Advanced Data Analysis

DATA 71200

Class 5

Schedule

8-Jun	**Evaluation Methods**
9-Jun	Async: DataCamp
13-Jun	Supervised Learning (k-Nearest Neighbors and Linear Models)
14-Jun	Async: DataCamp

Project 1

Curation and cleaning of a labeled data set that you will use for the supervised and unsupervised learning tasks in project 2 and 3. The dataset can be built from existing data and should be stored in your GitHub repository.

To **submit** the assignment submit a link to your project Jupyter notebook on Blackboard

The **goal** for this assignment is for you to create a usable dataset from an open-source data collection. You will use your curated dataset for a supervised classification task in Project 2 and an unsupervised learning task in Project 3.

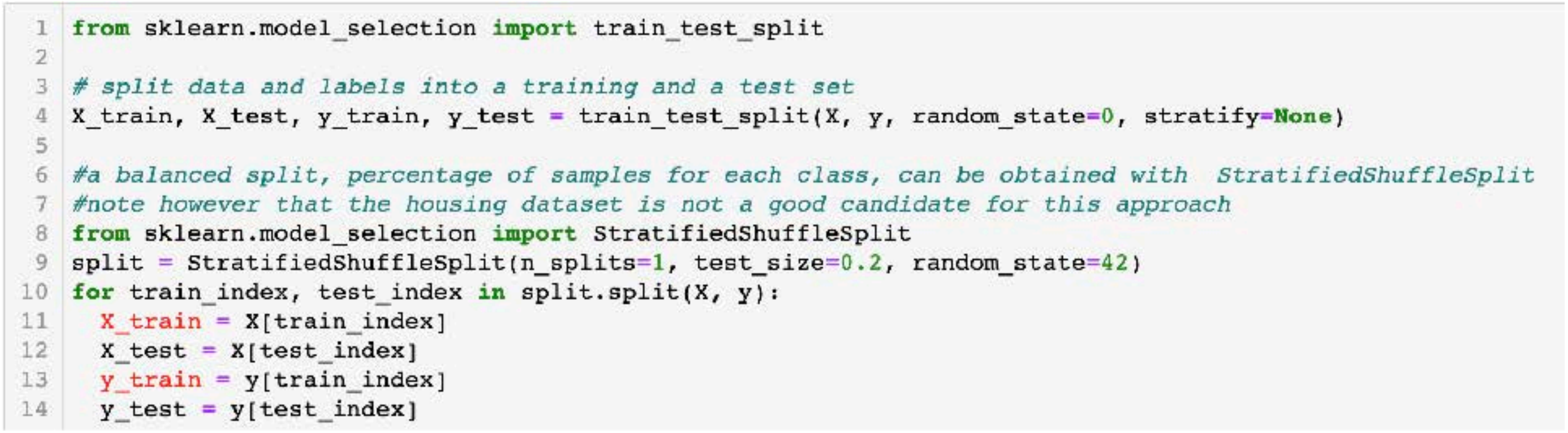
Step 1: Find and download a dataset. Here are some potential places to look

- Amazon's AWS datasets: https://aws.amazon.com/opendata/public-datasets/
- Data Portals: http://dataportals.org/
- Kaggle datasets: http://kaggle.com
- NYPL digitizations: http://libguides.nypl.org/eresources
- NYC Open Data: http://opendata.cityofnewyork.us/data/
- Open Data Monitor: http://opendatamonitor.eu/
- QuandDL: http://quandl.com/
- UC Irvine Machine Learning Repository: https://archive.ics.uci.edu/ml/index.php

Some guidelines regarding dataset selection:

- Since this data is going to be used for supervised learning, one of its features should be a value that you can predict (e.g., if you chose a real estate dataset and are interested in housing prices, then housing prices for each sample should be available in the dataset)
- Your dataset should be large enough to perform supervised and unsupervised learning on. A useful rule of thumb is that you should have at least 10 samples per dimension or parameter/feature you are fitting.

Step 2: Divide into a training set and a testing set. In a Jupyter notebook, use scikitlearn to divide your data into training and testing sets. Make sure that the testing and training sets are balanced in terms of target classes



```python
from sklearn.model_selection import train_test_split

# split data and labels into a training and a test set
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, stratify=None)

#a balanced split, percentage of samples for each class, can be obtained with  StratifiedShuffleSplit
#note however that the housing dataset is not a good candidate for this approach
from sklearn.model_selection import StratifiedShuffleSplit
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(X, y):
  X_train = X[train_index]
  X_test = X[test_index]
  y_train = y[train_index]
  y_test = y[test_index]
```

data71200class5.ipynb

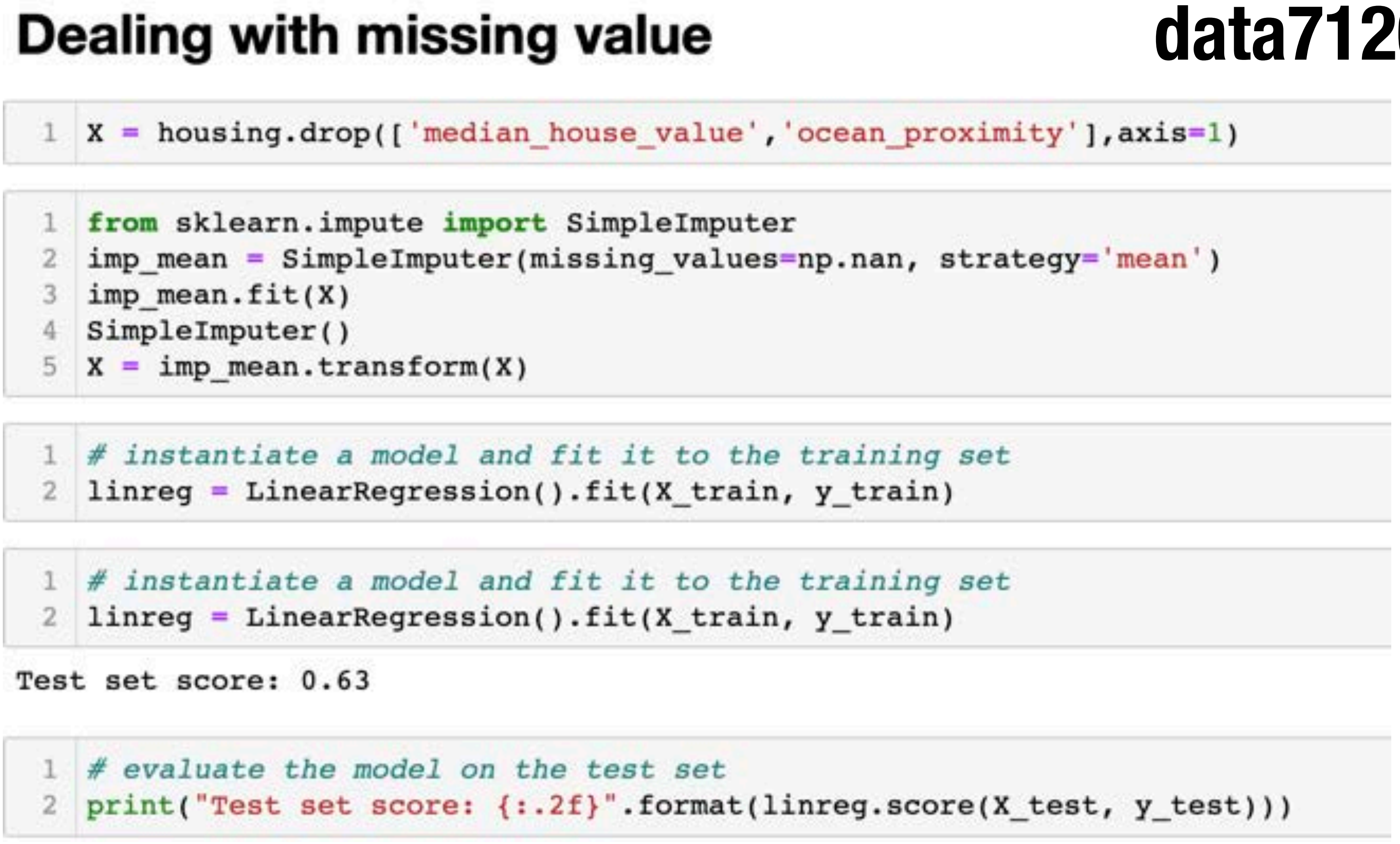
Step 3: Explore your training set. In a Jupyter notebook, import your data into a Pandas data frame and use the following pandas functions to explore your data

- DataFrame.info()
- DataFrame.describe()

Step 4: Data cleaning. Address any missing values in your training set. Include the code in your Jupyter notebook and create a second, cleaned, version of your dataset. Then apply the same procedure to your test set (if you are putting in replacement values use IMPUTER in scikitlearn).

