

Symbols

@tag syntax 211
\$HADOOP_CONF environment variable 151

Numerics

20 newsgroups data set
 classifying with naive Bayes 276–280
 data extraction 276–277
 testing model 278–280
 training classifier 277–278
classifying with SGD 265–273
 parsing and tokenizing features for 20
 newsgroups data 268
 previewing data set 266–268
 training code for 20 newsgroups data
 268–273
performance of SGD classifier with 291–295

A

AbstractVectorClassifier class 285, 290, 304, 354
Action column 267
AdaptiveLogisticRegression class 284–286,
291–293, 298, 304–306, 324, 331, 348–349
addToVector() method 351
aggregate(DoubleDoubleFunction)
 method 364
aggregate(DoubleFunction) method 364
aggregating 85
Algorithm parameter 305
algorithms
 canopy clustering. *See* canopy clustering
 algorithm
 clustering. *See* clustering algorithms

Dirichlet clustering 174
distributed item-based
 designing 95–98
 implementing with MapReduce
 paradigm 98–107
fuzzy k-means. *See* fuzzy k-means algorithm
item-based recommendation 57–58
k-means. *See* k-means algorithm
learning. *See* learning algorithm to train classi-
fication mode, specific algorithms by name
model-based. *See* model-based algorithm
slope-one recommender 60–61
user-based recommender 43–44
alpha method 305
alpha0 177
ALSWRFactorizer implementation 64
Alzheimer’s disease 232
Amazon 3
 AWS Elastic MapReduce console 112
 EC2 service 93
 Elastic MapReduce 108, 112
 S3 storage service 112
Amazon Elastic MapReduce tab, AWS Elastic
 MapReduce console 113
Analyzer class 160–162, 192
analyzing output
 inter-cluster distance 188–191
 intra-cluster distance 188–191
 mixed and overlapping clusters 191
anonymous users 83–85
 aggregating 85
 temporary users with PlusAnonymousUser-
 DataModel class 84–85
Ant build tool 8
Apache Avro 313, 315
Apache Cascading 309

Apache Commons Codec 213
 Apache Hadoop 1–2, 308, 313
 and math 366
 clusters 201–202, 211, 220, 224
 customizing configurations of 201–202
 installing 9
 pseudo-distributing recommender 111
 running MapReduce paradigm with 107–110
 configuring mappers and reducers 110
 running recommendations 108–109
 setting up 107–108
 scale with Apache Mahout and 5–6
 tutorial 99
 Apache Hadoop clusters, local, running cluster-
 ing on 199–200
 Apache Hive 309
 Apache Mahout 1–9
 injecting content-based techniques into 66–67
 installing 8–9
 launcher, executing jobs on Hadoop cluster
 using 201–202
 machine learning themes 3–5
 classification 4–5
 clustering 3–4
 recommender engines 3
 scale, with Apache Hadoop 5–6
 setting up 6–9
 installing 8–9
 Java platform and IDEs 7
 story of 2
 Apache Pig 309
 Apache Software Foundation 2
 Apache Thrift 310, 313, 326
 and classification 332
 IDL 335
 Apache Zookeeper 328, 332
 setup 337
 structuring classifier updates 330
 Appearances table 346–347
 applications of classification 231–232
 Applications/Utilities folder 7
 Area Under the Curve 251
 artists, dictionary of Last.fm 217–218
 ASCII character set 322
 assign(double) 365
 assign(Vector, DoubleDoubleFunction) 364
 AsterData 309
 asymmetrical normal distribution 172, 176
 attrition, classification problem 231
 AUC (area under the curve)
 computation of 285–287
 optimizing based on per-user, Shop It To Me
 case study 348–349
 Auc class 284–286, 288–289

auc() method 286
 authorEnc encoder 303
 authorPlusSubjectEnc encoder 303
 average log likelihood, computing 289–290
 AverageAbsoluteDifferenceRecommender-
 Evaluator implementation 20
 AveragingPreferenceInferer class 56
 AWS Console screen 113
 AWS Elastic MapReduce console, Amazon
 112–113

B

bad feature extraction 295
 bags of words, vectors as 261
 batch size, classification 311
 Bayes algorithms 304
 BIGINT data type 34
 bipartite graph 216–217
 Boolean data
 creating and evaluating with 37
 evaluating precision and recall with 38
 Boolean preferences 35
 bottom-up approach, to clustering
 algorithms 166–167
 broken feature extraction 298–300
 broken tokenizer 299
 built-in data, Mahout 248
 burn 175

C

cachePreparedStatements parameter 34
 CachingUserSimilarity class 53
 canopy centers, improving k-means clustering
 using 159–160
 canopy clustering 155, 158–160
 algorithm 156–159
 improving k-means clustering using canopy
 centers 159–160
 seeding k-means centroids using 156
 canopy generation 158
 CanopyClusterer 156–158, 183
 CanopyDriver 156, 158–159, 162, 171, 183
 categorical features, encoding 262–263
 categorical value 235–237, 319
 categories, defining for target variable 240
 CategoryFeatureEncoder 351
 centers, fixed number of 166
 centroids
 k, running canopy clustering algorithm to
 select 158–159
 k-means, seeding using canopy clustering 156

- classifiable data
 - converting into vectors 260–265
 - feature hashing with Mahout API 261–265
 - representing data as vector 260–261
 - preprocessing raw data into 257–260
 - computational marketing example 258–260
 - transforming raw data 258
- classification 4–5, 227–254
 - colored shape example 242
 - definition 229
 - fundamentals of systems for 229–232
 - applications of classification 231–232
 - classification versus recommendation and clustering 230–231
 - how systems work 232–239
 - four types of values for predictor variables 236–238
 - models 234
 - predictor variables versus target variable 234–235
 - records, fields, and values 235–236
 - supervised versus unsupervised learning 238–239
 - training versus test versus production 234
 - in batch 311
 - step-by-step simple example 245–254
 - choosing learning algorithm to train model 247–250
 - data and challenge 246
 - improving performance of color-fill classifier 250–254
 - training model to find color-fill 246–247
 - using Mahout for 228–229
 - versus clustering and recommendation 230
 - when to use Mahout 228
 - wine example 230, 235
 - work flow in typical projects 239–245
 - See also* deploying classifier; evaluating classifiers; Shop It To Me case study; training classification model
- classification algorithm, selection 247
- classification model, training 232
- classifier
 - as service 310, 324
 - client component 325, 338
 - deployment 307
 - logging 326
 - on relational database 313
 - response time 312
 - scalability and speed requirement 310
 - scoping input size 308
 - server component 325
 - server example 332
 - See also* training classification model; deploying classifier; evaluating classifiers
- classifier accuracy 235
- classifier algorithm
 - on continuous variables 243
 - selection 243
- classifier evaluation API 284–295
 - computation of AUC 285–287
 - computing average log likelihood 289–290
 - confusion matrices and entropy matrices 287–289
 - dissecting model 290–291
 - performance of SGD classifier with 20 newsgroups 291–295
- classifier model
 - at large scale 314
 - building with Mahout 249
 - evaluating 239, 245
 - records and fields 235
 - retesting 245
 - running 239, 245, 252
 - serialization 330
 - training 234, 239
 - updating at runtime 327
- classifier package 287
- classifyScalarNoLink method 354–355
- client-server
 - and load balancing 327
 - performance 327
- clone() method 364–365
- cloud, running recommendation computations in 112–113
- Cluster 149–150, 162, 169, 171, 175
- cluster-based recommendation 65–66
- clusterdump 185–186, 220
- ClusterDumper 153, 155, 170, 185–186, 197
- clusteredPoints directory 153
- clustering 3–4, 117–129, 198–209
 - basics of 118–119
 - batch and online 205–209
 - canopy. *See* canopy clustering
 - classification versus 230–231
 - Dirichlet 173–174
 - algorithm 174
 - MapReduce job version of 176–177
 - distance measures 125–128
 - cosine 127–128
 - Euclidean 126
 - Hello World example 129
 - Manhattan 126
 - squared Euclidean 126
 - Tanimoto 128
 - weighted 128

