Index

Symbols and Numerics

`__init__()` function, 353

`__lt__()` function, 353

(`.py`) extension, 11

3D matrix, 214

2D object, 214

1812 Overture, 104

A

accessing data, 45–50

activation function, at neuron level, 126

- hyperbolic tangent, 138–139
- leaky rectified linear unit, 139
- rectified linear unit, 139
- softmax, 139–140

activation phase, 151

AdaBoost algorithm, 110, 111

Adaptive Moment Estimation, 164

advanced program. *See* Python functionality

adversarial machine learning, 193

agentic AI, 309–311

agentic loop, 309, 310

agentic RAG (retrieval-augmented generation), 311–312

`AgentRAG.py`, 317–320

`AGN.py`, 286–287

agriculture, machine vision systems, 203–204

`AIAgent2.py`, 312–316

AI agents

- agentic AI, 309–311
- agentic RAG, 311–312
- AI RAG agent, 316–320
- AI web-scraping agent, 312–316
- limitations, 320–321
- programs in Python, 312–316
- traditional AI, 308–309

AI-focused packages, 9

AI mistakes

- complex or simple models selection, 368
- data exploration and quality checks, 366
- domain experts, 369
- ethical and bias considerations, 370
- evaluation metrics, 366–367
- hyperparameters, 368–369
- model's impact, 369
- model's performance monitoring, 370
- time-series data, 367–368
- training data, 367

COPYRIGHTED MATERIAL

AI projects development
data collection, 361
data exploration, 362
domain experts, 359–360
metrics choices, 362
model choices, 362
outputs, as solve a problem, 363
problem sorting by feasibility and impact, 360
problem types, 360
scope out the project, 361
training, 363
AI RAG agent, 316–320
AlexNets, 235
American Micro Systems, 270
Anaconda, 11
installation, 373–374
packages, 17
virtual environment, 15–16
Anaconda Navigator, 17, 374
application programming interface (API), 62
arguments, 24
artificial intelligence (AI). *See also* AI agents; AI mistakes; AI projects development
definitions, 29–30
expert systems/rule-based approaches, 31
machine learning, 31–33
taxonomy, 30
Artificial Intelligence (Minsky), 270
attention mechanism, 191
autoencoders, 181, 188–190
average pooling, 233

B

backpropagation
activation phase, 151
derivative of the error, 148
error phase, 150–151
example, 149–150
feedback, 311
hidden layer equation, 152–154
information phase, 151–152
learning rate and update step, 156–157
output layer equation, 152
process implementation, 154–156
three calculations, 148–149
`backward()` method, 178
backward propagation. *See* backpropagation
`BaggingClassifier`, 107, 108
bagging ensemble models
data subsets, 106–107
in Python, 108–109
scikit-learn library, 107–108
`bagging_model.py`, 108–109
basic DataFrame information, 44–45
basic program testing, 21–22

batch gradient descent, 146
beginners, 2
bias, in neuron, 126
Bidirectional Encoder Representations from Transformers (BERT), 274
big data revolution, 34
BigScience Large Open-Science Open-Access Multilingual (BLOOM), 275
"black-box" models, 118, 191
block diagram
  generative adversarial network, 285
  variational autoencoder, 291
`blur_image.py`, 220–221
blurry faces, 220–221
`BoostingClassifier`, 114
`boosting_model.py`, 112–113
boosting models
  in Python, 111–113
  types, 110–111
`bootstrap`, 107
`bootstrap_features`, 107
bottleneck, 299
bounding box, 231
box loss, 263
brain
  functions, 123
  infrastructure, 122
  plasticity, 124
Bumps, 43