- Recall from the *Hands on Machine Learning* book (p. 60) that some options are
    - "Get rid of the corresponding samples."
    - "Get rid of the whole attribute (column)."
    - "Set the values to some value (zero, the mean, the median, etc.)."

data71200class5.ipynb



Dealing with missing value

```
X = housing.drop(['median_house_value','ocean_proximity'],axis=1)
```

```
from sklearn.impute import SimpleImputer
imp_mean = SimpleImputer(missing_values=np.nan, strategy='mean')
imp_mean.fit(X)
SimpleImputer()
X = imp_mean.transform(X)
```

```
# instantiate a model and fit it to the training set
linreg = LinearRegression().fit(X_train, y_train)
```

```
# instantiate a model and fit it to the training set
linreg = LinearRegression().fit(X_train, y_train)
```

```
Test set score: 0.63
```

```
# evaluate the model on the test set
print("Test set score: {:.2f}".format(linreg.score(X_test, y_test)))
```

If you want to see which feature has the missing element

```
# you can check each one individually with the following code
housing['longitude'].isnull().values.any()
```

```
False
```

```
housing['latitude'].isnull().values.any()
```

```
False
```

```
housing['housing_median_age'].isnull().values.any()
```

```
False
```

```
housing['total_rooms'].isnull().values.any()
```

```
False
```

```
housing['total_bedrooms'].isnull().values.any()
```

```
True
```

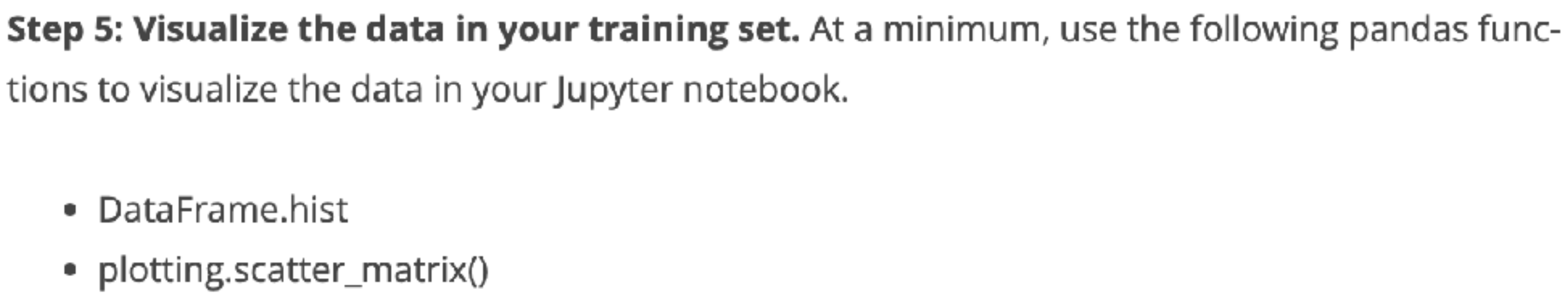
```
# although this isn't necessary for running Imputer it may be useful to know where exactly the data is missing
# you can also check how many elements are empty
housing['total_bedrooms'].isnull().values.sum()
```

```
207
```

```
# or you can generally check if you have any empty elements in your dataframe
#(and thus whether you need to run Imputer)
housing.isnull().values.any()
```

```
True
```

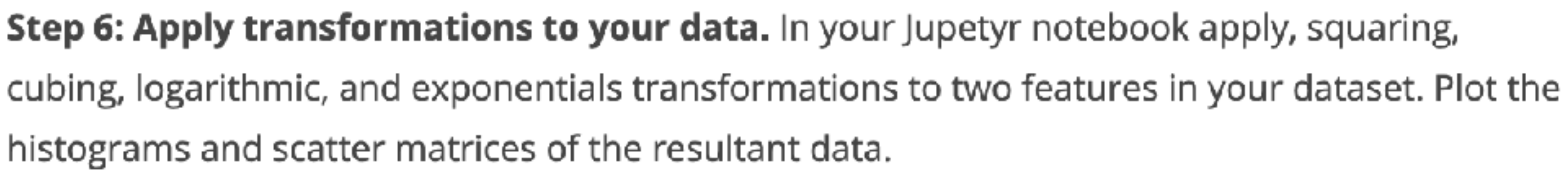
data71200class5.ipynb



Step 5: Visualize the data in your training set. At a minimum, use the following pandas functions to visualize the data in your Jupyter notebook.

- DataFrame.hist
- plotting.scatter_matrix()

data71200class3lab.ipynb



Step 6: Apply transformations to your data. In your Jupetyr notebook apply, squaring, cubing, logarithmic, and exponentials transformations to two features in your dataset. Plot the histograms and scatter matrices of the resultant data.

data71200class4lab.ipynb

Splitting the Data

- **Cross-validation**
  - Splitting the data into folds and iteratively switching up which fold is used for testing (while the remainder are used for training)
  - The data can be split up in a variety of ways

Splitting the Data

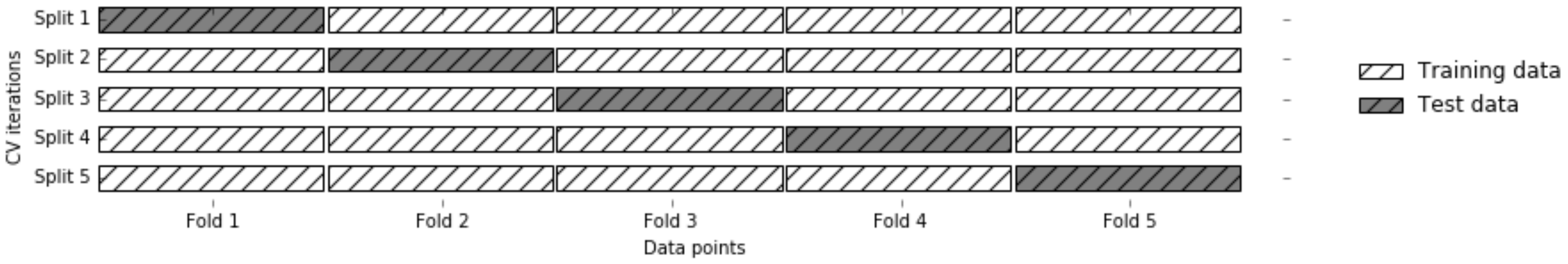


Figure 5-1. Data splitting in five-fold cross-validation

This will create issues if the data is ordered by class since some classes will appear disproportionately (or not at all) in either the testing or training sets

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Splitting the Data

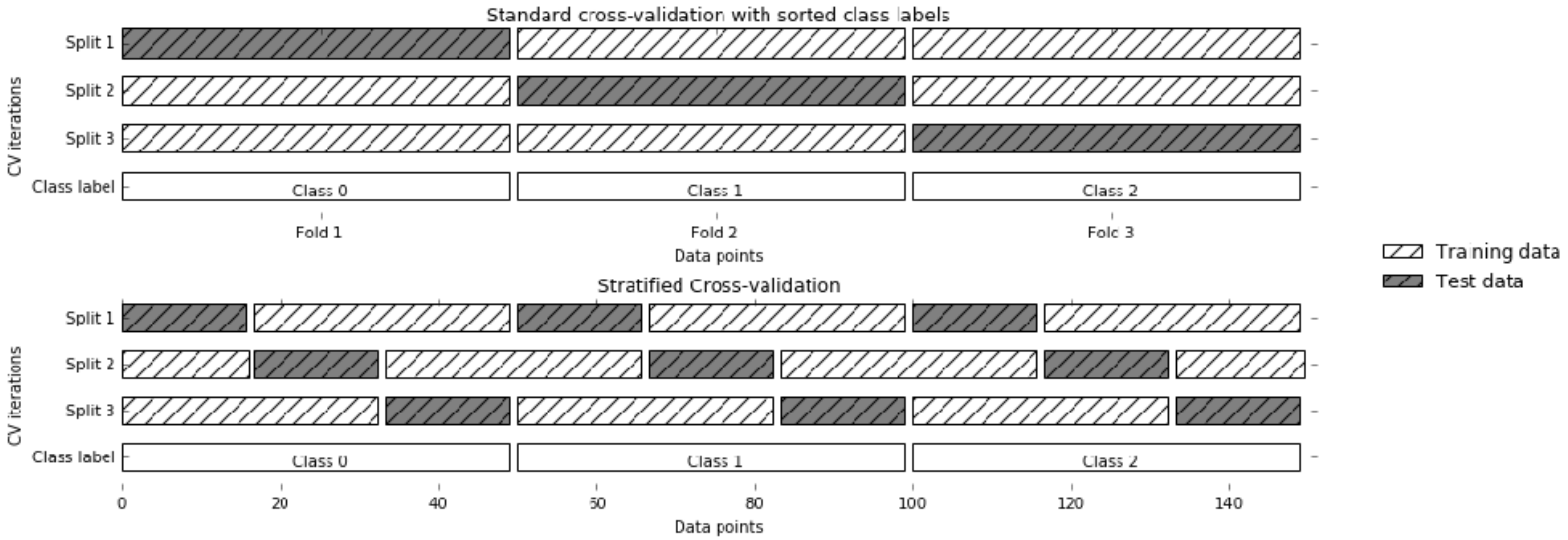


Figure 5-2. Comparison of standard cross-validation and stratified cross-validation when the data is ordered by class label