- clustering (*continued*)
 - Hello World example 120–125
 - analyzing output 125
 - input 120–121
 - measuring similarity of items 119–120
 - quality of 184–197
 - analyzing output 187–191
 - improving 192–197
 - inspecting output 185–187
 - quick-start tutorial for 199–202
 - customizing Apache Hadoop configurations 201–202
 - running clustering on local Hadoop cluster 199–200
 - real-world data 173
 - tuning performance of 202–205
 - See also* real-world applications of clustering
- clustering algorithms 145–183
 - approaches to 166–168
 - bottom-up 166–167
 - fixed number of centers 166
 - top-down 167
 - fuzzy k-means 168–171
 - case study 170–171
 - fuzziness factor 170
 - MapReduce implementation of 169–170
 - k-means 146–163
 - canopy clustering 155–160
 - case study 160–163
 - description of 147–148
 - MapReduce job 150–151
 - LDA (latent Dirichlet allocation) 177–183
 - applications of topic modeling 182–183
 - case study 180–181
 - TF-IDF vs. 179
 - tuning parameters of 179–180
 - model-based 171–177
 - deficiencies of k-means algorithm 172–173
 - Dirichlet clustering 173–174
 - example 174–177
 - problems 163–165
 - exclusive clustering 164
 - hierarchical clustering 164–165
 - overlapping clustering 164
 - probabilistic clustering 165
- clustering problems in Stack Overflow, finding 222–224
- clusters, reducing number of 205
- ClusterSimilarity interface 66
- Clusty 4
- cold start problem 84
- collaborative filtering framework 14–15
- collections, speeding up 28–29
- collective intelligence 1
- collisions, feature 264–265
- color-fill classification example 245–254
 - choosing learning algorithm to train model 247–250
 - building model using Mahout 249–250
 - checking Mahout's built-in data 248
 - start running Mahout 247
 - data and challenge 246
 - improving performance of color-fill classifier 250–254
 - building more interesting model 251–252
 - evaluating model 250–251
 - testing again 252
 - testing using new data 252–253
 - trying other models 253–254
 - training model to find color-fill 246–247
- combiner, for partial products 105
- Comparable interface 134
- compatibility, of implementations 37–40
- Complementary Naïve Bayes classifier 243
- component interaction, user-based recommender 17
- computational marketing example 258–260
- computations
 - distributed, benefits and drawbacks of 93–94
 - recommendation. *See* recommendation computations
- computing
 - AUC 285–287
 - average log likelihood 289–290
- conf/mapred-site.xml file 107
- Configuration class 211, 219, 222
- configure() method 196
- confusion matrices 251, 282, 287–290
- ConfusionMatrix class 284–285, 287–288
- Connector/J driver, MySQL 34
- content-based recommendation 14, 78–79
- content-based recommenders, injecting techniques into Apache Mahout 66–67
- continuous value versus categorical 237
- continuous features, encoding 262
- continuous values 236, 319
- ContinuousValueEncoder 323
- convergenceThreshold 122, 129, 150–151, 155, 169
- co-occurrence 100, 103, 216–217
- co-occurrence matrix 95–96
- coordinating model updates 327–330
- cosine distance measure 127–128
- cosine measure similarity 52
- CosineDistanceMeasure 128–129, 154, 190, 194–195
- cost of errors, recognizing difference in 284

CPU (Central Processing Unit)-bound operations, avoiding performance pitfalls in 203–204

- using appropriate vector representation 203–204
- using fast distance measure 204
- using SparseVector type based on distance-measure computation 204

Create New Job Flow option, AWS Elastic MapReduce console 113

createParameters() method 196

cross(Vector) method 364

CrossFoldLearner class 284–286, 292–293, 304–306

CrossFoldLearner, vector encoding 309

CSV data, parsing efficiently 321

Custom Jar type option, AWS Elastic MapReduce console 113

D

data

- acquiring and retaining large-scale 314–316
 - acquiring 315
 - partitioning and storing 315–316
 - working incrementally 316
- color-fill classification example 246
- converting classifiable into vectors 260–265
 - feature hashing with Mahout API 261–265
 - representing data as vector 260–261
- database-based 32
- denormalizing and downsampling 316–318
 - full reduce joins 318
 - in-memory joins 317
 - join target first 317
 - merge joins 317
- file-based 30–31
- Mahout built-in 248
- preprocessing raw into classifiable 257–260
 - computational marketing example 258–260
 - transforming raw data 258
- reading and encoding at speed 320–324
 - direct value interface to value encoders 323–324
 - segment bytes 321–323
- storing as files 315
 - using new to test model 252–253

data extraction for naive Bayes 276–277

data preprocessing 211–212

data set analysis. *See also* Stack Overflow, analyzing data set of

data set, for GroupLens project 23–25

- extracting recommender input 23–24
- slope-one recommender 24–25

- database-based data 32
- DataModel interface 17, 20, 31, 34, 36–38, 45, 62, 360
- DataModel models, in-memory 30–34
 - configuring 33–34
 - database-based data 32
 - file-based data 30–31
 - GenericDataModel model 30
 - JDBC and MySQL RDBMS 32–33
 - Refreshable interface components 31–32
 - update files 32
- DataModelBuilder interface 37
- DataSource object 33–34
- dating sites, example data from 71–72
- decayExponent method 305
- decomposition-based recommenders, singular value 63–64
- denormalizing 316–318
 - full reduce joins 318
 - in-memory joins 317
 - join target first 317
 - merge joins 317
- DenseMatrix class 365
- DenseVector class 132, 134, 144, 203, 205, 363
- dependencies, in user-based recommender system 31
- deploying classifier 307–340
 - building training pipeline for large systems 313–324
 - acquiring and retaining large-scale data 314–316
 - denormalizing and downsampling 316–318
 - reading and encoding data at speed 320–324
 - training pitfalls 318–319
 - determining scale and speed requirements 310–313
 - balancing big versus fast 312–313
 - determining size 310–312
 - in huge systems 308–310
 - deploying scalable classifier service 310
 - optimizing feature extraction as needed 309
 - optimizing vector encoding as needed 309
 - scoping out problem 308–309
 - integrating classifier 324–331
 - key issues for integration 325–330
 - model serialization 330–331
 - thrift-based classification server example 332–340
 - accessing classifier service 338–340
 - running classification server 336–338
- deployment, testing 87–88
- derived values, adding to tune classifier 301–304
- designing for speed 326–327
- determinant() method 366