C

`calc_population_fitness()` function, 354
canny edge detection, 225–227
`categorical_crossentropy` loss function, 171
categorical data, 42
Celsius to Fahrenheit calculation, 79
Center for Intelligent Industrial Robotics (CIIR), 312
characteristics, Python, 9–10
ChatGPT (Chat Generative Pre-Trained Transformer), 267, 274, 285, 302, 304, 312, 313
cheat sheet, 3
classical machine learning models, 32, 40
classification models, 71, 72
class loss, 263
Claude code, 285
`clf()` (clear figure) function, 56
clustering model, 71, 72
`cnn_keras.py`, 236–237
CNNs. *See* convolutional neural networks (CNNs)
coders, 8
coefficient of determination, 81–82
color spaces, 218–219
`color_spaces.py`, 218–219
community support, Python, 9–10

companion files, 4
compiled language, 8, 10
computer scientists, 8
continuous features, 42
Conv2D layer, 238
convolutional neural network (CNN), 32, 183–186
  activation function, 233
  architecture, 183, 232
  convolutional layer, 232
  filters, 183–184, 231
  fully convolutional neural networks, 186
  ImageNet, 184
  in Keras, 235–239
  MNIST dataset, 184, 236
  pooling layer, 233
  transfer learning, 234
  uses, 184
  Yolo, 185
convolutional VAE, 291–297
copy_member() function, 354
correct answers, of model, 37
create_new_generation() function, 354
create_random_string() functions, 353
createTrackbar() function, 227
CrewAI, 321
criterion, 98, 100, 115
cross-entropy loss function, 142
CrossEntropyLoss() function, 178
crossover, 346
customIrisDataset class, 178
CVAT account, 245–247
cvat.ai, 245
cv.destroyAllWindows() function, 208
cv.imread() function, 210
cv.imshow() function, 208, 209
cvtColor() function, 218
cv.VideoCamera() function, 209
cv.waitKey() function, 208, 210

D

data
  accessing and filtering of, 45–50
  basic DataFrame information, 44–45
  collection, AI projects development, 361
  data-driven techniques, 356
  domain knowledge importance, 56–57
  exploration, 44–45, 362, 366
  finding and handling missing data, 50–53
  pandas library installation, 43
  preparation, neural networks, 165–167
  preprocessing hybrid architecture, 63–68, 192–193

and privacy risks, 281
quality checks, 366
transformation, 131–134
variables, 42–43
visualizations, 53–56
data_exploration_filter.py, 45–50
data_exploration_missing_data.py, 51–53
data_exploration_visualize.py, 53–56
DataLoader, 178
data.yaml file, 252–253
DecisionTreeClassifier modules, 97, 98
decision_tree_parameters.py, 98–100
decision_tree.py, 96–98
decision trees
depth, 91
entropy, 93–95
flowcharts, 90–91
Gini index, 95
information theory, 92–93
leaf nodes, 91
nodes, 91
parameters, 98–100
scikit-learn library, 96–98
split of node, 91
decoder, 32
decoder networks, 189–190
deep-learning powerhouses, 9
deep neural network problem, 368
def keyword, 25
dendrites, 122
dependent variable, 78
depth, decision trees, 91
describe() function, 44
developers, 8
Dickinson, John, 270
diffusion models, 298–302
vs GAN, 298
vs VAE, 298
Diffusion.py, 300–301
Dirtiness, 43
distributed focal loss (DFL), 263
diversity, 346
domain experts, 369
AI projects development, 359–360
domain knowledge, 56–57
drop() function, 75
dropna() function, 53
DummyClassifier model, 73–75

E

EarlyStopping callback, 171
edges.py, 226
elitism, 346

embedding layer, 269
encoder networks, 189–190
encoders, 32
encoding outputs, in scikit-learn, 63–65
ensemble models
  accuracy, 116–117
  bagging model, 105–109
  boosting model, 110–113
  description, 104–105
  designing, 105
  overfitting and explainability trade-offs, 117–118
  random forest model, 113–115
entropy, 93–95
environment, 326, 330
equation of a line, 78–79
error, of model, 37
error function, at network level
  cross-entropy loss, 142
  mean average error, 141–142
  mean squared error, 141
  root mean squared error, 141–142
error phase, 150–151
  derivative, 148
error surfaces, gradient descent, 143–145
ethical and bias considerations, 370
`evaluate()` function, 172
evaluation metrics, 366–367
evolutionary computation. *See also* genetic algorithm
  data-driven techniques, 356
  definition, 341
  examples, 340–341
  fit, in environment, 340
  generations, 340
  individual, 340
  mutate, 340
  population, 340
  principles, 340
  runtime performance, 356
  search strategies, 356
  subfields, 342
`expand_dims` NumPy function, 239
experience, 2–3
expert systems/rule-based approaches, 31, 309
exploration, of data, 44–45
extension (`.py`), 11