Stratified Cross-validation explicitly takes data from each class in order to count the issue of over-/under-sampling that can arise in standard cross-validation

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Splitting the Data

Small Data Sets

Leave-one out cross validation can be used where each fold is a single sample

Large Data Sets

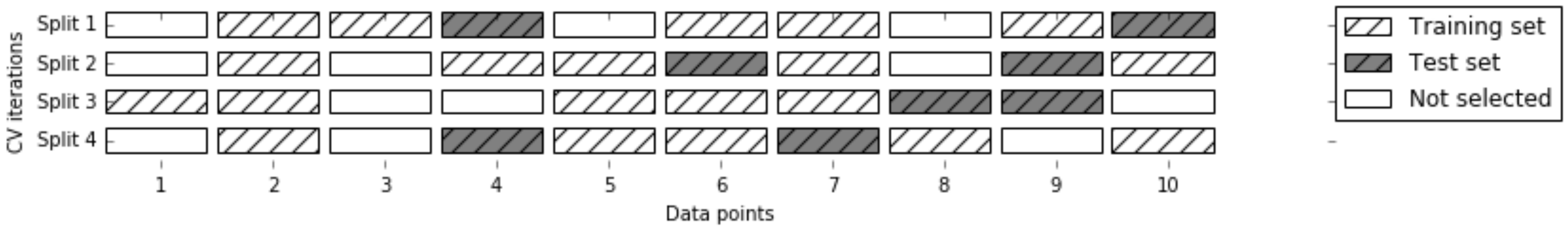


Figure 5-3. ShuffleSplit with 10 points, train_size=5, test_size=2, and n_splits=4

Shuffle-Split provides a system way to sample from your dataset without using all of the data

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Splitting the Data

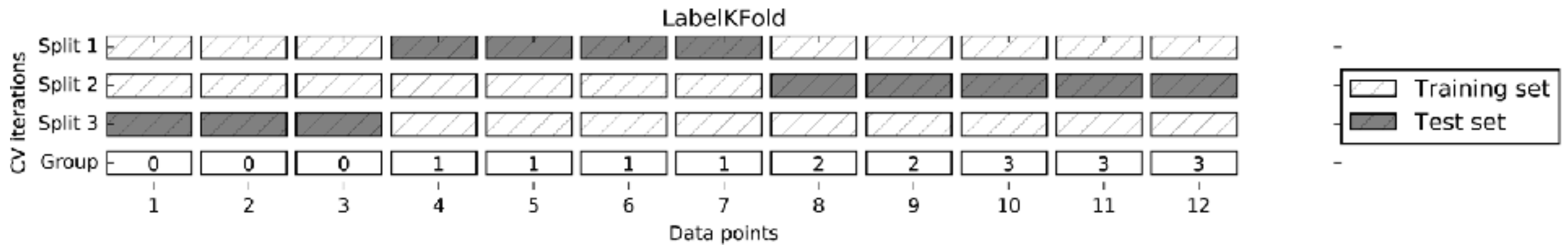


Figure 5-4. Label-dependent splitting with GroupKFold

When groups are highly related (e.g., individual speakers in speech recognition) you may not want to train and test on a single group

GroupKFold allows you select training and testing sets that either include or exclude an entire group

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Splitting the Data

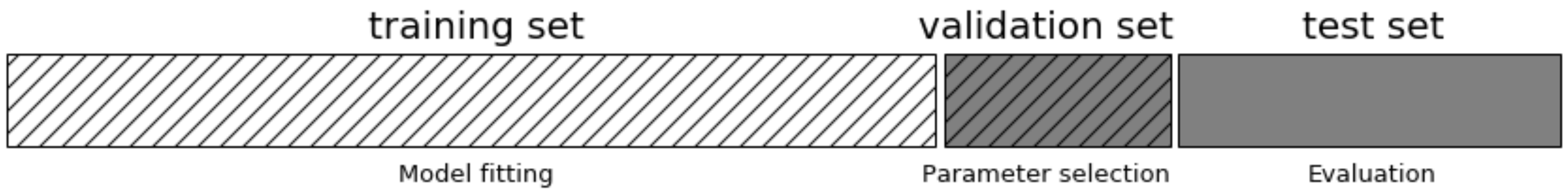


Figure 5-5. A threefold split of data into training set, validation set, and test set

If you are evaluating different models or tuning parameters (e.g., using Grid Search), you will want to have a reserved test set and then split the remaining data into training and validation sets

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Grid Search

Grid search can be performed simply (one split) or iteratively (cross-validation) - in the latter the optimal parameter values need to be summarized