Developer Tools, Apple 8
 DF (document frequency) 136
 diabetes, classification problem 232
 dictionary of Last.fm artists 217–218
 Dictionary type 186
 DictionaryVectorizer class 138, 151, 162
 DiffStorage implementation, and memory 62
 direct value interface to value encoders 323–324
 directories, and data partitioning 315
 Dirichlet clustering 173–174
 algorithm 174
 MapReduce job version of 176–177
 DirichletClusterer class 174–175, 183
 DirichletDriver class 174, 176, 183
 discriminative 179
 DisplayCanopy class 158
 DisplayDirichlet class 175–176
 DisplayFuzzyKMeans class 169
 DisplayKMeans class 149
 dissecting model 290–291
 distance measure 125–128
 and feature selection, weighting features 188
 computation, using SparseVector type based
 on 204
 cosine 127–128
 Euclidean 126
 fast 204
 Hello World example 129
 Manhattan 126
 squared Euclidean 126
 Tanimoto 128
 weighted 128
 writing custom 195–197
 distance(Vector v1, Vector v2) 195
 DistanceMeasure interface 126, 148, 155–156,
 168, 190, 195, 197
 distributed computation, benefits and drawbacks
 of 93–94
 distributed item-based algorithm
 designing 95–98
 computing user vectors 96
 co-occurrence matrix 95–96
 producing recommendations 96–97
 understanding results 97
 implementing with MapReduce paradigm
 98–107
 calculating co-occurrence 100
 generating user vectors 99–100
 making recommendations 105–107
 matrix multiplication 101–105
 distribution, asymmetrical 172
 dividing responsibilities, when integrating
 classifier 325–326
 document vectors, generation of 192–195

domain-specific information 77–83
 IDRescorer method
 incorporating gender in 80–82
 modifying recommendations with 79–80
 packaging custom recommender 82–83
 recommending based on content 78–79
 dot(Vector) method 364
 DoubleMetaphone 213–214, 223
 downsampling 309, 312, 316–318
 full reduce joins 318
 in-memory joins 317
 join target first 317
 merge joins 317
 DSL (domain-specific language) 344

E

EC2 service, Amazon 93
 Eclipse 7
 efficient I/O, parallelism 324
 Elastic MapReduce 6, 108, 112
 Element objects 363, 365
 EM (expectation maximization) algorithm 148
 email marketing system, Shop It To Me case
 study 344–346
 EMR (Elastic MapReduce) service 208
 encoded data, training SGD model with 273
 encoders
 direct value interface to value 323–324
 setting up vector 269
 encoding
 categorical and word-like features 262–263
 continuous features 262
 data at speed 320–324
 direct value interface to value encoders
 323–324
 segment bytes 321–323
 feature vector, in Shop It To Me case
 study 349–352
 semantic mismatches during 318–319
 text-like features 263–264
 vector, optimizing 309
 entropy matrices 287–289
 ephemeral file 332
 errors, recognizing difference in cost of 284
 Euclidean distance 38, 51
 Euclidean distance similarity metric 73–74
 EuclideanDistanceMeasure class 122, 124, 126,
 129, 148, 150, 157–158
 EuclideanDistanceSimilarity class 37, 51
 evaluate() method 20, 46, 53, 59
 evaluating classification model
 color-fill classification example 250–251
 overview 245

- evaluating classifiers 281–306
 - classifier evaluation API 284–295
 - computation of AUC 285–287
 - computing average log likelihood 289–290
 - confusion matrices and entropy matrices 287–289
 - dissecting model 290–291
 - performance of SGD classifier with 20 newsgroups 291–295
 - deciding what good means 282–283
 - getting rapid feedback 282
 - problems with classifiers 295–300
 - broken feature extraction 298–300
 - target leaks 295–298
 - recognizing difference in cost of errors 284
- evaluating, recommenders 18–21
 - assessing result 20–21
 - GroupLens project data set 23–25
 - precision and recall 21–23
 - RecommenderEvaluator evaluator 19–20
 - training data and scoring 18–19
- evaluation package 286
- evaluation, multi-threaded 313
- examples module 45
- exclusive clustering 164
- extracting features to build classifier 256–257
- extraction feature, broken 298–300
- extraction feature, optimizing 309
- ExtractReuters class 139

F

- Facebook 3
- FarthestNeighborClusterSimilarity class 66
- fast distance measure 204
- FastByIDMap class 29–30
- FastIDSet class 29, 36–37
- FastLine class 323–324
- FastMap class 29
- feature collisions 264–265
- feature encoding 312
- feature extraction 235
 - broken 298–300
 - incremental 316
 - optimizing 309
 - optimizing at scale 309
 - to build classifier 256–257
- feature hashing
 - overview 261
 - with Mahout API 261–265
 - encoding categorical and word-like features 262–263
 - encoding continuous features 262

- encoding text-like features 263–264
 - feature collisions 264–265
- feature selection
 - avoiding common pitfalls in 212–215
 - distance measure and 188
- feature vector encoding 309
- feature vectors
 - encoding, Shop It To Me case study 349–352
 - linear combination of, Shop It To Me case study 353
- FeatureEncoder class 355–356
- features, parsing and tokenizing for 20 newsgroups data 268
- feature-weighting method 211–212
- feedback, rapid, when evaluating classifiers 282
- fields, in classification 235–236
- file-based data 30–31
- file-based preference data 89
- FileDataModel class 30–32, 39, 45, 72, 89
- files
 - and data partitioning 315
 - large 316
 - update 32
- FileSystem class 151, 161
- fixed-size neighborhoods 46–47
- fkmeans 169
- FLOAT data type 34
- float data type 34
- floating-point, efficient parsing 323
- fluff variables, removing 301
- forests, random 276
- fraud detection
 - classification problem 231
 - classifier problem 245
- Freebase Wikipedia Extraction project 92
- FreeBSD tcsh shell 7
- full reduce joins 318
- Functions 364–365
- fuzziness factor 170
- fuzzy k-means algorithm 168–171
 - case study, clustering news articles using fuzzy k-means 170–171
 - fuzziness factor 170
 - MapReduce job implementation of 169–170
- FuzzyKMeansClusterer class 168–169, 183
- FuzzyKMeansDriver class 168, 170–171, 183

G

- GaussianClusterDistribution 175–177
- gender, incorporating in IDRescorer method 80–82
- gender.dat file 72, 86
- generateSamples 148–149, 157, 168, 174–175