F

FANUC CRX10 robots, 304
"father" of AI, 270
faults, 43
faults.csv file, 44
`faults_df` DataFrame object, 44
feature engineering, 161
features, of data, 42

feedback, 310–311
reinforcement learning, 325–327
loop, 309
feed-forward networks, 182–183
feed-forward phase, 131–135
file extension (`.py`), 11
file not saved, troubleshooting for, 27
filtering, of data, 45–50
finding and handling missing data, 50–53
`fit()` function, 73, 75, 98, 171
fitness functions, 340, 341, 345, 346
five-epoch training session, 289
`forward()` function, 178
fully connected neural networks, 32
fully convolutional neural networks, 186
function, 25, 73

G

GAN. *See* generative adversarial network (GAN)
gated recurrent unit (GRU) networks, 187
Gaussian blur function, 221
Gemini, 275, 276
generations, 340, 341
generative adversarial network (GAN), 284–290
block diagram, 285
vs diffusion models, 298
minimal GAN program, 286–287
generative AI, 305–306
types of, 283–284
genetic algorithm, 194
basic process, 344
implementation, 348–356
mutation, 347
population initialization, 344–345
population scoring, 345–346
repeat the process, 347–348
search optimization, 342–344
selection mechanism, 346–347
`genetic_algorithm.py`, 348–352
`genetic_algorithm_with_crossover.py`, 355–356
`getTrackbarPos()` function, 227
Gini index, 95
GitHub, 377
global minimum, gradient descent, 145–146
Google AI Studio, 277
Google Colab, 12, 253–254, 375
Gradient Boosting, 110, 111
`GradientBoostingClassifier` model, 111–112, 114

gradient descent, 111
error surfaces, 143–145
local and global minimum, 145–146
search space, 342
types, 146–147

H

hallucinations, 280
`head()` function, 44, 50, 53
`hello_world.py`, 21–22
`h_enc`, 297
hidden layer equation, 152–154
high-resolution images, 218
high-value knowledge work, 273
`hist()` function, 56
Hugging Face Agents, 321
human vision system, 200
aperture, 201–202
color interpretation, 202
depth perception, 202
eyeball, 200
focus, 201
information processing, 202
with machine vision, 201–202
shutter speed, 202
hybrid architectures, 192
hybrid model, 104
hyperbolic tangent (tanh) function, 138–139
hyperparameter, 37, 143
AI mistakes, 368–369
neural networks, 163, 164
hysteresis thresholding, 225

I

`iloc` indexer, 50
`image_gradient.py`, 223–224
ImageNet, 184
independent variable, 78
individual, in biological evolution, 340, 341
individual class, 353
inference engines, 309
information gain, 94
information phase, 151–152
information theory, 92–93
input data, of model, 35, 36
Input text, 268
instances, of data, 42
instance segmentation, 231
integer features, 42
integrated development environment (IDE), 12–13, 375–376
interface, 273
internal parameters, of model, 37
interpreted language, 8, 10

Iris dataset, 165–167
isna() properties, 52
Isolation Forests, 105
iterator, 25

J

Jagger, Mick, 270
Jupyter Notebooks, 375

K

k1, 297
Keras, 9, 235–239
  neural networks in, 167–172
kernel operations, 222–223
knowledge representations, 309
kraken, evolution of, 340–341
K-Scatch, 43