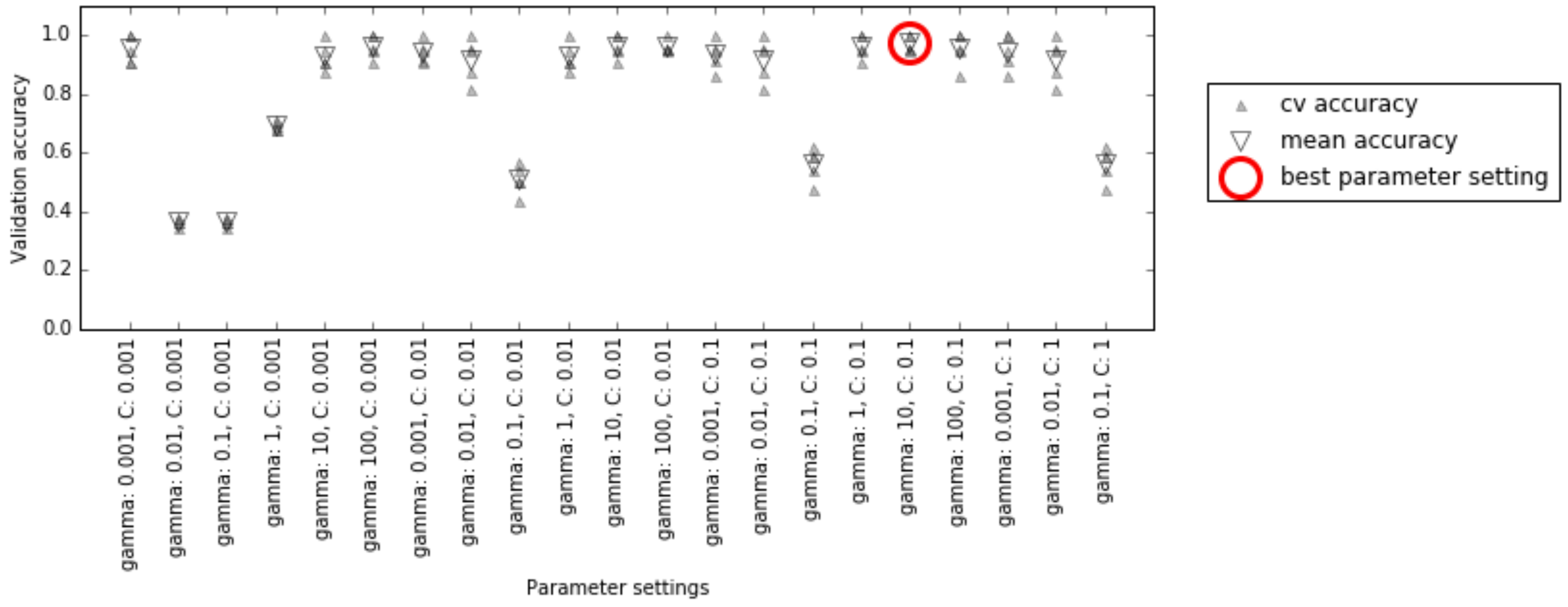


Figure 5-6. Results of grid search with cross-validation

Grid Search

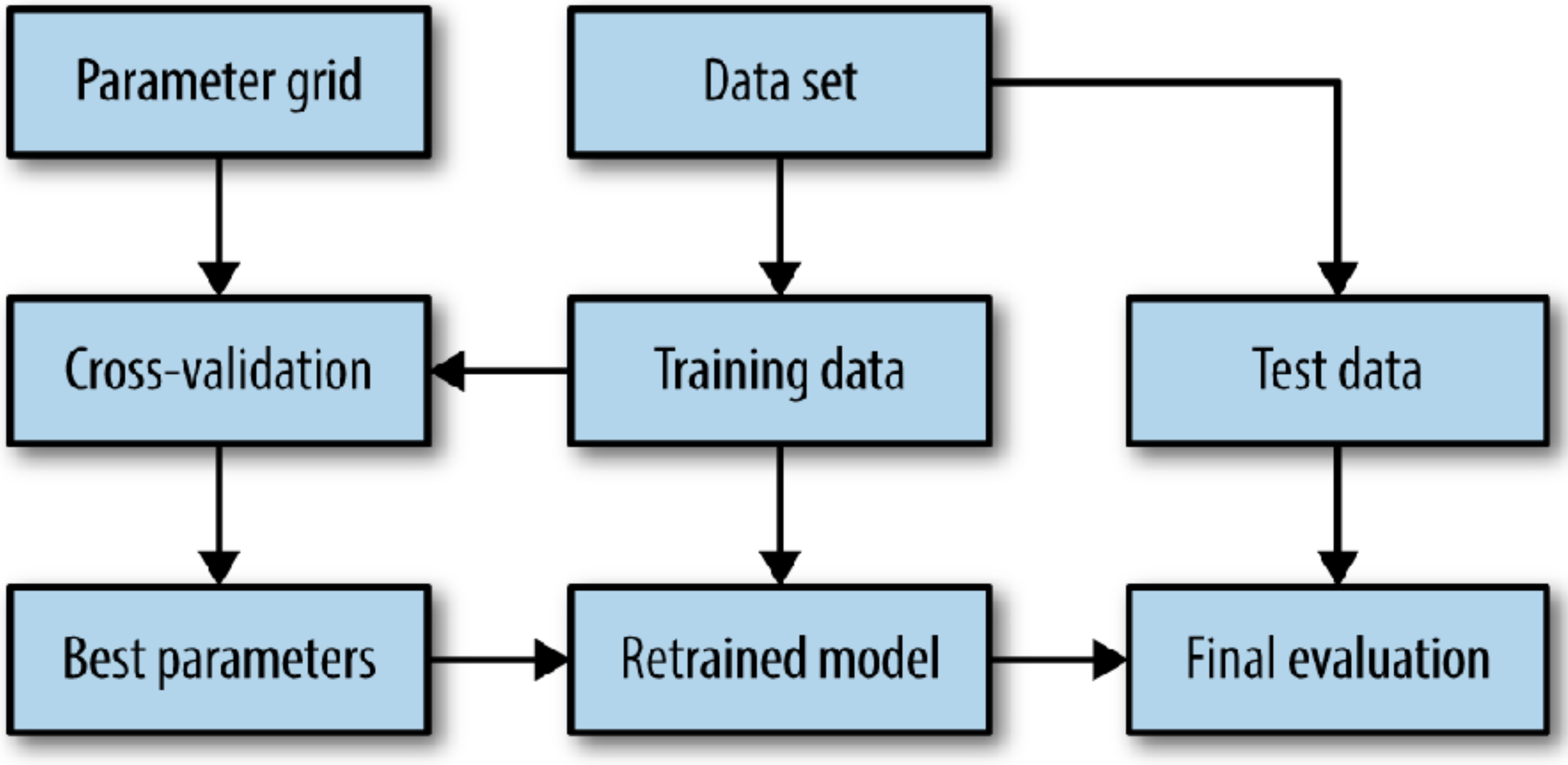
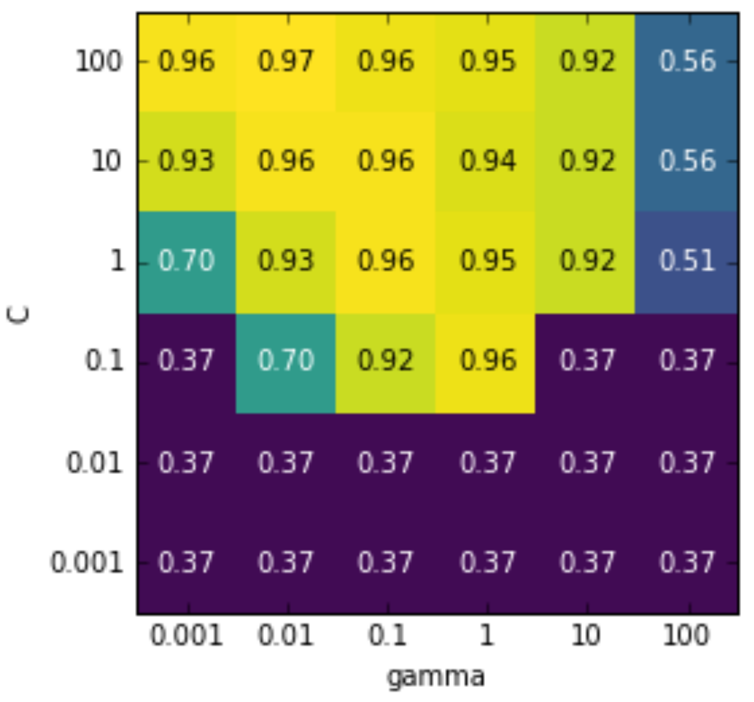


Figure 5-7. Overview of the process of parameter selection and model evaluation with GridSearchCV

Grid Search

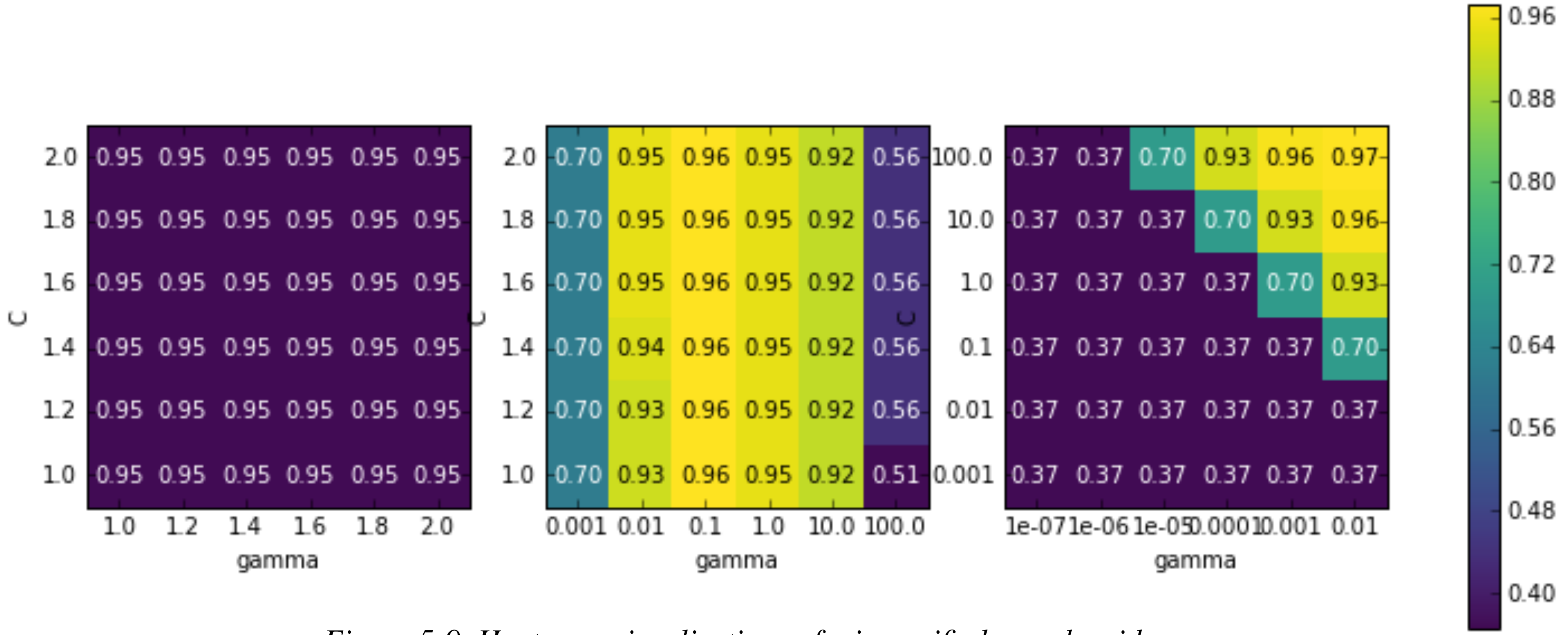
It is important than the parameter range reached is appropriate

Heat maps can help understand whether the range is appropriate



C \ gamma	0.001	0.01	0.1	1	10	100
100	0.96	0.97	0.96	0.95	0.92	0.56
10	0.93	0.96	0.96	0.94	0.92	0.56
1	0.70	0.93	0.96	0.95	0.92	0.51
0.1	0.37	0.70	0.92	0.96	0.37	0.37
0.01	0.37	0.37	0.37	0.37	0.37	0.37
0.001	0.37	0.37	0.37	0.37	0.37	0.37