GenericBooleanPrefDataModel class 36–37,
 39, 76
 GenericBooleanPrefUserBasedRecommender
 class 39, 76
 GenericDataModel class 30, 36–37
 GenericItemBasedRecommender class 58–59,
 68, 78
 GenericItemPreferenceArray implementation 28
 GenericItemSimilarity class 58
 GenericJDBCDataModel class 33
 GenericPreference object 27
 GenericUserBasedRecommender class 39, 45, 68
 GenericUserPreferenceArray implementation 28
 Genius feature, iTunes 4
 GET request 87
 get(int, int) 365
 get(int) 363
 getItemIDsForUser() method 36
 getMean() method 289–290
 getParameters() method 196
 getPoints function 121, 123
 getPreferencesFromUser() method 36
 getPreferenceValue() method 36
 getQuartile() method 290
 getQuick(int) method 363
 GitHub site 9
 GlobalOnlineAuc class 286, 349
 Google News 4–5
 Greenplum 309
 GroupedOnlineAuc 286
 GroupLens data set 45–46
 GroupLens example 36
 GroupLens project, data set 23–25
 extracting recommender input 23–24
 slope-one recommender 24–25
 GroupLensDataModel implementation 45

H

hadoop.jar 200
 HADOOP_CONF_DIR 201
 HADOOP_CONF_DIR variable 201
 HADOOP_HOME 201
 Hadoop. *See* Apache Hadoop
 Haselgrove, Henry 92
 HashMap 29, 132, 218–219
 HDFS (Hadoop Distributed Filesystem) 98–99,
 106–108, 153, 205
 Hello World example
 clustering 120–125
 analyzing output 125
 input 120–121
 distance measures 129
 hierarchical clustering 164–165

historical data, collecting 240
 huge systems, deploying classifier in 308–310
 deploying scalable classifier service 310
 optimizing feature extraction as needed 309
 optimizing vector encoding as needed 309
 scoping out problem 308–309
 See also training pipeline for large systems

I

I/O (input/output)-bound operations, avoiding
 pitfalls in 204–205
 reducing number of clusters 205
 using appropriate vector representation 205
 using HDFS replication 205
 I/O bottleneck 205
 IDEs (Integrated Development Environments),
 and Java platform 7
 IDRescorer method
 incorporating gender in 80–82
 modifying recommendations with 79–80
 IDs, in Mahout 15
 implementations, selecting compatible 37–40
 incremental large-scale data acquisition 316
 in-memory DataModel models 30–34
 configuring
 programmatically 34
 via JNDI 33–34
 database-based data 32
 file-based data 30–31
 GenericDataModel model 30
 JDBC and MySQL RDBMS 32–33
 Refreshable interface components 31–32
 update files 32
 in-memory joins 317
 in-memory representations 36–37
 input
 clustering 120–121
 recommender, extracting 23–24
 to recommender engine 15–16
 input size, and classification scalability 228
 insertTokenError 300
 integrating classifier 324–331
 key issues for integration 325–330
 coordinating model updates 327–330
 designing for speed 326–327
 dividing responsibilities 325–326
 logging requests 326
 real-time features are different 326
 model serialization 330–331
 IntelliJ IDEA 7
 interaction variables 302
 interactions, adding to tune classifier 301–304
 inter-cluster distance 188–191

IntPairWritable, DoubleWritable 182
 intra-cluster distance 188–191
 intro.csv file 15
 ISBN (International Standard Book Number) 66
 isFiltered() method 80, 82
 item-based algorithm distributed
 designing 95–98
 implementing with MapReduce paradigm
 98–107
 item-based recommendations 56–59, 64–65
 item-based recommenders 14, 74–75
 ItemSimilarity interface 57–59, 67, 77–78, 95
 iterateNonZero() method 363, 365
 Iterator 363
 iterator() method 363

J

JAR file 86, 108, 125, 139, 200, 208, 246, 248
 Java library 2
 Java platform 7
 Java Preferences application 7
 Java servlet container 86
 JAVA_HOME 139
 JDBC (Java database connectivity) 32–33
 JDBCDataModel interface 32–33, 89
 Jetty plugin 87
 JMS (Java Messaging Service) 223
 JNDI (Java Naming and Directory Interface),
 configuring in-memory DataModel models
 via 33–34
 jobs
 executing on Hadoop clusters using Apache
 Mahout launcher 201–202
 MapReduce, running using random seed
 generator 150–155
 joins 317
 full reduce 318
 in-memory 317
 merge 317
 overview 317
 JVM (Java Virtual Machine) 7, 359

K

k centroids, running canopy clustering algorithm
 to select 158–159
 key-value pairs 99
 kmeans 152–154, 159
 k-means algorithm 146–163
 canopy clustering 155–160
 algorithm 156–158
 improving k-means clustering using canopy
 centers 159–160

 seeding k-means centroids using 156
 case study, clustering news articles using
 k-means algorithm 160–163
 deficiencies of 172–173
 asymmetrical normal distribution 172
 clustering real-world data 173
 description of 147–148
 MapReduce job 150–155
 KMeansClusterer class 148–149, 183
 KMeansDriver 208
 KMeansDriver class 148, 150, 152, 159, 162, 183
 Knn (k nearest neighbors) 64
 KnnItemBasedRecommender class 64–65, 69

L

lambda method 305
 large systems. *See* huge systems, deploying classi-
 fier in
 large-scale data 314–316
 acquiring 315
 partitioning and storing 315–316
 working incrementally 316
 Last.fm, tag suggestion for artists on 216–221
 converting tags into vectors 219–220
 creating dictionary of artists 217–218
 running k-means over data 220–221
 using co-occurrence 216–217
 LDA (latent Dirichlet allocation) 146, 177–183
 applications of topic modeling 182–183
 case study, finding topics in news
 documents 180–181
 TF-IDF vs. 179
 tuning parameters of 179–180
 LDADriver class 180, 183
 LDAPrintTopics class 181
 learning algorithm to train classification model
 choosing 273–276
 naive Bayes and complementary naive
 Bayes 275
 random forests 276
 SGD 274–275
 SVM 275
 color-fill classification example 247–250
 naive Bayes, classifying 20 newsgroups data set
 with 276–280
 data extraction 276–277
 testing model 278–280
 training classifier 277–278
 selecting 243–244
 trying alternatives to tune classifier 304
 tuning 305–306
 tweaks to in Shop It To Me case study 348–349
 mixed rank and score learning 349

learning algorithm, Shop It To Me case study (*continued*)
 optimizing based on per-user AUC 348–349
 using 244
See also stochastic gradient descent (SGD) algorithm

learning, supervised versus unsupervised 238–239

learningRate method 305

LengthFilter 193

Libimseti dating site 71

like() method 364–365

Line class 321–322

linear combination of feature vectors, Shop It To Me case study 353

linear expansion of model score, Shop It To Me case study 354–356

linear interpolation item-based recommendation 64–65

List 148–149, 156–157, 168, 175

LoadEvaluator class 76

local Hadoop clusters, running clustering on 199–200

log likelihood, computing average 289–290

logging requests, when integrating classifier 326

logistic regression 252

log-likelihood tests, computing smarter similarity with 55–56

LogLikelihoodSimilarity class 38, 55, 76, 92

long data type 34

LongWritable 212, 217, 219, 222

LowerCaseFilter 193

LSI (latent semantic indexing) 137

Lucene Analyzer 192, 195, 197

Lucene project 2

Lucene tokenizer 268, 298

M

m2eclipse plugin 8

machine learning library 1

machine learning techniques 5–6

machine learning, themes 3–5
 classification 4–5
 clustering 3–4
 recommender engines 3