L

LangChain, 321
large language model (LLM), 267, 307, 324
  AI RAG agent, 317, 319
  AI web-scraping agent, 316
  architectures, 268–270
  feedback, 311
  limitations, 280–281
  LLM-written code, 304
  training, 279
  transformer, 271–272
  via API, 274–279
Large Language Model Meta AI (LLAMA), 275
lasso regression, 88
latent space, 188, 189, 291
layer feature hierarchy, 128–129
leaf nodes, 91
leaky rectified linear unit (leaky ReLU), 139
learning, reinforcement, 324–325
learning algorithm, 326
learning capacity, 129
learning rate, 156–157
len() (length) function, 50, 53
libraries/packages, 8, 9
linear regression, 36
  Celsius to Fahrenheit calculation, 79
  coefficient of determination, 81–82
  equation of a line, 78–79
  logistic regression, 88
  modeling linear regression, 79–80
  multiple regression, 87–88
  ordinary least squares, 80–81
  polynomial regression, 88
  in Python, 82–84
  ridge and lasso regression, 88
  in scikit-learn, 85–87

linear_ regression_ library.py, 85–87
linear_regression_ manual.py, 82–84
linear_regression_model, 87
LinearRegression modules, 86
LineWorld class, 334–335
Linux, 12, 371, 374
list() function, 44
list_of_ numbers, 24
live_test_object_detector.py, 260
LLM. *See* large language models (LLMs)
load_iris() function, 167
local maximum, 325
local minimum, gradient descent, 145–146
loc indexer, 48–50
logic-based programming language, 31
logistic regression, 88
long short-term memory (LSTM), 187, 271
loop, 25
loss, input code, 297
LSTM. *See* long short-term memory (LSTM)

M

machine learning, 31
  big data revolution, 34
  classical machine learning models, 32
  development, problem, 284
  libraries, 9
  neural network models, 32–33
  reinforcement learning, 33, 324–325
machine vision, 199
  aperture, 201–202
  color interpretation, 202
  coverage, 205
  data collection, 204
  data privacy and permission, 206
  dataset bias, 206
  depth perception, 202
  ethical concerns, 205–206
  focus, 201
  hierarchy, 230
  human vision with, 201–202
  image classification, 231
  information processing, 202
  instance segmentation, 231
  lighting, 205
  object detection, 231
  segmentation, 231
  shutter speed, 202
  system use, 206
  tasks, 231
  uses, 203–204
Mac system, 12, 15, 371, 374
MAE function. *See* mean average error (MAE)

manufacturing, machine vision, 203
Markov chain model, 33, 327–328
Matplotlib, 53, 86, 171, 330
`max_depth`, 98, 100, 115
`max_features`, 99, 107, 115
`max_leaf_nodes`, 99, 115
max pooling, 233
`max_samples`, 107
mean average error (MAE), 141–142
mean squared error (MSE), 141, 366
Median blurring, 221
medical imaging, machine vision systems, 204
Metcalf, Robert, 270
metrics choices, 362
Microsoft AutoGen, 321
Microsoft Visual Studio Code (VS Code). *See* VS Code interface
mimic brain architectures, 196
mini-batch gradient descent, 147
`MinMaxScaler` module, 66–68
`MinMaxScaler()` normalizer, 167
`min_samples_leaf`, 99, 100, 115
`min_samples_split`, 99, 115
Minsky, Marvin, 270, 308
MINST data, 286–287
missing data, finding and handling of, 50–53
MNIST library, 285
model
  API on `DummyClassifier`, 73–75
  choices, 362
  correct answers, 37
  error, 37
  evaluation, 38–39
  impact, 369
  improvement, 39
  input data, 35, 36
  internal parameters, 37
  output data, 35, 37
  performance monitoring, 370
  selection, 368
  training process, 38
modeling linear regression, 79–80
`model_selection` module, 69, 109
modules, 8
Monte Carlo simulations, 33
MSE function. *See* mean squared error (MSE)
multiple regression, 87–88
multiplicative coefficient, 78
`mutate()` function, 353
`mutate_population()` function, 354
mutation, 340, 341, 347
mutation rate, 347