Figure 5-8. Heat map of mean cross-validation score as a function of C and gamma



C \ gamma	1.0	1.2	1.4	1.6	1.8	2.0
2.0	0.95	0.95	0.95	0.95	0.95	0.95
1.8	0.95	0.95	0.95	0.95	0.95	0.95
1.6	0.95	0.95	0.95	0.95	0.95	0.95
1.4	0.95	0.95	0.95	0.95	0.95	0.95
1.2	0.95	0.95	0.95	0.95	0.95	0.95
1.0	0.95	0.95	0.95	0.95	0.95	0.95

C \ gamma	0.001	0.01	0.1	1.0	10.0	100.0
2.0	0.70	0.95	0.96	0.95	0.92	0.56
1.8	0.70	0.95	0.96	0.95	0.92	0.56
1.6	0.70	0.95	0.96	0.95	0.92	0.56
1.4	0.70	0.94	0.96	0.95	0.92	0.56
1.2	0.70	0.93	0.96	0.95	0.92	0.56
1.0	0.70	0.93	0.96	0.95	0.92	0.51

C \ gamma	1e-07	1e-06	1e-05	0.0001	0.001	0.01
100.0	0.37	0.37	0.70	0.93	0.96	0.97
10.0	0.37	0.37	0.37	0.70	0.93	0.96
1.0	0.37	0.37	0.37	0.37	0.70	0.93
0.1	0.37	0.37	0.37	0.37	0.37	0.70
0.01	0.37	0.37	0.37	0.37	0.37	0.37
0.001	0.37	0.37	0.37	0.37	0.37	0.37

Figure 5-9. Heat map visualizations of misspecified search grids

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Accuracy

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

TP - true positive

TN - true negative

FP - false positive

FN - false negative

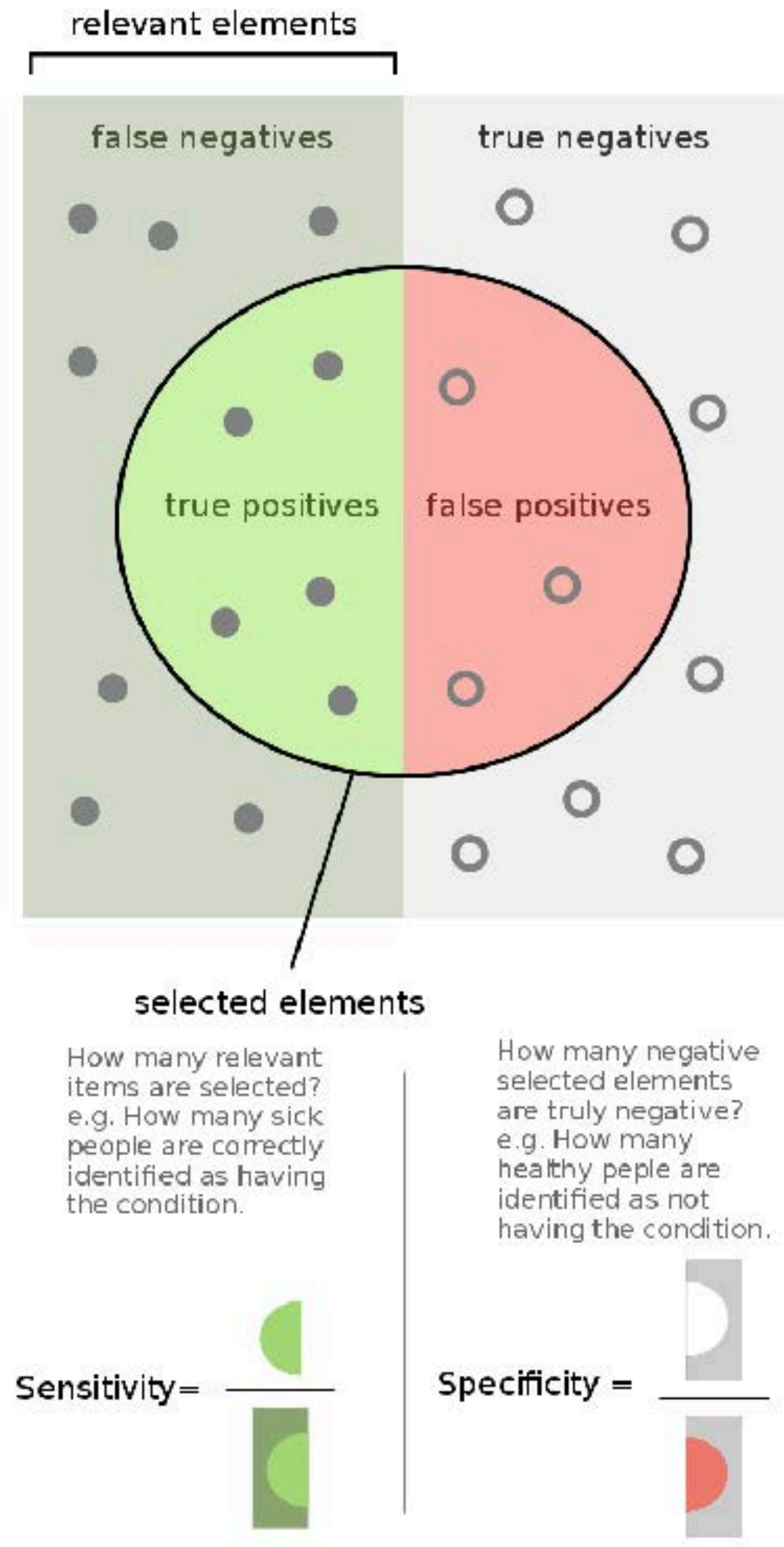
How many predictions were correct out of all the predictions made

Kinds of Errors

Type I: false positive

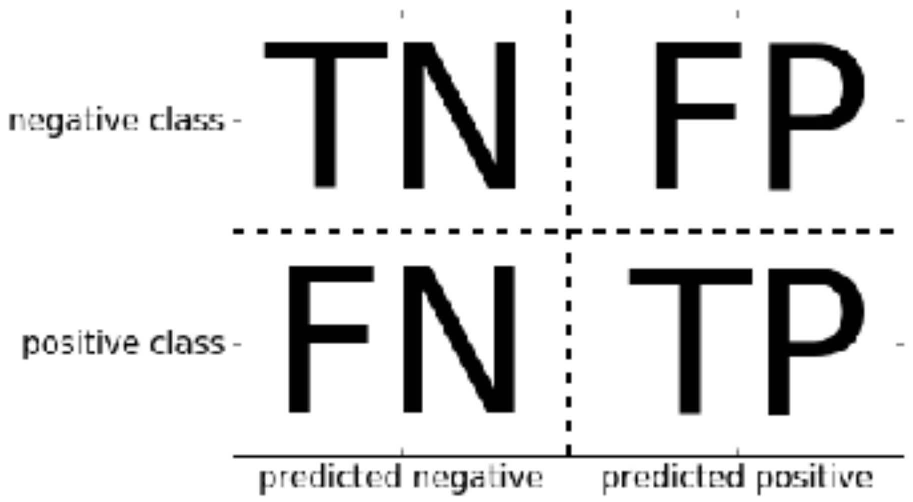
Type II: false negative

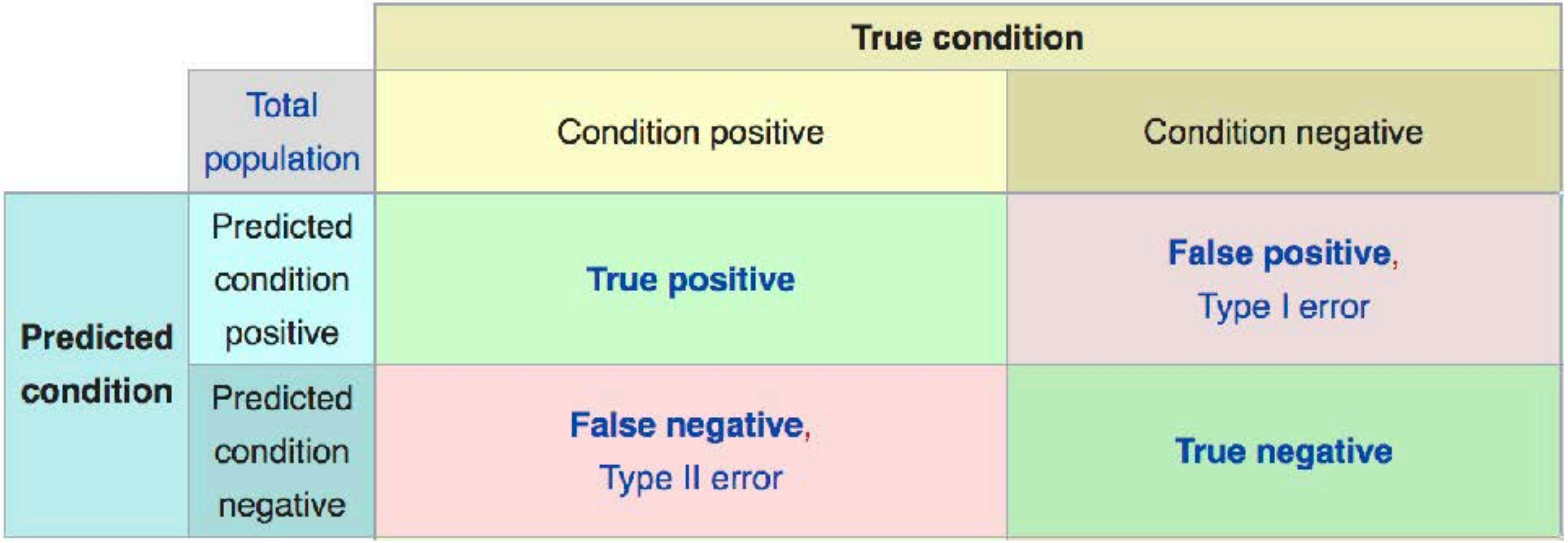
Balance of the classes in the dataset will influence how you interpret performance errors


