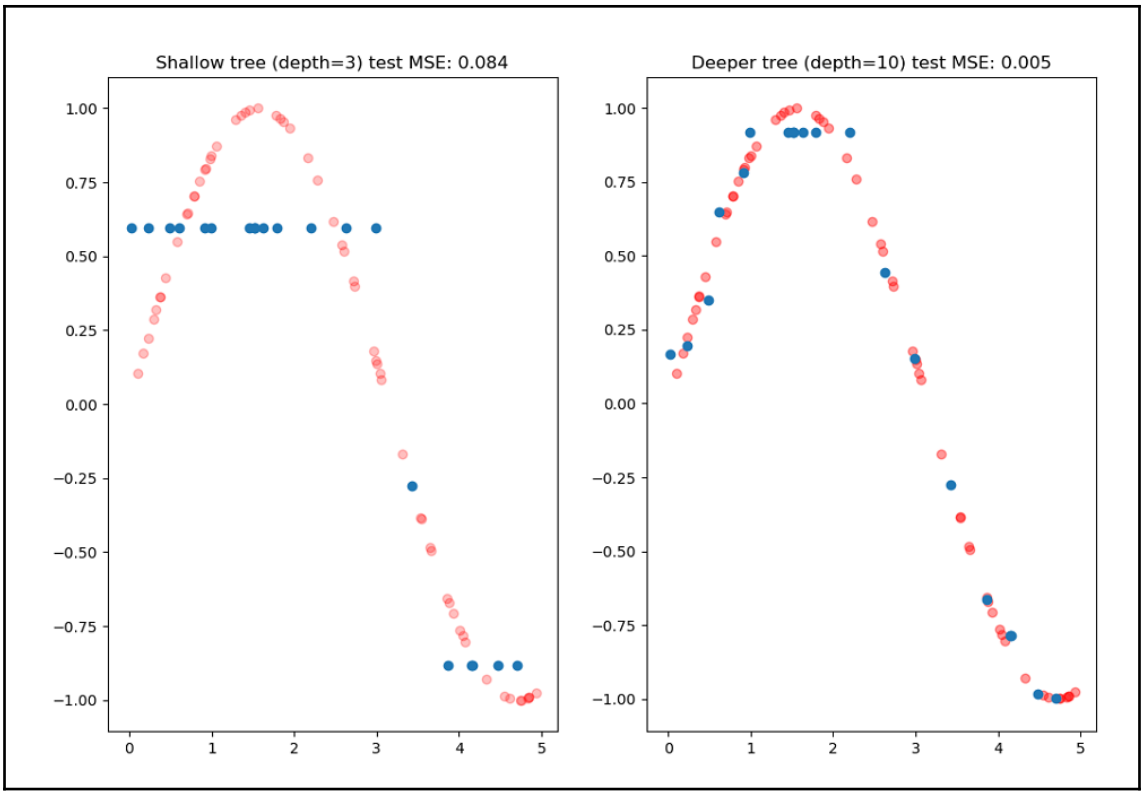


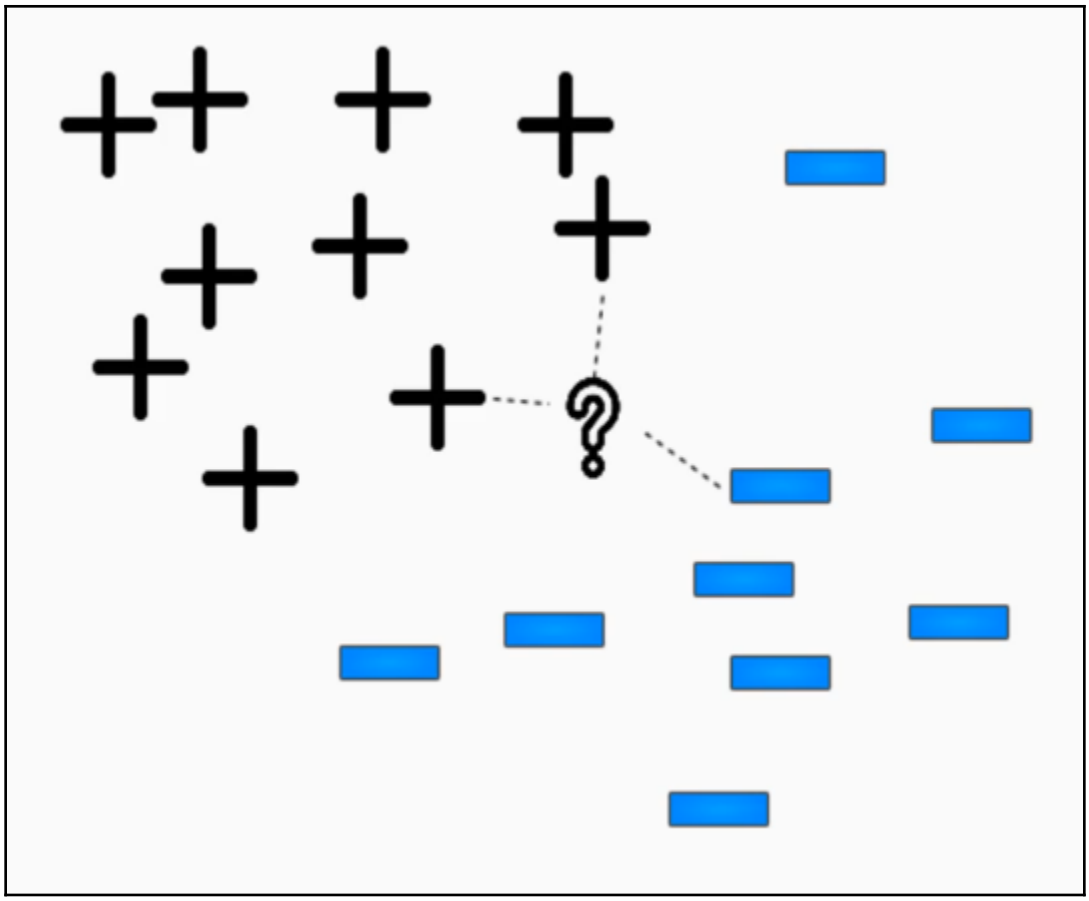

```
(packt-sml) test@test-Veriton-Series:~/Documents/Hands-on-Supervised-Machine-Learning-with-Python-master/examples/decision_tree$ pyth
on example_classification_split.py
Best feature=0, best value=21, information gain: 0.444
```

```
(packt-sml) test@test-Veriton-Series:~/Documents/Hands-on-Supervised-Machine-Learning-with-Python-master$ python examples/decision_tr
ee/example_classification_decision_tree.py
Test accuracy (depth=2): 0.952
Test accuracy (depth=25): 0.972
```