Mahout. *See* Apache Mahout

Manhattan distance measure 126

ManhattanDistanceMeasure class 126, 129

Manning site 9

map function, MapReduce paradigm 6

Mapper class 199, 202, 211–212, 217–220, 222, 366

Mapper interface 99–101, 104–105, 108

mappers, configuring 110

mapred.child.java.opts property 107

mapred.map.tasks 201

mapred.reduce.tasks 201

MapReduce job 150–151
 implementation of fuzzy k-means algorithm 169–170
 running using random seed generator 151–155
 version of Dirichlet clustering 176–177

MapReduce paradigm 6
 implementing distributed item-based algorithm with 98–107
 calculating co-occurrence 100
 generating user vectors 99–100
 making recommendations 105–107
 matrix multiplication 101–105
 running with Apache Hadoop 107–110
 configuring mappers and reducers 110
 running recommendations 108–109
 setting up 107–108

maps, FastByIDMap 29

marketing system, Shop It To Me case study 344–346

math 362–366
 and Apache Hadoop 366
 matrices 365–366
 vectors 362–365
 advanced methods 364–365
 implementation 362–363
 operations 363–364

matrices 365–366
 confusion and entropy 287–289
 co-occurrence 95–96
 Matrix operations 365–366

Matrix 365–366

matrix multiplication 96
 by partial products 102–105
 rethinking 101–102

MatrixWritable 366

Maven command 152

Maven tool, installing 8

maximum response time 311–312

maxIterations 151, 169

MD5 hash 330

measuring
 distance 125–128
 and feature selection 188
 cosine distance measure 127–128
 Euclidean distance measure 126
 Hello World example 129
 Manhattan distance measure 126
 squared Euclidean distance measure 126
 Tanimoto distance measure 128
 weighted distance measure 128

- measuring, distance (*continued*)
 - writing custom distance measure 195–197
 - similarity of items 119–120
 - memory, DiffStorage implementation 62
 - merge joins 317
 - minus(Vector) 364
 - mixed clusters 191
 - mixed rank and score learning, Shop It To Me
 - case study 349
 - MixedGradient class 349
 - mixture modeling 173
 - model score, linear expansion of, Shop It To Me
 - case study 354–356
 - model-based algorithm 171–177
 - deficiencies of k-means algorithm 172–173
 - asymmetrical normal distribution 172
 - clustering real-world data 173
 - Dirichlet clustering 173–174
 - example 174–177
 - MapReduce job version of Dirichlet clustering 176–177
 - model-based recommenders 67–69
 - ModelDissector class 290–291, 293, 297
 - ModelDistribution class 174
 - modeling, applications of 182–183
 - models
 - DataModel. *See* DataModel models
 - GenericDataModel 30
 - models, classification
 - color-fill classification example
 - building more interesting 251–252
 - choosing learning algorithm 247–250
 - evaluating 250–251
 - preliminary thinking on training 246–247
 - testing again 252
 - testing using new data 252–253
 - trying other 253–254
 - coordinating updates 327–330
 - dissecting 290–291
 - evaluating 245
 - overview 234
 - serialization 330–331
 - training 255–280
 - balancing big versus fast 312
 - choosing algorithm 273–276
 - classifying 20 newsgroups data set with SGD 265–273
 - classifying 20 newsgroups data with naive Bayes 276–280
 - converting classifiable data into vectors 260–265
 - extracting features to build classifier 256–257
 - preprocessing raw data into classifiable data 257–260
 - versus test versus production 234
 - work flow for 240–244
 - using in production 245
 - using ModelSerializer for SGD models 331
 - See also* evaluating classifiers; tuning classifiers
 - ModelSerializer class 330–331
 - monitoring recommenders 88–90
 - MovieLens data set 45
 - MyAnalyzer 193–195
 - myFasterOperation 364
 - myOperation 364
 - MySQL RDBMS (relational database management system) 32–33
 - MySQL, Connector/J driver 34
 - MySQL.BooleanPrefDataModel class 39
 - MySQLJDBCDataModel class 33–34, 39, 62
 - MySQLJDBCDiffStorage class 62
-
- ## N
-
- Naive Bayes algorithm
 - classifying 20 newsgroups data set with 276–280
 - data extraction 276–277
 - testing model 278–280
 - training classifier 277–278
 - overview 275
 - Naive Bayes classifier 243
 - scalability 311
 - serialization 330
 - NaN (not a number) symbol 74
 - NearestNeighborClusterSimilarity class 66
 - NearestNUserNeighborhood class 46, 64
 - neighborhoods, user 46
 - fixed-size neighborhoods 46–47
 - threshold-based neighborhood 47–48
 - NetBeans 7–8
 - Netflix 3, 5
 - news articles, clustering
 - using fuzzy k-means algorithm 170–171
 - using k-means algorithm 160–163
 - news documents, finding topics in 180–181
 - NewsKMeansClustering 161, 163
 - numCategories method 305
 - numFeatures method 305
 - numModels method 175
-
- ## O
-
- Object overhead 28
 - Object references 28
 - one cell per word approach to representing data as vector 261

- online clustering techniques 206
- OnlineAuc class 284–286
- OnlineLogisticRegression class 260, 262, 268–270, 273, 306, 324
- OnlineSummarizer class 285, 289, 299
- Ops class 335–336
- optical character recognition software 4
- optimizing
 - based on per-user AUC, Shop It To Me case study 348–349
 - feature extraction 309
 - vector encoding 309
- OutOfMemoryError 360
- output
 - clustering
 - analyzing 125, 187–191
 - inspecting 185–187
 - of recommender engine, analyzing 17–18
- output/directory 109, 111
- overhead of Object in Java 27
- overlapping clustering 164, 191

P

- packaging
 - custom recommender 82–83
 - WAR file 86–87
- parseMenWomen() method 81
- parsing features for 20 newsgroups data 268
- parsing Stack Overflow data set 222
- partial products, matrix multiplication by 102–105
- PartialMultiplyMapper class 105
- partitioning by time, Shop It To Me case study 348
- partitioning large-scale data 315–316
- peak load estimation 310
- Pearson correlation 37–38, 48–50, 56
- Pearson correlation-based metric 73–74
- PearsonCorrelationSimilarity class 37–38, 48, 50–52, 58
- percentCorrect() method 292
- performance
 - DiffStorage implementation and memory 62
 - evaluating 76–77
 - of clustering 202–205
 - speeding up collections 28–29
- performance of color-fill classifier, improving 250–254
 - building more interesting model 251–252
 - evaluating model 250–251
 - testing again 252
 - testing using new data 252–253
 - trying other models 253
- Picasa 4–5
- pipeline, training. *See* training pipeline for large systems
- plus(Matrix) mathematical operation 365
- plus(Vector) mathematical operation 364
- PlusAnonymousUserDataModel class, temporary users with 84–85
- pom.xml file 8
- PorterStemFilter 193–195
- position, using as predictor variable 241–242
- PostgreSQL, implementation of JDBCDataModel for 33
- precision, recall 21–23
 - evaluating 75–76
 - problems with 23
 - RecommenderIRStatsEvaluator evaluator 21–23
- precomputation distributing 62
- predictive analytics 230
- predictor variables, in classification
 - defining 240–241
 - different needed with different data 243
 - four types of values for 236–238
 - position, using as 241–242
 - versus target variable 234–235
- preference data 27–29
 - FastByIDMap map and FastIDSet set 29
 - Preference object 27
 - PreferenceArray interface and implementations 28
 - speeding up collections 28–29
- Preference objects 27–28, 36
- preference values 48
 - ignoring with Tanimoto coefficient 54–55
 - recommender data without 34–40
 - in-memory representations 36–37
 - selecting compatible implementations 37–40
 - when to ignore values 35–36
- PreferenceArray interface 28–30, 36
- PreferenceInferer interface 56
- preferences, inferring 56
- preprocessing raw data into classifiable data 257–260
 - computational marketing example 258–260
 - transforming raw data 258
- preprocessing, data 211–212
- previewing data set, 20 newsgroups 266–268
- prior method 305
- probabilistic clustering 165
- problems, clustering algorithms 163–165
 - exclusive clustering 164
 - hierarchical clustering 164–165
 - overlapping clustering 164
 - probabilistic clustering 165