relevant elements
false negatives
true negatives
true positives
false positives
selected elements
How many relevant items are selected? e.g. How many sick people are correctly identified as having the condition.
How many negative selected elements are truly negative? e.g. How many healthy peple are identified as not having the condition.
Sensitivity=
Specificity =

https://commons.wikimedia.org/wiki/File:Precisionrecall.svg

Confusion Matrices





		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error
	Predicted condition negative	**False negative**, Type II error	**True negative**

Confusion matrices visualize not only how many correct estimates a model made but also what and how it mis-classified

https://en.wikipedia.org/wiki/Precision_and_recall

Precision

$$\text{Precision} = \frac{tp}{tp + fp}$$

tp - true positive
fp - false positive

Positive predictive value

Number of the positive predicted values that are actually positive

Recall

$$\text{Recall} = \frac{tp}{tp + fn}$$

tp - true positive
fn - false negative

Sensitivity

Proportion of the actual positives identified

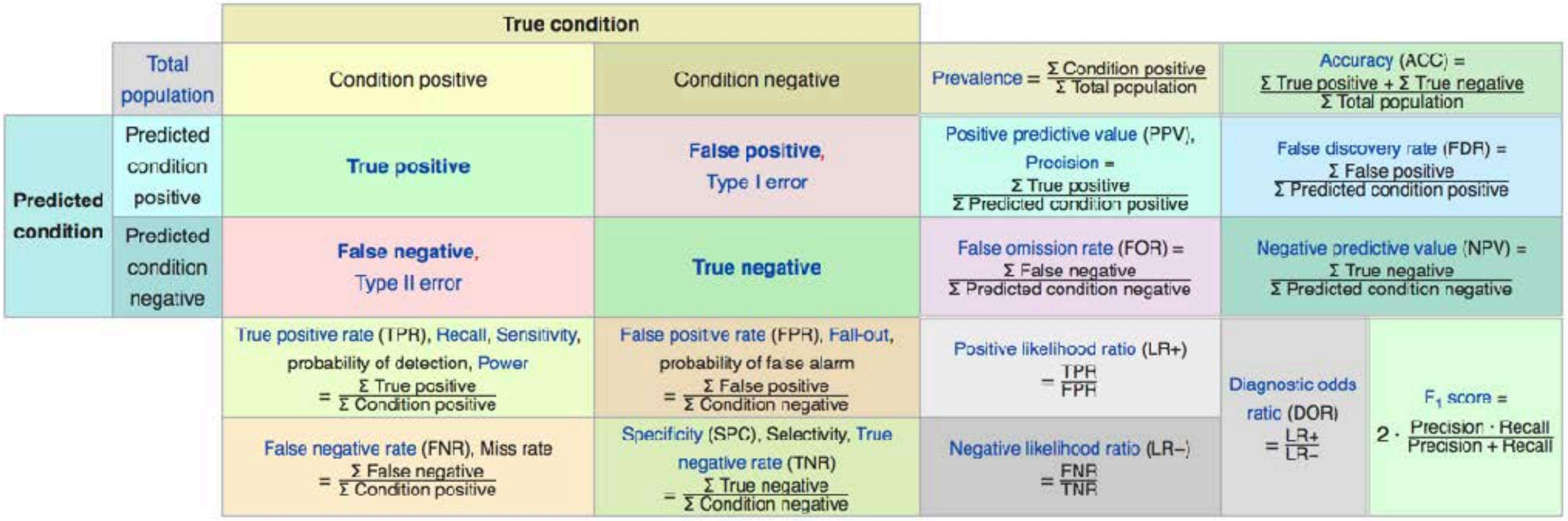
F-1 Score (F-score, F-measure)

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

Harmonic mean of precision and recall

Summary measurement of the classifier’s accuracy

Putting it All Together



		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$		

https://en.wikipedia.org/wiki/Precision_and_recall

In-Class Activity

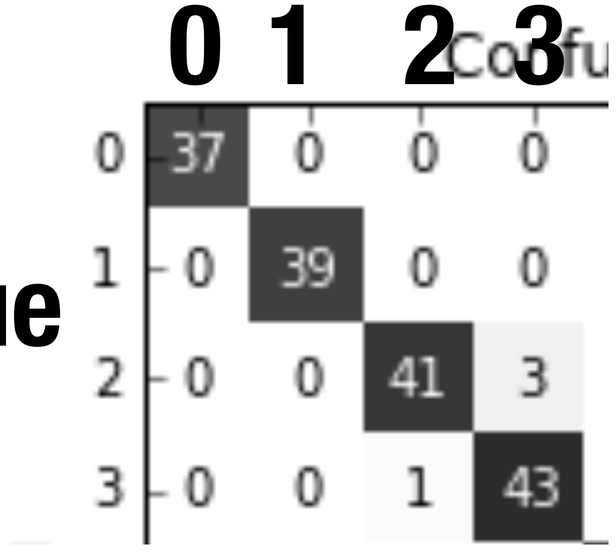
Calculate each of these metrics for the confusion matrix on the right

