

Table of Contents

Python Machine Learning
2nd Ed.

[Python Machine Learning Second Edition](#)

[Credits](#)

[About the Authors](#)

[About the Reviewers](#)

[www.PacktPub.com](#)

[eBooks, discount offers, and more](#)

[Why subscribe?](#)

[Customer Feedback](#)

[Preface](#)

[What this book covers](#)

[What you need for this book](#)

[Who this book is for](#)

[Conventions](#)

[Reader feedback](#)

[Customer support](#)

[Downloading the example code](#)

[Downloading the color images of this book](#)

[Errata](#)

[Piracy](#)

[Questions](#)

[1. Giving Computers the Ability to Learn from Data](#)

[Building intelligent machines to transform data into knowledge](#)

[The three different types of machine learning](#)

[Making predictions about the future with supervised learning](#)

[Classification for predicting class labels](#)

[Regression for predicting continuous outcomes](#)

[Solving interactive problems with reinforcement learning](#)

[Discovering hidden structures with unsupervised learning](#)

[Finding subgroups with clustering](#)

[Dimensionality reduction for data compression](#)

[Introduction to the basic terminology and notations](#)

[A roadmap for building machine learning systems](#)

[Preprocessing – getting data into shape](#)

[Training and selecting a predictive model](#)

[Evaluating models and predicting unseen data instances](#)

[Using Python for machine learning](#)

[Installing Python and packages from the Python Package Index](#)

[Using the Anaconda Python distribution and package manager](#)

[Packages for scientific computing, data science, and machine learning](#)

[Summary](#)

[2. Training Simple Machine Learning Algorithms for Classification](#)

[Artificial neurons – a brief glimpse into the early history of machine learning](#)

[The formal definition of an artificial neuron](#)

[The perceptron learning rule](#)

[Implementing a perceptron learning algorithm in Python](#)

[An object-oriented perceptron API](#)

[Training a perceptron model on the Iris dataset](#)

[Adaptive linear neurons and the convergence of learning](#)

[Minimizing cost functions with gradient descent](#)

[Implementing Adaline in Python](#)

[Improving gradient descent through feature scaling](#)

[Large-scale machine learning and stochastic gradient descent](#)

[Summary](#)

[3. A Tour of Machine Learning Classifiers Using scikit-learn](#)

[Choosing a classification algorithm](#)

[First steps with scikit-learn – training a perceptron](#)

[Modeling class probabilities via logistic regression](#)

[Logistic regression intuition and conditional probabilities](#)

[Learning the weights of the logistic cost function](#)

[Converting an Adaline implementation into an algorithm for logistic regression](#)

[Training a logistic regression model with scikit-learn](#)

[Tackling overfitting via regularization](#)

[Maximum margin classification with support vector machines](#)

[Maximum margin intuition](#)

[Dealing with a nonlinearly separable case using slack variables](#)

[Alternative implementations in scikit-learn](#)

[Solving nonlinear problems using a kernel SVM](#)

[Kernel methods for linearly inseparable data](#)

[Using the kernel trick to find separating hyperplanes in high-dimensional space](#)

[Decision tree learning](#)

[Maximizing information gain – getting the most bang for your buck](#)

[Building a decision tree](#)

[Combining multiple decision trees via random forests](#)

[K-nearest neighbors – a lazy learning algorithm](#)

[Summary](#)

[4. Building Good Training Sets – Data Preprocessing](#)

[Dealing with missing data](#)

[Identifying missing values in tabular data](#)

[Eliminating samples or features with missing values](#)

[Imputing missing values](#)

[Understanding the scikit-learn estimator API](#)

[Handling categorical data](#)

[Nominal and ordinal features](#)

[Creating an example dataset](#)

[Mapping ordinal features](#)

[Encoding class labels](#)

[Performing one-hot encoding on nominal features](#)

[Partitioning a dataset into separate training and test sets](#)

[Bringing features onto the same scale](#)

[Selecting meaningful features](#)

[L1 and L2 regularization as penalties against model complexity](#)

[A geometric interpretation of L2 regularization](#)

[Sparse solutions with L1 regularization](#)

[Sequential feature selection algorithms](#)

[Assessing feature importance with random forests](#)

[Summary](#)

[5. Compressing Data via Dimensionality Reduction](#)

[Unsupervised dimensionality reduction via principal component analysis](#)

[The main steps behind principal component analysis](#)

[Extracting the principal components step by step](#)

[Total and explained variance](#)

[Feature transformation](#)

[Principal component analysis in scikit-learn](#)

[Supervised data compression via linear discriminant analysis](#)

[Principal component analysis versus linear discriminant analysis](#)