production
 and classification systems 234
 using classification models in 245
 programmatic configuration, of in-memory Data-
 Model models 34
 Protocol Buffers 313, 315
 pseudo-distributed Hadoop cluster 107
 pseudo-distributed operation 107
 pseudo-distributing, recommenders 110–111

Q

quick-start tutorial, for clustering 199–202
 customizing Apache Hadoop
 configurations 201–202
 running clustering on local Hadoop
 cluster 199–200

R

random forests 276
 Random Forests classifier 243
 random seed generator, running MapReduce job
 using 151–155
 RandomAccessSparseVector 121, 123, 132, 144,
 204, 208, 363
 RandomPointsUtil 148–149, 168
 RandomSeedGenerator class 151–152, 155, 159,
 169–170
 RandomUtils.useTestSeed() method 20, 22
 rapid feedback, when evaluating classifiers 282
 ratesMoreMen() method 81
 ratings.dat file 71, 86
 raw data, preprocessing into classifiable
 data 257–260
 computational marketing example 258–260
 transforming raw data 258
 Reader 190, 192–193, 195
 reading data at speed 320–324
 direct value interface to value encoders
 323–324
 segment bytes 321–323
 reading data, 20 newsgroups data set 270–271
 real-time features 326
 real-world applications of clustering 210–224
 analyzing Stack Overflow data set 221–224
 clustering posts data for surfacing related
 questions 222–223
 clustering user data for surfacing similar
 users 223–224
 finding clustering problems in 222–224
 parsing data set 222
 finding similar users on Twitter 211–215

avoiding common pitfalls in feature
 selection 212–215
 data preprocessing and feature
 weighting 211–212
 suggesting tags for artists on Last.fm 216–221
 converting Last.fm tags into vectors 219–220
 creating dictionary of artists 217–218
 running k-means over data 220–221
 using co-occurrence 216–217
 rec.motorcycles 294–295, 297
 recall, precision and 21–23
 evaluating 75–76
 problems with 23
 RecommenderIRStatsEvaluator evaluator
 21–23
 recommend(long userID, int howMany,
 IDRescorer rescorer) method 79
 recommendation computations 91–114
 distributed item-based algorithm
 designing 95–98
 implementing with MapReduce
 paradigm 98–107
 imagining unconventional uses 113–114
 MapReduce paradigm, running with Apache
 Hadoop 107–110
 pseudo-distributing recommender 110–111
 running in cloud 112–113
 Wikipedia data set 92–94
 benefits and drawbacks of distributing
 computations 93–94
 problems with scale 93
 recommendations 41–69
 item-based 56–59
 making 105–107
 producing from distributed item-based
 algorithm 96–97
 recommenders
 content-based 66–67
 model-based 67–69
 new and experimental 63–66
 slope-one 59–63
 user-based 43–48
 running with Apache Hadoop 108–109
 similarity metrics 48–56
 computing smarter with log-likelihood
 test 55–56
 cosine measure 52
 defining 51–53
 ignoring preference values with Tanimoto
 coefficient 54–55
 inferring preferences 56
 Pearson correlation-based 48–49
 weighting 50–51

- recommendations (*continued*)
 - user-based 42–43
 - recommendations gone right 42–43
 - recommendations gone wrong 42
 - recommender data 26–40
 - in-memory DataModel models 30–34
 - configuring 33–34
 - database-based data 32
 - file-based data 30–31
 - GenericDataModel model 30
 - JDBC and MySQL RDBMS 32–33
 - Refreshable interface components 31–32
 - update files 32
 - preference 27–29
 - FastByIDMap map and FastIDSet set 29
 - Preference object 27
 - PreferenceArray interface and implementations 28
 - speeding up collections 28–29
 - without preference values 34–40
 - in-memory representations 36–37
 - selecting compatible implementations 37–40
 - when to ignore values 35–36
 - recommender engines 3, 15–18
 - analyzing output 17–18
 - input 15–16
 - recommender input file 15
 - Recommender interface 17, 20–21, 31, 45, 79, 84, 86, 110–111
 - Recommender.recommend() method 79
 - Recommender.refresh() method 89
 - RecommenderBuilder object 20
 - RecommenderEvaluator evaluator 19–20
 - RecommenderIRStatsEvaluator evaluator 21–23
 - RecommenderIRStatsEvaluator interface 76
 - RecommenderJob class 98, 108–109
 - recommenders 13–25, 70–90
 - content-based 66–67
 - definition of 14–15
 - evaluating 18–21
 - assessing result 20–21
 - GroupLens project data set 23–25
 - precision and recall 21–23
 - RecommenderEvaluator evaluator 19–20
 - training data and scoring 18–19
 - example data from dating site 71–72
 - finding effective 72–77
 - evaluating 75–77
 - item-based recommenders 74–75
 - slope-one recommender 75
 - user-based recommenders 73–74
 - injecting domain-specific information 77–83
 - IDRescorer method 79–82
 - packaging custom recommender 82–83
 - recommending based on content 78–79
 - model-based 67–69
 - new and experimental 63–66
 - cluster-based recommendation 65–66
 - linear interpolation item-based recommendation 64–65
 - singular value decomposition-based recommenders 63–64
 - pseudo-distributing 110–111
 - recommending to anonymous users 83–85
 - aggregating 85
 - temporary users with PlusAnonymousUserDataModel class 84–85
 - running recommender engine 15–18
 - analyzing output 17–18
 - input 15–16
 - slope-one 24–25, 59–63
 - algorithm 60–61
 - DiffStorage implementation and memory 62
 - distributing precomputation 62–63
 - in practice 61–62
 - updating and monitoring 88–90
 - user-based 43–48
 - algorithm 43–45
 - GroupLens data set 45–46
 - user neighborhoods 46
 - web-enabled 86–88
 - packaging WAR file 86–87
 - testing deployment 87–88
- records, in classification 235–236
- reduce function, MapReduce paradigm 6
- Reducer class 199, 202, 211–212, 217–220, 366
- Reducer interface 99–101, 104, 108
- Reducers 205
- reducers, configuring 110
- refresh() method 31, 89
- Refreshable interface 31–32
- relative ranks, with Spearman correlation 52–53
- reloading data 31
- replication, HDFS 205
- representations, in-memory 36–37
- required throughput 311–312
- rescore() method 80, 82
- response time, maximum 311–312
- responsibilities, dividing when integrating classifier 325–326
- RMSRecommenderEvaluator implementation 20
- root-mean-square calculation 19–20
- Run Your Own Application option, AWS Elastic MapReduce console 113
- runClustering 158
- runJob method 150
- runlogistic method 251
- running thrift-based classification server 336–338