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$



Predicted

True	0	1	2	3
0	37	0	0	0
1	0	39	0	0
2	0	0	41	3
3	0	0	1	43

Figure 5-18. Confusion matrix for the 10-digit classification task

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Precision Recall Curves

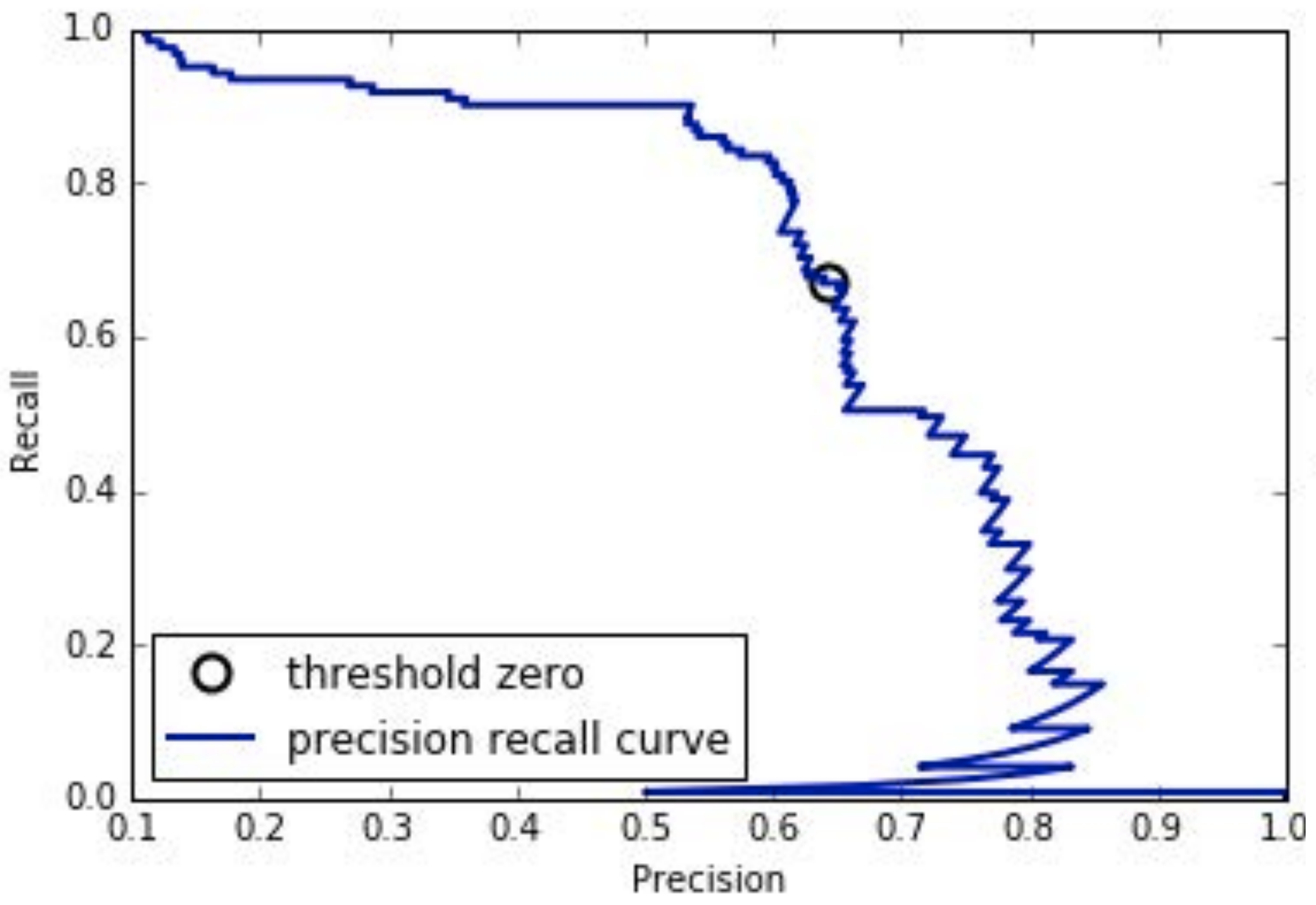


Figure 5-13. Precision recall curve for SVC(gamma=0.05)

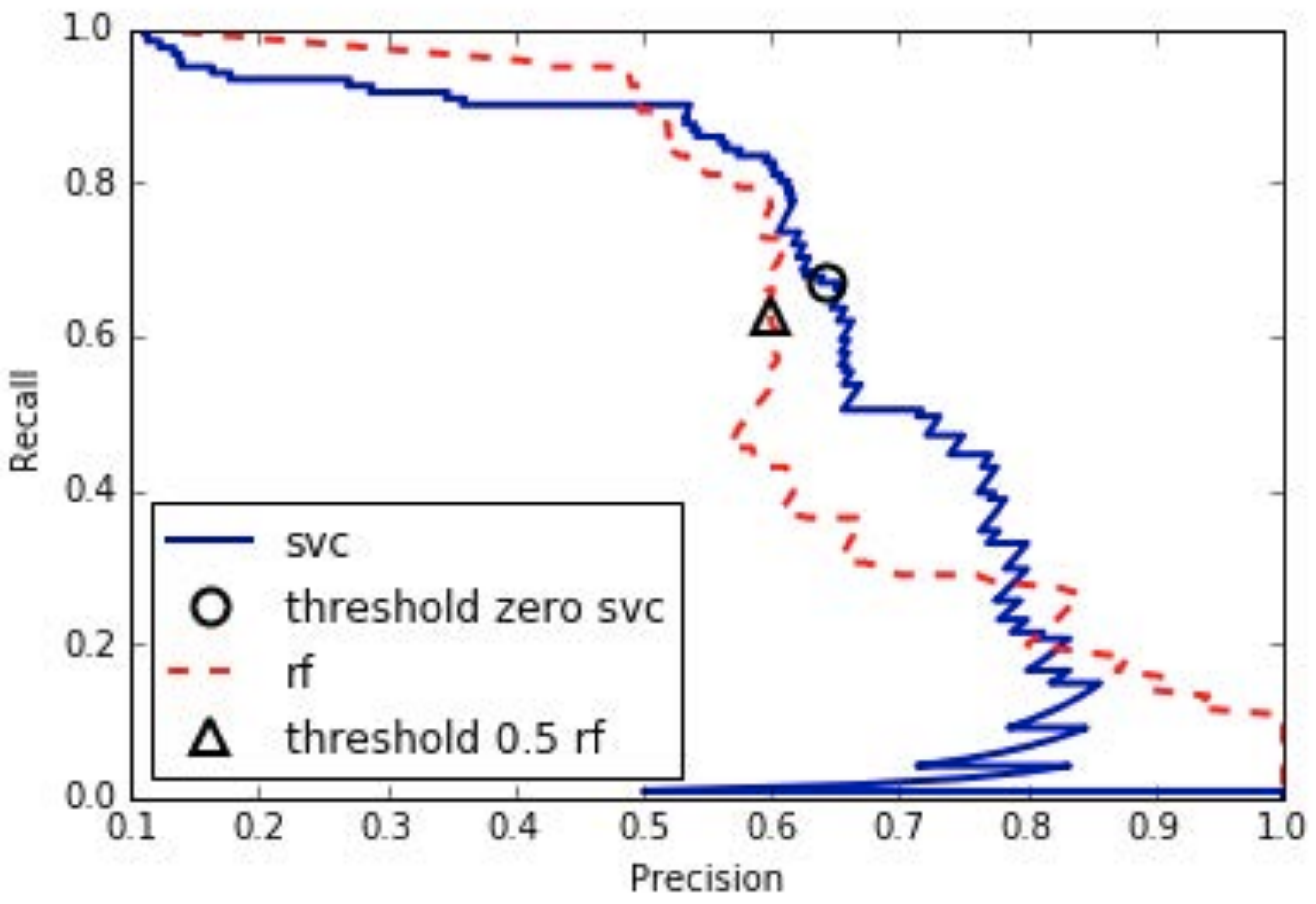


Figure 5-14. Comparing precision recall curves of SVM and random forest

Visualizes trade-off between positive predictive rate and true positive rate (sensitivity)