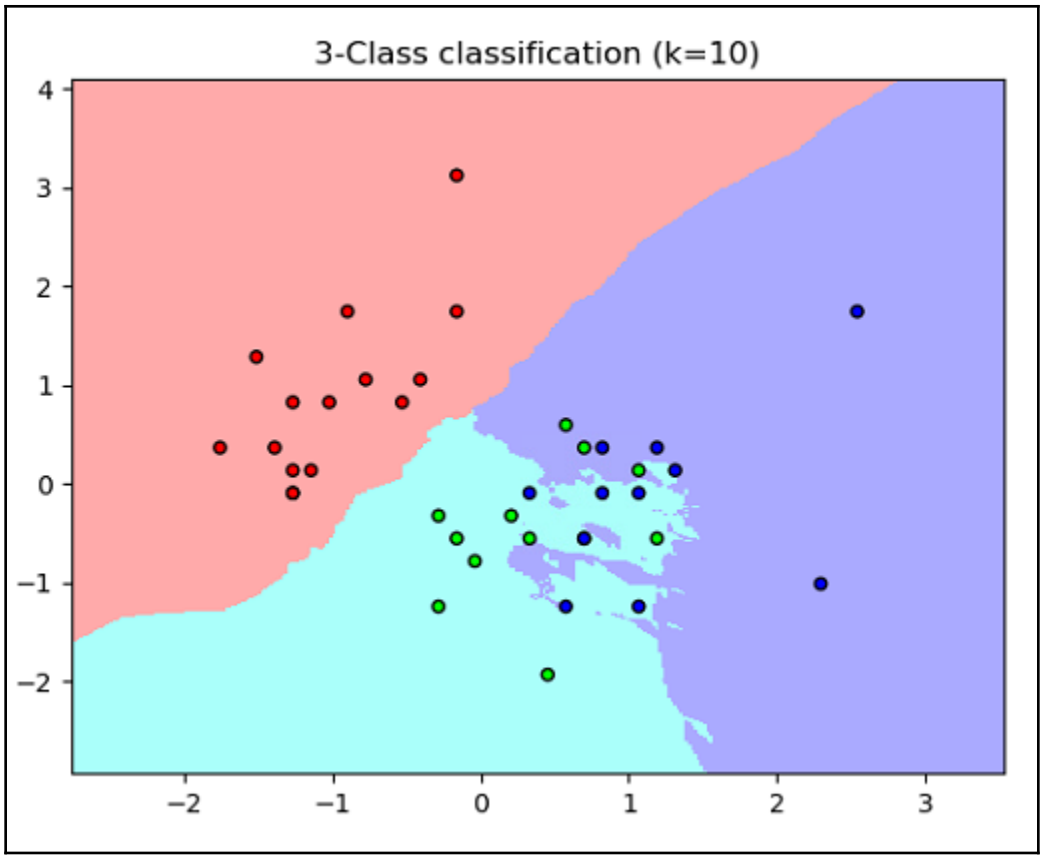



```
(packt-sml) test@test-Veriton-Series:~/Documents/Hands-on-Supervised-Machine-Learning-with-Python-master$ python examples/decision_tr
ee/example_regression_decision_tree.py
Test MSE (depth=3): 0.084
Test MSE (depth=10): 0.005
```

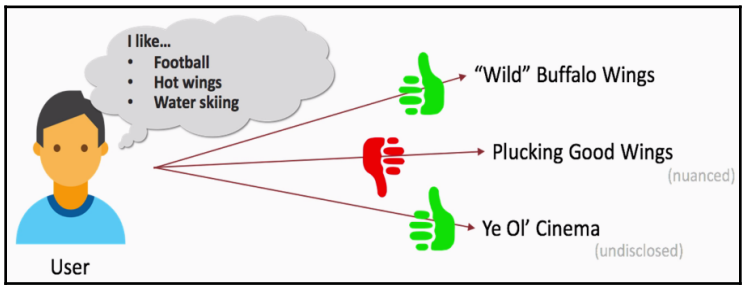

Shallow tree (depth=3) test MSE: 0.084
Deeper tree (depth=10) test MSE: 0.005
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
1
2
3
4
5



```
(packt-sml) test@test-Veriton-Series:~/Downloads/Hands-on-Supervised-Machine-Learning-with-Python-master/examples/clustering$ python
example_knn_classifier.py
Test accuracy: 0.711
```