N

NaN (Not a Number) value, 50, 52, 160
neural architecture search (NAS), 164, 345

neural connections
biological concepts, 124
computational model, 129–131
neural network, 181
activation function, 126, 138–140
architectures, 163–164
autoencoders, 188–190
backpropagation, 147–157
basic feed-forward networks, 182–183
bias, 126
categories, 182
components, 125
connections, 129–131
convolutional layers, 183–186, 191–192
data pitfalls, 160–161
data preprocessing, 192–193
dataset preparation, 165–167
data transformation, 131–134
encoder and decoder layers, 192
ensemble models, 194
error function, 140–142
feedback loops, 195–196
feed-forward phase, 131–135
gradient descent, 143–147
grouping neurons together, 126–128
hybrid architectures, 192
hyperparameters, 163
hyperparameter tuning techniques, 164
in Keras, 167–172
layer feature hierarchy, 128–129
learning capacity, 129
models, 32–33, 40
overfitting and underfitting, 162
parameters, 125–126, 163
parameters optimization, 194–195
in PyTorch, 172–179
recurrent neural networks, 186–188
sequential models, 193–194
three-layer neural network, 126–127
transformers, 190–191
validating models, 161–162
weights, 125
`NeuralNetwork()` class, 178
`neural_network_data_prep.py`, 165–167
`neural_network_keras.py`, 168–170
`neural_network_pytorch.py`, 173–176
neural networks, biological concepts
neural connections, 124
neuron, 122
sensory architecture, 123–124
neuron
biological concepts, 122–124
computational model, 125–128

new_num, 25
node, 91, 126, 171
nondeterministic polynomial time (NP), 343
non-object image, 243, 244
notation, 135
nothing function, 227
num argument, 25
numeric data, 42
NumPy, 330

O

object detector
- annotation job, 248–250
- create steps, 241
- dataset collection, 242–243
- data.yaml file, 252–253
- download labels, 250–251
- free CVAT account, 245–247
- Google Colab, 253–254
- model training program, 255–259
- organize your images, 244–245
- real time, 260–263
- rinse and repeat, 251–252
- runtime, 254–255
- from scratch, 241
- troubleshooting, 259–260

object_detector_program.py, 255–256
OneHotEncoder() object, 167
one-hot encoding in scikit-learn, 64–66
one hundred-epoch training session, 290
OpenAI, 274, 275, 312, 315
Open Source Computer Vision (OpenCV)
- blurry faces, 220–221
- canny edge detection, 225–227
- color spaces, 218–219
- description, 206–207
- image gradient with, 223–225
- kernel operations, 222–223
- pixel manipulation in, 215–217
- processing techniques, 227–228
- reading and displaying, 207–208
- troubleshooting toolchain, 210–211
- webcam with, 209–210

ordinary least squares, 80–81
output data, of model, 35, 37
output layer, 269
output layer equation, 152
overfit, neural networks, 162

P

packages, 8
- installation, 16–17
- managers installation, 373–375

pandas library
accessing and filtering of data, 46–47
data exploration, 44
data visualizations, 54
finding and handling missing data, 51
installation, 43
panel data, 43
parameters, 98–100, 125–126, 143
neural networks, 163
particle swarm optimization (PSO), 342
pasting subsets, 106
Pastry fault, 43, 56
perception-action loop, 309
pitfalls, in program writing. *See* program writing pitfalls
pixel, 214
`Pixels_Areas` feature, 66
`pixels.py`, 215
planning algorithms, 309
`plot()` function, 56, 172
`plot_results` function, 335
`plot_tree()` function, 98, 100
`plot_tree()` modules, 97
plt library, 56
`PolicyGradientAgent` class, 335
policy gradient (PG) methods, 329
policy optimization, 33
polynomial regression, 88
polynomial time, 343
population, 340, 341
converges, 347
diversity, 346
initialization, 344–345
scoring, 345–346
size, 344
PowerShell terminal, 20
`predicted_sales` variable, 84
`predict()` function, 73, 75
`predict_proba` function, 73
`price_format` variables, 87
`price` (the independent variable), 83
principal component analysis (PCA), 88
`print_population()` function, 353
`print_self()` function, 353
problem types, 360
programmers, 8
programming components
integrated development environment, 12–13
Python interpreter, 11
terminal, 11–12
text editor, 11
programming languages, 8
programs/software, 8
program writing pitfalls
installing python packages, 16–17
terminal, understanding, 13–15
virtual environment, 15–16