[The inner workings of linear discriminant analysis](#)

[Computing the scatter matrices](#)

[Selecting linear discriminants for the new feature subspace](#)

[Projecting samples onto the new feature space](#)

[LDA via scikit-learn](#)

[Using kernel principal component analysis for nonlinear mappings](#)

[Kernel functions and the kernel trick](#)

[Implementing a kernel principal component analysis in Python](#)

[Example 1 – separating half-moon shapes](#)

[Example 2 – separating concentric circles](#)

[Projecting new data points](#)

[Kernel principal component analysis in scikit-learn](#)

[Summary](#)

[6. Learning Best Practices for Model Evaluation and Hyperparameter Tuning](#)

[Streamlining workflows with pipelines](#)

[Loading the Breast Cancer Wisconsin dataset](#)

[Combining transformers and estimators in a pipeline](#)

[Using k-fold cross-validation to assess model performance](#)

[The holdout method](#)

[K-fold cross-validation](#)

[Debugging algorithms with learning and validation curves](#)

[Diagnosing bias and variance problems with learning curves](#)

[Addressing over- and underfitting with validation curves](#)

[Fine-tuning machine learning models via grid search](#)

[Tuning hyperparameters via grid search](#)

[Algorithm selection with nested cross-validation](#)

[Looking at different performance evaluation metrics](#)

[Reading a confusion matrix](#)

[Optimizing the precision and recall of a classification model](#)

[Plotting a receiver operating characteristic](#)

[Scoring metrics for multiclass classification](#)

[Dealing with class imbalance](#)

[Summary](#)

[7. Combining Different Models for Ensemble Learning](#)

[Learning with ensembles](#)

[Combining classifiers via majority vote](#)

[Implementing a simple majority vote classifier](#)

[Using the majority voting principle to make predictions](#)

[Evaluating and tuning the ensemble classifier](#)

[Bagging – building an ensemble of classifiers from bootstrap samples](#)

[Bagging in a nutshell](#)

[Applying bagging to classify samples in the Wine dataset](#)

[Leveraging weak learners via adaptive boosting](#)

[How boosting works](#)

[Applying AdaBoost using scikit-learn](#)

[Summary](#)

[8. Applying Machine Learning to Sentiment Analysis](#)

[Preparing the IMDb movie review data for text processing](#)

[Obtaining the movie review dataset](#)

[Preprocessing the movie dataset into more convenient format](#)

[Introducing the bag-of-words model](#)

[Transforming words into feature vectors](#)

[Assessing word relevancy via term frequency-inverse document frequency](#)

[Cleaning text data](#)

[Processing documents into tokens](#)

[Training a logistic regression model for document classification](#)

[Working with bigger data – online algorithms and out-of-core learning](#)

[Topic modeling with Latent Dirichlet Allocation](#)

[Decomposing text documents with LDA](#)

[LDA with scikit-learn](#)

[Summary](#)

[9. Embedding a Machine Learning Model into a Web Application](#)

[Serializing fitted scikit-learn estimators](#)

[Setting up an SQLite database for data storage](#)

[Developing a web application with Flask](#)

[Our first Flask web application](#)

[Form validation and rendering](#)

[Setting up the directory structure](#)

[Implementing a macro using the Jinja2 templating engine](#)

[Adding style via CSS](#)

[Creating the result page](#)

[Turning the movie review classifier into a web application](#)

[Files and folders – looking at the directory tree](#)

[Implementing the main application as app.py](#)

[Setting up the review form](#)

[Creating a results page template](#)

[Deploying the web application to a public server](#)

[Creating a PythonAnywhere account](#)

[Uploading the movie classifier application](#)

[Updating the movie classifier](#)

[Summary](#)

10. Predicting Continuous Target Variables with Regression Analysis

Introducing linear regression

Simple linear regression

Multiple linear regression

Exploring the Housing dataset

Loading the Housing dataset into a data frame

Visualizing the important characteristics of a dataset

Looking at relationships using a correlation matrix

Implementing an ordinary least squares linear regression model

Solving regression for regression parameters with gradient descent

Estimating coefficient of a regression model via scikit-learn

Fitting a robust regression model using RANSAC

Evaluating the performance of linear regression models

Using regularized methods for regression

Turning a linear regression model into a curve – polynomial regression

Adding polynomial terms using scikit-learn

Modeling nonlinear relationships in the Housing dataset

Dealing with nonlinear relationships using random forests

Decision tree regression

Random forest regression

Summary

11. Working with Unlabeled Data – Clustering Analysis

Grouping objects by similarity using k-means

K-means clustering using scikit-learn

A smarter way of placing the initial cluster centroids using k-means++

Hard versus soft clustering

Using the elbow method to find the optimal number of clusters

Quantifying the quality of clustering via silhouette plots

Organizing clusters as a hierarchical tree

Grouping clusters in bottom-up fashion

Performing hierarchical clustering on a distance matrix

Attaching dendrograms to a heat map

Applying agglomerative clustering via scikit-learn

Locating regions of high density via DBSCAN

Summary

12. Implementing a Multilayer Artificial Neural Network from Scratch

Modeling complex functions with artificial neural networks

Single-layer neural network recap

[Introducing the multilayer neural network architecture](#)