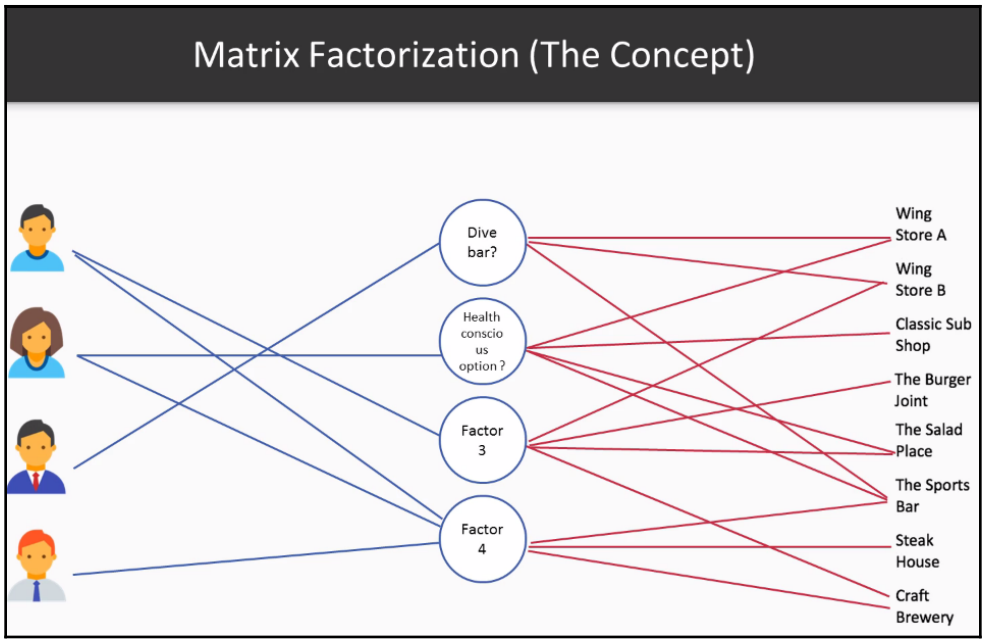

3-Class classification (k=10)
4
3
2
1
0
−1
−2
−2
−1
0
1
2
3

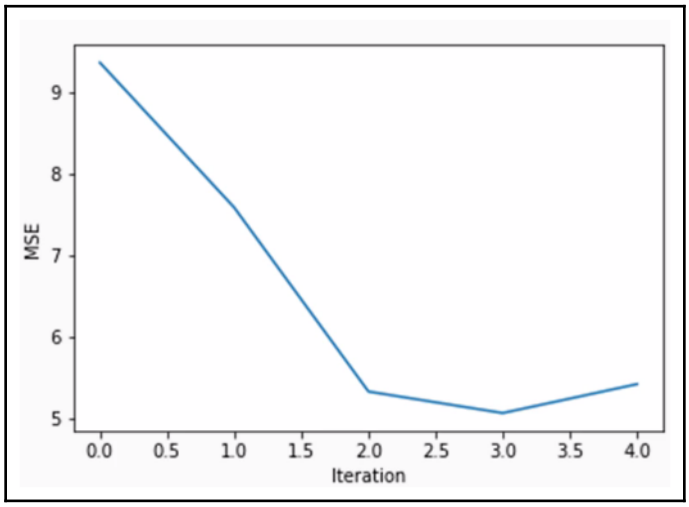
Chapter 4: Advanced Topics in Supervised Machine Learning



	Wing Store A	Wing Store B	Classic Sub Shop	The Burger Joint	The Salad Place	The Sports Bar	Steak House	Craft Brewery
	5.0	1.0			2.5	4.5		
			3.5	2.0	3.0			
	1.5				4.0		4.5	4.0
	?	1.0	?	?	1.0	?	?	5.0

```
(packt-sml) test@test-Veriton-Series:~/Downloads/Hands-on-Supervised-Machine-Learning-with-Python-master/examples/recommendation$ pyt
hon example_item_item_recommender.py
User 0's top 3 rated movies are: ['Ghost Busters', 'The Goonies', 'Pulp Fiction']
User 0's top 3 recommended movies are: ['Big Trouble in Little China', 'A Clockwork Orange', 'The Rocky Horror Picture Show']
Mean average precision: 0.667
(packt-sml) test@test-Veriton-Series:~/Downloads/Hands-on-Supervised-Machine-Learning-with-Python-master/examples/recommendation$
```