Runtime.freeMemory() method 30
 Runtime.totalMemory() method 30

S

-
- S3 storage service, Amazon 112
 - SaleMail 343–344
 - scalability
 - parallel algorithm advantage 228
 - problems with 93
 - tradeoffs 312
 - with Apache Mahout and Hadoop 5–6
 - scalable classifier service, deploying 310
 - scalable machine learning implementations 1
 - scale requirements 310–313
 - balancing big versus fast 312–313
 - at classification time 313
 - at training time 312
 - determining size 310–312
 - classification batch size 311
 - implications of number of training examples 311
 - maximum response time and required throughput 311–312
 - scoring, training data and 18–19
 - search engines 21
 - seeding, k-means centroids using canopy clustering 156
 - segment bytes 321–323
 - selecting features, distance measure and 188
 - semantic mismatch, integer data 318–319
 - seq2sparse command 139, 142–144
 - SequenceFileDumper class 180
 - SequenceFileFromDirectory class 139
 - SequenceFiles class 161
 - SequenceFilesFromDirectory class 138
 - SequentialAccessSparseVector class 132, 141–142, 177, 204, 219–220
 - Serializable 366
 - serialization, model 330–331
 - Set implementation 29
 - set(int, double) 363
 - setPreferenceInferer() method 56
 - setProbes() method 262
 - setQuick(int, double) 363
 - sets, FastIDSet 29
 - SGD (stochastic gradient descent). *See* stochastic gradient descent
 - sgd package 290
 - Shop It To Me case study 341–357
 - general structure of email marketing system 344–346
 - reasons for choosing Mahout 342–344
 - Mahout outscales competition 343–344
 - need for classification system 342–343
 - what Shop It To Me does 342
 - speeding up classification 352–357
 - linear combination of feature vectors 353
 - linear expansion of model score 354–356
 - training model 346–352
 - avoiding target leaks 348
 - defining goal of classification project 346–348
 - feature vector encoding 349–352
 - learning algorithm tweaks 348–349
 - partitioning by time 348
 - similar users, clustering user data for surfacing 223–224
 - similarity metrics 48–56
 - computing smarter with log-likelihood test 55–56
 - cosine measure 52
 - defining
 - by Euclidean distance 51
 - by relative rank with Spearman correlation 52–53
 - ignoring preferences, with Tanimoto coefficient 54–55
 - inferring preferences 56
 - Pearson correlation-based 48, 50
 - weighting 50–51
 - similarity of items, measuring 119–120
 - Single Node Setup documentation 9
 - singular value decomposition-based recommenders 63–64
 - size() method 363, 365
 - slope-one recommenders 24–25, 59–63
 - algorithm 60–61
 - DiffStorage implementation and memory 62
 - distributing precomputation 62–63
 - in practice 61–62
 - SlopeOneRecommender class 24, 61–62, 69, 111
 - SOAP response 88
 - SOAP-based web service API 88
 - sources 367–368
 - spam detection 230
 - SparseColumnMatrix 365–366
 - SparseMatrix 365–366
 - SparseRowMatrix 365
 - SparseVector 204
 - SparseVector type, using based on distance-measure computation 204
 - SparseVectorsFromSequenceFile class 139, 151, 208
 - SparseVectorsFromSequenceFiles class 138–139, 200, 208
 - Spearman correlation, relative rank with 52–53
 - SpearmanCorrelationSimilarity class 53

speed requirements 310–313
 balancing big versus fast 312–313
 at classification time 313
 at training time 312
 determining size 310–312
 classification batch size 311
 implications of number of training
 examples 311
 maximum response time and required
 throughput 311–312
 speed, designing for 326–327
 SquaredEuclideanDistanceMeasure 126, 129,
 151–152, 154, 169
 Stack Overflow, analyzing data set of 221–224
 clustering posts data for surfacing related
 questions 222–223
 clustering user data for surfacing similar
 users 223–224
 finding clustering problems 222–224
 parsing data set 222
 StandardAnalyzer class 140, 142, 193, 195, 197,
 213–214, 223, 270
 StandardTokenizer class 192–193
 stepOffset method 305
 stochastic gradient descent (SGD) algorithm
 choosing 274–275
 classifying 20 newsgroups data set with 265–273
 parsing and tokenizing features for 20 news-
 groups data 268
 previewing data set 266–268
 training code for 20 newsgroups data
 268–273
 performance of classifier with 20
 newsgroups 291–295
 using ModelSerializer for models 331
 stochastic gradient descent classifier 243, 309
 scalability 311
 serialization 331
 StopFilter class 193, 213, 215
 StopwordsFilter class 195
 storing large-scale data 315–316
 String object 320
 subjectEnc subject text encoder 303
 Subversion 8
 supervised learning 238–239
 supervised learning algorithms 231
 support vector machine (SVM) algorithm 275
 surfacing related questions, clustering posts data
 for 222–223
 surfacing similar users, clustering user data
 for 223–224
 SVD decomposition 347
 SVDR recommender class 63–64, 69
 SVM (support vector machine) 274, 304

System.gc() method 30
 systems, classification. *See* classification

T

tag suggestion. *See* Last.fm, tag suggestion for
 artists on
 Tanimoto coefficient 54–55, 74
 Tanimoto distance measure 128
 TanimotoCoefficientSimilarity class 54–55, 76
 TanimotoDistanceMeasure class 128–129, 159,
 162, 171
 target leaks 241, 295–298, 301, 318
 target variables 232–233
 defining categories for 240
 versus predictor variables 234–235
 Taste open source 2
 taste_preferences table 33
 temporary users, with PlusAnonymousUserData-
 Model class 84–85
 term() method 298
 test data 18, 234
 testing classification models, color-fill classifica-
 tion example
 testing again 252
 using new data 252–253
 testing naive Bayes model 278–280
 testing, deployment 87–88
 text-like features, encoding 263–264
 text-like value 237
 TF (term frequency) 136
 TF-IDF (term frequency–inverse document
 frequency) 120, 140–142, 144, 179
 tfidf method 142
 threshold-based neighborhoods 47–48
 ThresholdUserNeighborhood class 47
 thrift-based classification server example 332–340
 accessing classifier service 338–340
 running classification server 336–338
 throughput, required 311–312
 time, partitioning by in Shop It To Me case
 study 348
 times(double) operation 364
 times(Matrix) operation 365
 toDataMap() method 37
 TokenFilter class 193
 Tokenizer 193
 tokenizing data
 preparing for 270
 reading and 270–271
 tokenizing features for 20 newsgroups data 268
 tokenStream(String field, Reader r) 192
 Tomcat, configuring JNDI (Java Naming and
 Directory Interface) DataSource in 33