[Activating a neural network via forward propagation](#)

[Classifying handwritten digits](#)

[Obtaining the MNIST dataset](#)

[Implementing a multilayer perceptron](#)

[Training an artificial neural network](#)

[Computing the logistic cost function](#)

[Developing your intuition for backpropagation](#)

[Training neural networks via backpropagation](#)

[About the convergence in neural networks](#)

[A few last words about the neural network implementation](#)

[Summary](#)

[13. Parallelizing Neural Network Training with TensorFlow](#)

[TensorFlow and training performance](#)

[What is TensorFlow?](#)

[How we will learn TensorFlow](#)

[First steps with TensorFlow](#)

[Working with array structures](#)

[Developing a simple model with the low-level TensorFlow API](#)

[Training neural networks efficiently with high-level TensorFlow APIs](#)

[Building multilayer neural networks using TensorFlow's Layers API](#)

[Developing a multilayer neural network with Keras](#)

[Choosing activation functions for multilayer networks](#)

[Logistic function recap](#)

[Estimating class probabilities in multiclass classification via the softmax function](#)

[Broadening the output spectrum using a hyperbolic tangent](#)

[Rectified linear unit activation](#)

[Summary](#)

[14. Going Deeper – The Mechanics of TensorFlow](#)

[Key features of TensorFlow](#)

[TensorFlow ranks and tensors](#)

[How to get the rank and shape of a tensor](#)

[Understanding TensorFlow's computation graphs](#)

[Placeholders in TensorFlow](#)

[Defining placeholders](#)

[Feeding placeholders with data](#)

[Defining placeholders for data arrays with varying batch sizes](#)

Variables in TensorFlow

Defining variables

Initializing variables

Variable scope

Reusing variables

Building a regression model

Executing objects in a TensorFlow graph using their names

Saving and restoring a model in TensorFlow

Transforming Tensors as multidimensional data arrays

Utilizing control flow mechanics in building graphs

Visualizing the graph with TensorBoard

Extending your TensorBoard experience

Summary

15. Classifying Images with Deep Convolutional Neural Networks

Building blocks of convolutional neural networks

Understanding CNNs and learning feature hierarchies

Performing discrete convolutions

Performing a discrete convolution in one dimension

The effect of zero-padding in a convolution

Determining the size of the convolution output

Performing a discrete convolution in 2D

Subsampling

Putting everything together to build a CNN

Working with multiple input or color channels

Regularizing a neural network with dropout

Implementing a deep convolutional neural network using TensorFlow

The multilayer CNN architecture

Loading and preprocessing the data

Implementing a CNN in the TensorFlow low-level API

Implementing a CNN in the TensorFlow Layers API

Summary

16. Modeling Sequential Data Using Recurrent Neural Networks

Introducing sequential data

Modeling sequential data – order matters

Representing sequences

The different categories of sequence modeling

RNNs for modeling sequences

Understanding the structure and flow of an RNN

[Computing activations in an RNN](#)

[The challenges of learning long-range interactions](#)

[LSTM units](#)

[Implementing a multilayer RNN for sequence modeling in TensorFlow](#)

[Project one – performing sentiment analysis of IMDb movie reviews using multilayer RNNs](#)

[Preparing the data](#)

[Embedding](#)

[Building an RNN model](#)

[The SentimentRNN class constructor](#)

[The build method](#)

[Step 1 – defining multilayer RNN cells](#)

[Step 2 – defining the initial states for the RNN cells](#)

[Step 3 – creating the RNN using the RNN cells and their states](#)

[The train method](#)

[The predict method](#)

[Instantiating the SentimentRNN class](#)

[Training and optimizing the sentiment analysis RNN model](#)

[Project two – implementing an RNN for character-level language modeling in TensorFlow](#)

[Preparing the data](#)

[Building a character-level RNN model](#)

[The constructor](#)

[The build method](#)

[The train method](#)

[The sample method](#)

[Creating and training the CharRNN Model](#)

[The CharRNN model in the sampling mode](#)

[Chapter and book summary](#)

[Index](#)