Matrix Factorization (The Concept)
Dive bar?
Health conscio us option ?
Factor 3
Factor 4
Wing Store A
Wing Store B
Classic Sub Shop
The Burger Joint
The Salad Place
The Sports Bar
Steak House
Craft Brewery


MSE
Iteration
9
8
7
6
5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

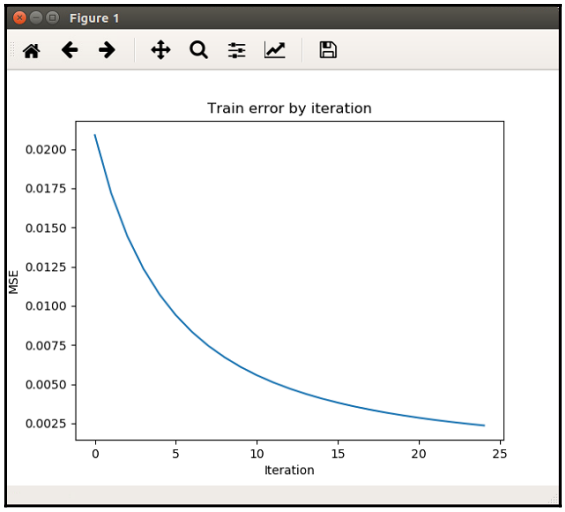
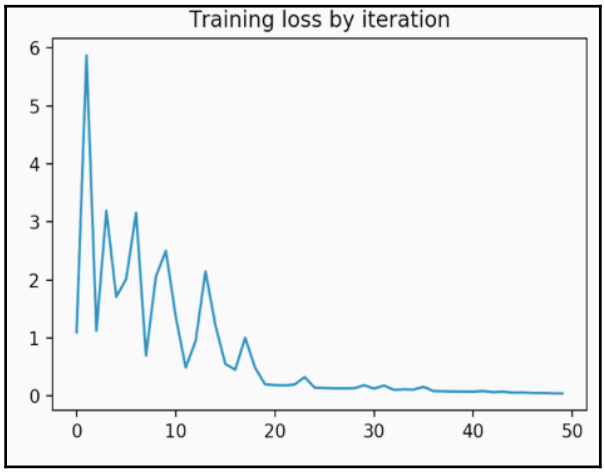
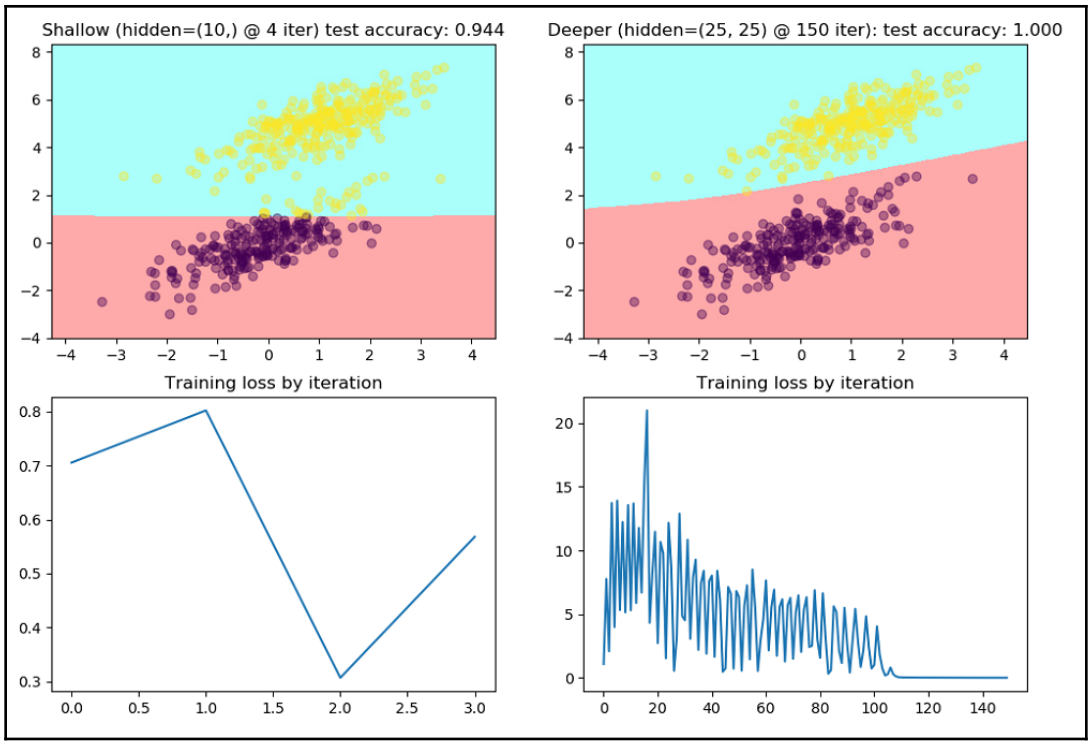
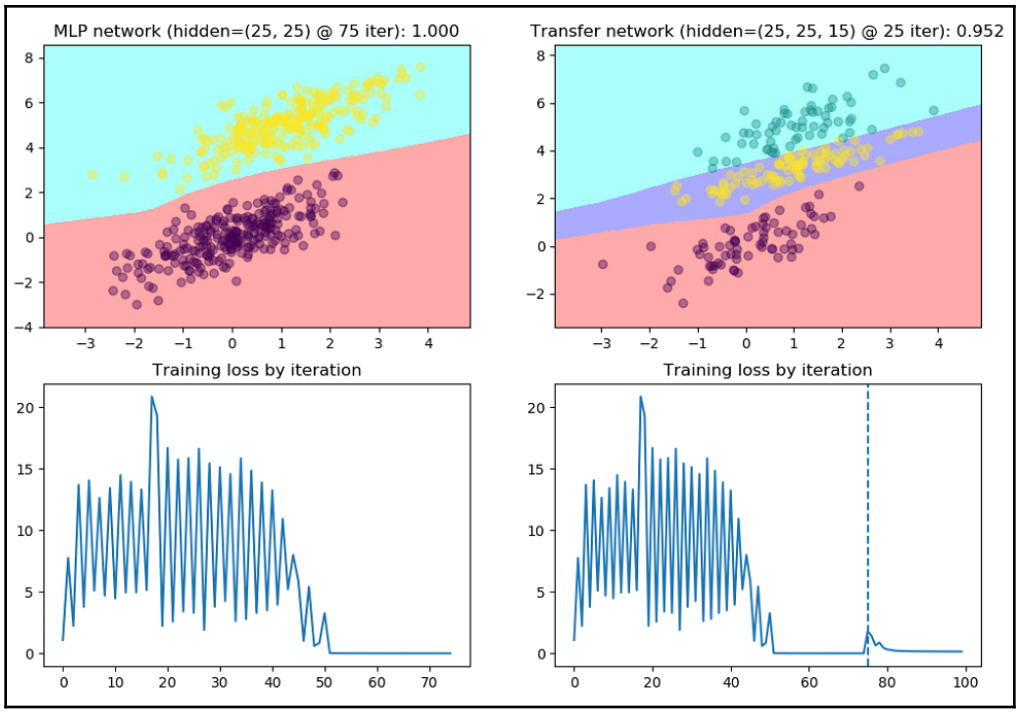


Figure 1
Train error by iteration
MSE
0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0
5
10
15
20
25
Iteration


Training loss by iteration
6
5
4
3
2
1
0
0
10
20
30
40
50


Shallow (hidden=(10,) @ 4 iter) test accuracy: 0.944
Deeper (hidden=(25, 25) @ 150 iter): test accuracy: 1.000
Training loss by iteration
Training loss by iteration


MLP network (hidden=(25, 25) @ 75 iter): 1.000
Transfer network (hidden=(25, 25, 15) @ 25 iter): 0.952
Training loss by iteration
Training loss by iteration