projects, learn and try, 2
prompt sensitivity, 281
PyCharm, 12
`pyplot show()` function, 98
Python
  agentic AI program, 312–316
  AI-focused packages, 9
  bagging ensemble models, 108–109
  boosting models, 111–113
  characteristics, 9–10
  description, 8–9
  linear regression, 82–84
  program, 11
  random forests model, 114–115
  reinforcement learning, 330–338
  script, 11
  simplicity, 9
  support, high community, 9–10
  tips for safe programming, 26
  troubleshooting, 27
Python Debugger, 377
Python functionality
  access a function, 24
  calls the function, 25
  commands for `more.py`, 22–23
  condition-checking, 24–25
  creates a *loop,* 25
  creating a function, 22
  defines a *function,* 25
  defines a list, 24
  `#Import a library`, 23
  `import random`, 23–24
  terminal output, 23
Python interpreter, 11
Python interpreter installation
  browser-based, 373
  natively on your system, 372
  virtual environment manager, 372–373
Python Software Foundation, 8
Python terminal, troubleshooting for trapped in, 27
PyTorch, 9
  neural networks in, 160, 164, 172–179

Q

Q-learning, 33

R

`randint` function, 24
random, 23
random forest(s)
  parameters, 115
  in Python, 114–115
  scikit-learn library, 113–114
`RandomForestClassifier` class, 114, 115

`random` library, 24
random number, program, 22–25
`random_number`, 24
random patches, 107
random subspaces, 106
`range` function, 25
`read_csv()` function, 44
`read_display_image.py`, 207
`read_in_dataset` function, 65, 75, 98, 109
`recon`, reconstruction loss, 297
rectangle tool, 248
rectified linear unit (ReLU) function, 139
recurrent neural networks (RNNs), 32, 186–188
  architecture, 187
  sequences of data, 187
  time-series data, 186–188
regression models, 71, 72, 77. *See also* linear regression
regular neural network layer, 234
reinforcement learning (RL), 32, 33, 194
  feedback, 325–327
  learning, 324–325
  machine learning, 324–325
  Markov chain decision parameters, 327–328
  policy gradient (PG) methods, 329
  in Python, 330–338
Reinforcement Learning Using a Policy Gradient (RL.py), 330–335
ReLU function. *See* rectified linear unit (ReLU) function
`reset_index()` function, 53
`residual_squares` variable, 84
residual sum of squares, 80
ResNets, 235
`returned_number`, 25
reward function, 326
ridge regression, 88
RL. *See* reinforcement learning (RL)
RMSE function. *See* root mean squared error (RMSE)
RNNs. *See* recurrent neural networks (RNNs)
`robotics_relevance_score` function, 314
Robot Operating System 2 (Ros2), 321
root mean squared error (RMSE), 141–142
rule-based approaches, 31
rule-based systems based on if-then statements, 309
`run_fitness_ function()` functions, 353
`run_generation()` function, 354, 355
runtime performance, 356