- top-down approach, to clustering algorithms 167
- topic modeling, applications of 182–183
- ToVectorAndPrefReducer class 104
- training classification model 255–280
 - balancing big versus fast 312
 - choosing algorithm 273–276
 - naive Bayes and complementary naive Bayes 275
 - random forests 276
 - SGD 274–275
 - SVM 275
 - classifying 20 newsgroups data set with SGD 265–273
 - paring and tokenizing features for 20 newsgroups data 268
 - previewing data set 266–268
 - training code for 20 newsgroups data 268–273
 - classifying 20 newsgroups data with naive Bayes 276–280
 - data extraction 276–277
 - testing model 278–280
 - training classifier 277–278
 - color-fill classification example
 - choosing learning algorithm 247–250
 - preliminary thinking 246–247
 - converting classifiable data into vectors 260–265
 - feature hashing with Mahout API 261–265
 - representing data as vector 260–261
 - extracting features to build classifier 256–257
 - preprocessing raw data into classifiable data 257–260
 - computational marketing example 258–260
 - transforming raw data 258
- Shop It To Me case study 346–352
 - avoiding target leaks 348
 - defining goal of classification project 346–348
 - feature vector encoding 349–352
 - learning algorithm tweaks 348–349
 - partitioning by time 348
- versus test versus production 234
- work flow for 240–244
 - collecting historical data 240
 - defining categories for target variable 240
 - defining predictor variables 240–241
 - different predictor variables are needed with different data 243
 - position, using as predictor variable 241–242
 - selecting learning algorithm to train model 243–244
 - using learning algorithm to train model 244
- training code for 20 newsgroups data 268–273
 - accessing data files 269–270
 - configuring learning algorithm 269
 - measuring progress so far 272
 - preparing to tokenize data 270
 - reading and tokenizing data 270–271
 - setting up vector encoders 269
 - training SGD model with encoded data 273
 - vectorizing data 271–272
- training data 232–234
 - and scoring 18–19
 - collection 240
 - from logs 326
- training examples
 - implications of number of 311
 - scale 311
- training model to find color-fill, preliminary thinking 246–247
- training pipeline for large systems 313–324
 - acquiring and retaining large-scale data 314–316
 - acquiring data 315
 - partitioning and storing data 315–316
 - working incrementally 316
 - denormalizing and downsampling 316–318
 - full reduce joins 318
 - in-memory joins 317
 - join target first 317
 - merge joins 317
 - reading and encoding data at speed 320–324
 - direct value interface to value encoders 323–324
 - segment bytes 321–323
 - training pitfalls 318–319
 - semantic mismatch during encoding 318–319
 - target leaks 318
- TrainingExample object 302
- trainingPercentage argument 53
- trainlogistic 249
- TrainNewsGroups 336
- transforming raw data 258
- transpose() method 366
- TreeClusteringRecommender class 65, 69
- TreeSet implementation 29
- tuning
 - JVM 359
 - parameters of LDA 179–180
- tuning classifiers 300–306
 - learning algorithm, tuning 305–306
 - problems, tuning 300–304
 - adding new variables, interactions, and derived values 301–304

- tuning classifiers, problems, *tuning (continued)*
 - removing fluff variables 301
 - trying alternative algorithms 304
- tutorials, for clustering 199–202
- Twitter, finding similar users on 211–215
 - avoiding common pitfalls in feature selection 212–215
 - data preprocessing and feature weighting 211–212
- TwitterAnalyzer class 213–214
- TwitterDownloader class 211

U

- ua.base file 24
- unscored outputs 287
- unsupervised learning 230, 238–239
- update files 32, 89
- updates, coordinating model 327–330
- updating recommenders 88–90
- user data for surfacing similar users,
 - clustering 223–224
- user neighborhoods 46
 - fixed-size 46–47
 - threshold-based 47–48
- user vectors
 - computing 96
 - generating 99–100
 - splitting 103
- user-based recommendations 42–43
 - gone right 42–43
 - gone wrong 42
- user-based recommenders 14, 43–48, 73–74
 - algorithm 43–44
 - component interaction in Mahout 17
 - GroupLens data set 45–46
 - user neighborhoods 46
 - fixed-size 46–47
 - threshold-based 47–48
- UserNeighborhood interface 17, 44–45, 64
- users, anonymous 83–85
- users.txt file 108
- UserSimilarity interface 17, 44–45, 48, 53–54, 56, 58–59, 66

V

- values
 - preference
 - ignoring with Tanimoto coefficient 54–55
 - recommender data without 34–40
 - when to ignore 35–36
- variables
 - adding new, to tune classifier 301–304

- fluff, removing 301
- predictor
 - defining 240–241
 - different needed with different data 243
 - four types of values for 236–238
 - position, using as 241–242
- target
 - defining categories for 240
 - versus predictor 234–235
- vector encoders, setting up 269
- vector encoding, optimizing 309
- Vector format 217
- Vector interface 121, 261
- Vector.aggregate() method 204
- Vector.assign() method 204
- Vector.iterateNonZero() method 204
- Vector.iterator() method 204
- VectorAndPrefsWritable class 104
- VectorBenchmarks class 204
- vectorization 257, 260, 265, 276
- vectorizing data, 20 newsgroups data set 271–272
- VectorOrPrefWritable class 103
- VectorReducer class 220
- vectors 362–365
 - advanced methods 364–365
 - computing partial recommendation 104
 - converting classifiable data into 260–265
 - converting Last.fm tags into 219–220
 - document, generation of 192–195
 - feature hashing with Mahout API 261–265
 - implementation 362–363
 - operations 363–364
 - producing recommendations from 106
 - representing data as vector 260–261
 - user
 - computing 96
 - generating 99–100
 - using appropriate representation 203–205
- Vectors file 208
- VectorWritable class 134–135, 175, 366
- Vertica 309
- Vowpal Wabbit 343
- VSM (vector space model) 135, 137

W

- WAR (web archive) file 86–87
- web-enabled recommenders 86–88
 - packaging WAR file 86–87
 - testing deployment 87–88
- weighted distance measure 128
- WeightedDistanceMeasure class 128
- weighting 50–51
 - features 188
 - in SlopeOneRecommender class 61

When class 295
WhitespaceAnalyzer 142, 144
Wikipedia data set 92–94
 benefits and drawbacks of distributing
 computations 93–94
 problems with scale 93
Wikipedia link files 99
Windows shell 7
word-like features, encoding 262–263
word-like value 237, 319
work flow in classification projects 239–245
Writable 134, 366
Writable interface 134, 366
WritableComparable 134
writePointsToFile 122–124
WSDL (Web Services Definition Language)
 file 88

X

XMLInputFormat class 222
-XX: +NewRatio=12 360
-XX:+UseParallelGC –
 XX:+UseParallelOldGC 360
-XX:NewRatio 359

Y

Yahoo! Mail 4–5

Z

ZK file 334, 337
ZooKeeper command 337
ZooKeeper file 332, 335