Chapter 4 Pt. 1 ANSWER KEY

Fundamentals of Machine Learning

1)

Step 1. (B) Defining the problem and assembling a dataset

Step 2. (D) Choosing a measure of success

Step 3. (A) Deciding on an evaluation protocol

Step 4. (E) Preparing your data

Step 5. (C) Developing a model that does better than a baseline

Step 6. (G) Scaling up: developing a model that overfits

Step 7. (F) Regularizing your model and tuning your hyperparameters

2) (94) b. Supervised learning
3) (94) c. Targets
4) (94) d. All the above
5) (94) b. Supervised learning
6) (94)

B. Dimensionality reduction & Clustering → Unsupervised learning

A. Classification & Regression → Supervised learning

7) (95-96)

Sample or input	j) One data point that goes into your model.
Prediction or output	i) What comes out of your model.
Target	f) The truth. What your model should ideally have predicted, according to an external source of data.
Prediction error or loss value	m) A measure of the distance between your model's prediction and the target.
Classes	h) A set of possible labels to choose from in a classification problem.
Label	d) A specific instance of a class annotation in a classification problem.
Ground-truth or annotations	g) All targets for a dataset, typically collected by humans.
Binary classification	b) A classification task where each input sample should be categorized into two exclusive categories.
Multiclass classification	l) A classification task where each input sample should be categorized into more than two categories.
Multilabel classification	e) A classification task where each input sample can be assigned multiple labels. The number of labels per image is usually variable.

Scalar regression	k) A task where the target is a continuous scalar value.
Vector regression	a) A task where the target is a set of continuous values.
Mini-batch or batch	c) A small set of samples (typically between 8 and 128) that are processed simultaneously by the model. The number of samples is often a power of 2, to facilitate memory allocation on GPU.

8) (97) a. Achieve models that generalize
9) (97) c. Training, validation, and test sets
10) (97) c. Hyperparameters are the number and size of layers and parameters are the weights
11) (97) b. Overfitting
12) (97) b. Tuning hyperparameters causes information from the validation data to leak into the model
13) (98) d. Simple hold-out validation, K-fold validation, and iterated K-fold validation with shuffling
14) (99-100) K-fold cross-validation

```
k = 4
num_validation_samples = len(data) // k

np.random.shuffle(data)

validation_scores = []
for fold in range(k):
    validation_data = data[num_validation_samples * fold:
     num_validation_samples * (fold + 1)]
    training_data = data[:num_validation_samples * fold] +
     data[num_validation_samples * (fold + 1):]

    model = get_model()
    model.train(training_data)
    validation_score = model.evaluate(validation_data)
    validation_scores.append(validation_score)

validation_score = np.average(validation_scores)

model = get_model()
model.train(data)
test_score = model.evaluate(test_data)
```

Selects the validation-data partition

Uses the remainder of the data as training data. Note that the + operator is list concatenation, not summation.

Creates a brand-new instance of the model (untrained)

Validation score: average of the validation scores of the k folds

Trains the final model on all non-test data available

15) (98) Simple hold-out (or Hold-out) validation

```
num_validation_samples = 10000

np.random.shuffle(data)

validation_data = data[:num_validation_samples]
data = data[num_validation_samples:]

training_data = data[:]

model = get_model()
model.train(training_data)
validation_score = model.evaluate(validation_data)

# At this point you can tune your model,
# retrain it, evaluate it, tune it again...

model = get_model()
model.train(np.concatenate([training_data,
                            validation_data]))
test_score = model.evaluate(test_data)
```

Shuffling the data is usually appropriate.

Defines the validation set

Defines the training set

Trains a model on the training data, and evaluates it on the validation data

Once you've tuned your hyperparameters, it's common to train your final model from scratch on all non-test data available.