S

saccades, 202
`sales_format` variables, 87

sales (the dependent variable), 83
sampleChatGPT.py, 275
sampleGemini2.py, 277–278
scaling data, 66–68
scatter() function, 56
scikit-learn library, 9
  bagging ensemble models, 107–108
  classification models, 71, 72
  clustering model, 71, 72
  data preprocessing, 63–68
  decision trees, 96–98
  functions, 73
  installation, 63
  linear regression, 85–87
  model API on DummyClassifier, 73–75
  neural networks, 171
  random forests model, 113–114
  regression model, 71, 72
  submodules, 62
  training and test sets, 68–71
  validation sets, 68, 69
score() function, 73, 75, 98
search
  optimization, 342–344
  searching methods, 309
  strategies, 356
selection mechanism, genetic algorithm, 346–347
sensory architecture, 123–124
sensory system, 123
sequential model architecture, 193–194
Sequential model type, 171
Shovic, John, 270
show() function, 56
sigmoid activation function, 133
simplicity, of Python, 9
skip connections, 299
sklearn.datasets, 62, 69
sklearn.dummy, 62, 73
sklearn.ensemble, 62
sklearn.linear_model, 62
sklearn.metrics, 62
sklearn.model_selection, 62
sklearn.preprocessing, 62, 63
sklearn.tree, 62
sklearn.utils, 62
slope coefficient, 78
slope-intercept, 78
softmax, 139–140
software, 8
software engineers, 8
spiking neural networks, 32
split of node, 91
Spyder, 12
Stacking Classifiers and Regressors, 105

Stains, 43
Steel Plates Faults dataset, 41–42
`step()` method, 178
stochastic gradient descent, 147
`string_list` variable, 353
`sum_x` and `sum_y`, 83
`sum_x_squareds`, 84
`sum_x_times_y`, 84
`superclass train()`, 178
supervised learning, 324, 325
supervised machine learning, 78
suppressing non-maximum pixels, 225
synapses, 122
syntax, 8, 9
syntax errors, troubleshooting for, 27

T

tabular dataset, 42
tanh function. *See* hyperbolic tangent (tanh) function
Tchaikovsky, Pyotr Ilyich, 104
ten-epoch training session, 290
terminal, 11–12
  problems and troubleshooting, 13–15
`test()` function, 178, 179
test sets, 68–71
`test_set size` function, 71
text editor, 11
three-layer neural network, 126–127
time-series data, 367–368
`toarray()` function, 66
tokenizer, 269
`tolist()` property and function, 47
`torch.no_grad()` method, 178
`total_ loss and total_correct` variables, 178
`total_squares` variable, 84
tournament selection, 346
`tournament_selection()` function, 354
`tournament_selection_with_crossover()` function, 354
trackbars, 225, 227
traditional AI, 308–309
`train()` function, 178, 179
training data, 367
training sets, 68–71
`train_set` function, 71, 75
`train_test_split` class, 109
`train_test_split()` function, 69–71, 75, 97, 98, 167
`train_test_split` modules, 86
transfer learning, 234, 235
transformers, 32, 271–272
  blocks, 269
  model, 190–191, 302–304
`trees.jpg`, 215–216
trial-and-error process, 39
troubleshooting, 27

U

UC Irvine Machine Learning Repository, 41

`ultralytics`, 256

underfitting, neural networks, 162

U-Nets, 299–300

unit, 126

unsupervised learning, 324, 325

V

validating models, neural networks, 161–162

validation sets, 68, 69

`value_counts()` properties, 52

value function, 326

variables, 42–43

variational autoencoder (VAE), 291–297

- block diagram, 291
- typical architecture, 291

VGG-16, 235

VGGNets, 235

virtual environment, 11

- creating a new, 17–18
- pitfalls in, 15–16

visualizations, of data, 53–56

VLSI (very large-scale integrated circuit), 270

Voting Classifiers and Regressors, 105

VS Code interface

- extension installation, 376–377
- new file creation, 18–19
- open a terminal, 20–21
- python interpreter selection, 19–20
- troubleshooting, 27

W

webcam, 209–210

`webcam.py`, 209

web-scraping agent, AI, 312–316

weights, as neural input, 125

“white-box” model, 118

Windows system, 12, 371, 374

X

`x_mean and y_mean`, 83

`X_Minimum_series`, 47

Y

You Only Look Once (Yolo), 185, 235, 263

Z

Z_Scratch, 43