Area under curve summarizes this information

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

ROC Curves

$$FPR = \frac{FP}{FP + TN}$$

TP - true positive
TN - true negative
FP - false positive

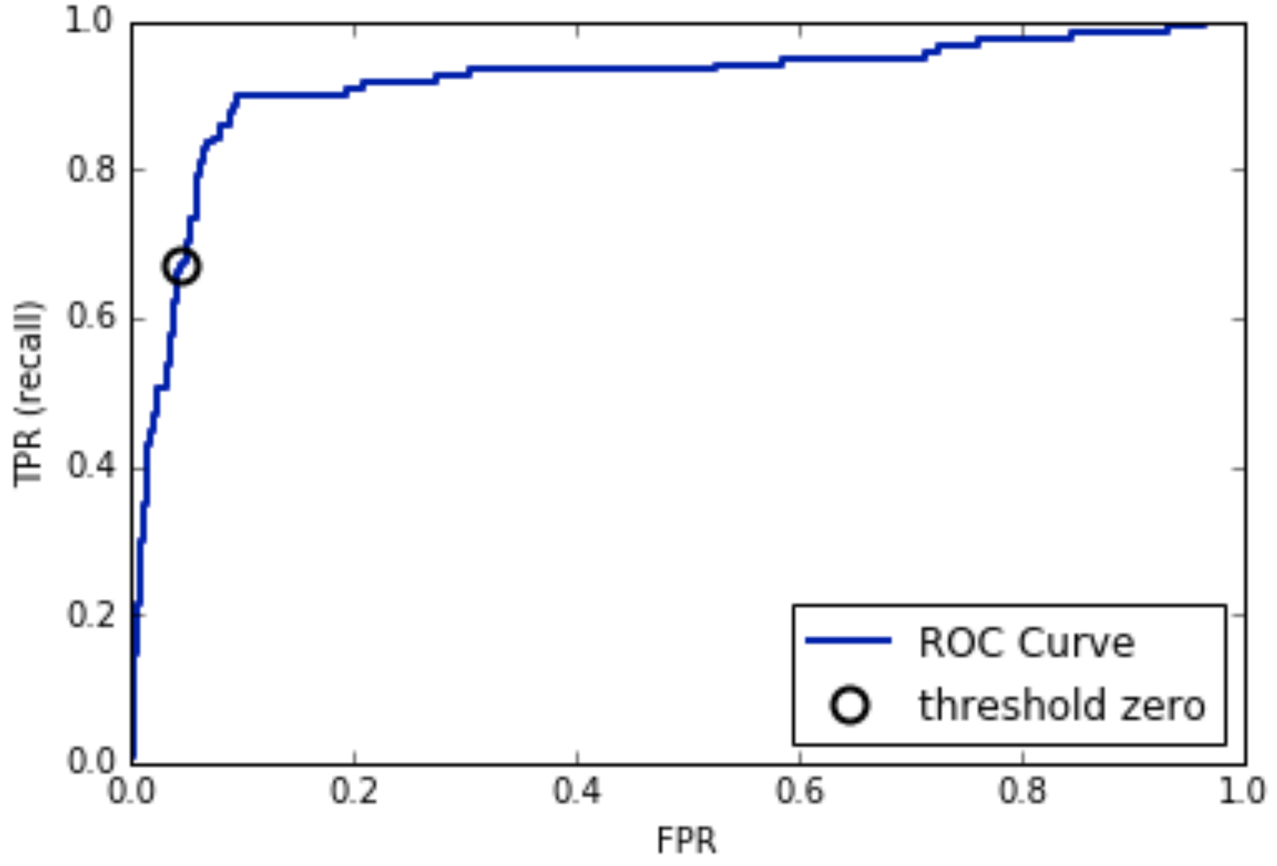


Figure 5-15. ROC curve for SVM

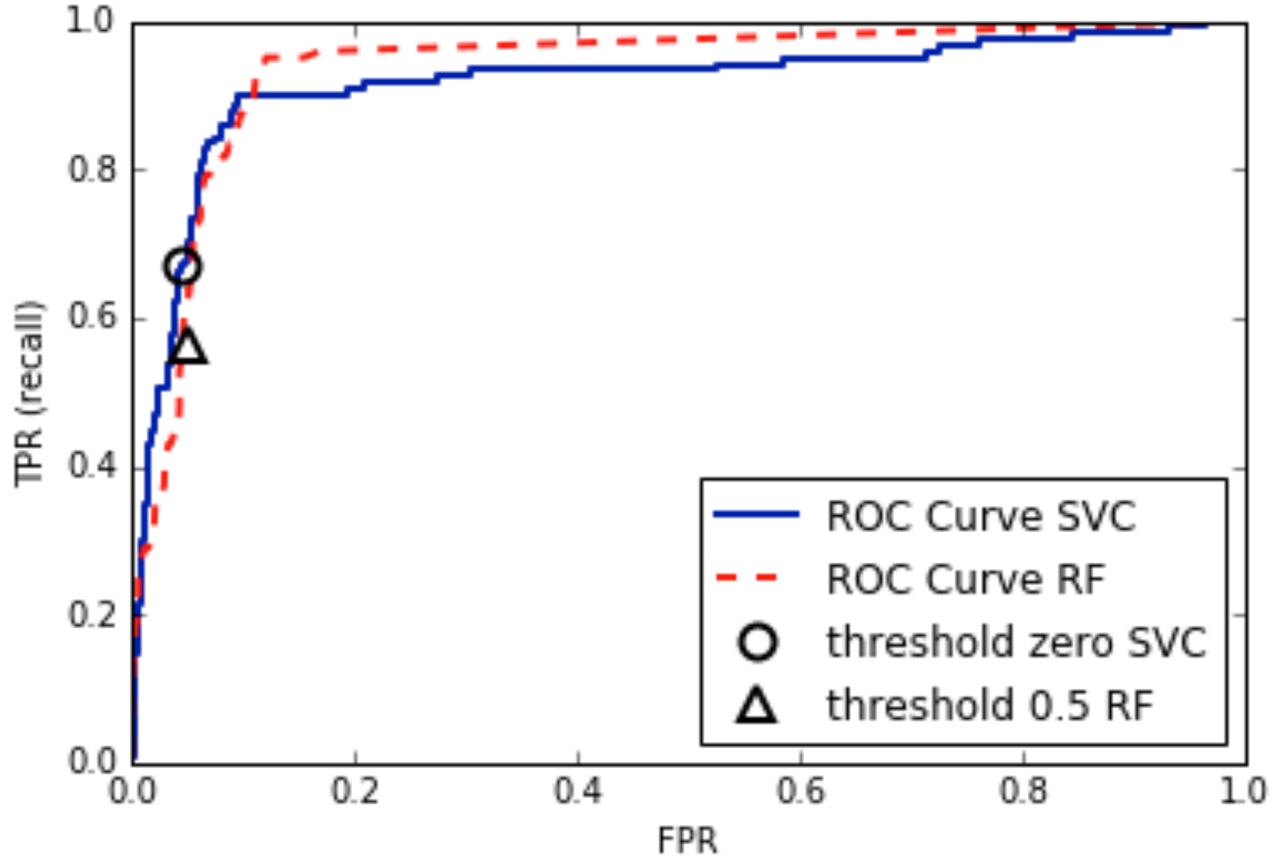


Figure 5-16. Comparing ROC curves for SVM and random forest

Receiver operating characteristics

Visualizes relationship between true positive rate (sensitivity) and false positive rate - the closer to the top right the better

Area under curve also useful to calculate here

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Decision Thresholds

How much uncertainty the classifier will permit to make a classification

Can be adjusted to change the prioritization between precision and recall

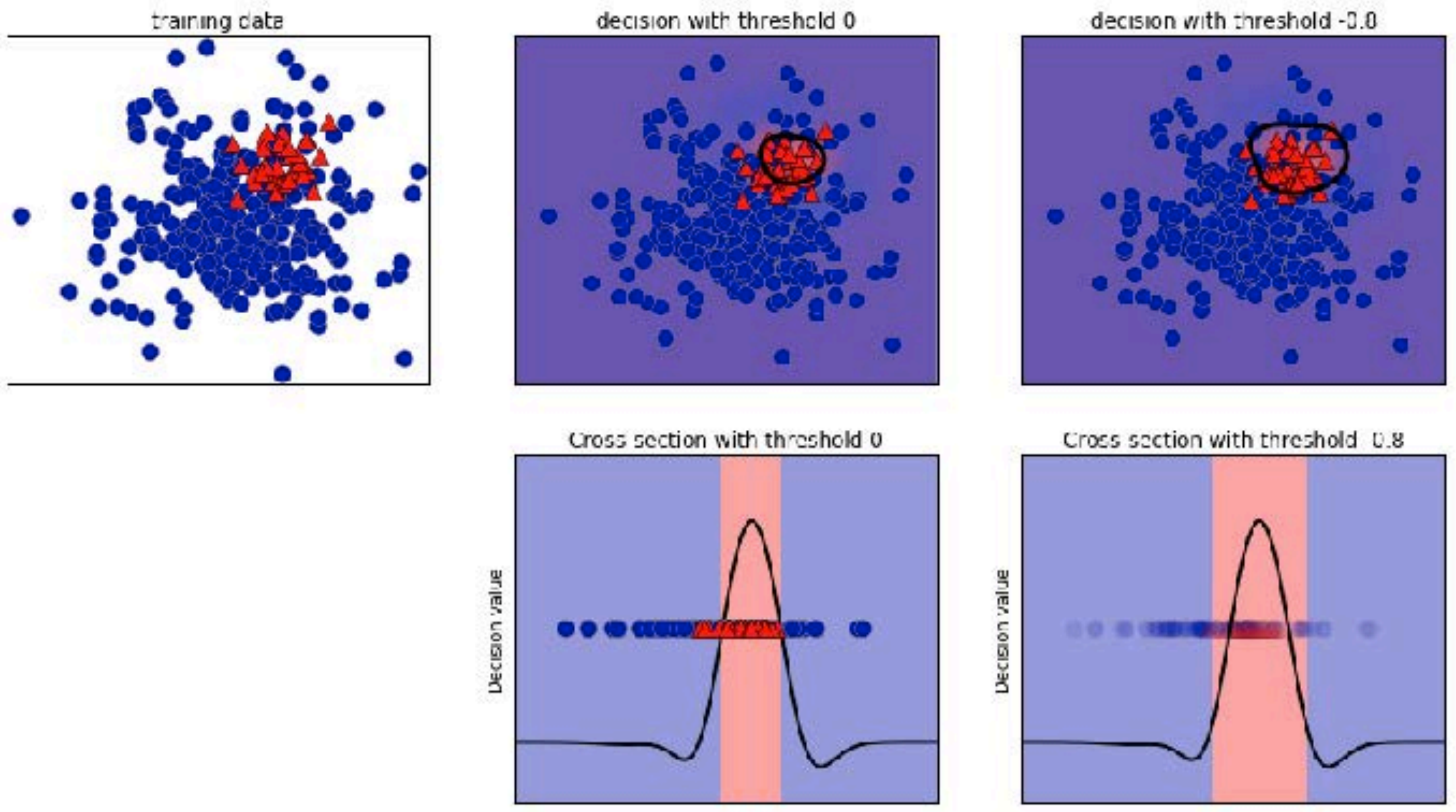


Figure 5-12. Heatmap of the decision function and the impact of changing the decision threshold

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Regression Performance Measures

Equation 2-1. Root Mean Square Error (RMSE)

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

m

- *m* is the number of instances in the dataset you are measuring the RMSE on.
  — For example, if you are evaluating the RMSE on a validation set of 2,000 districts, then $m = 2{,}000$.

$\mathbf{x}^{(i)}$ $y^{(i)}$

- $\mathrm{x}^{(i)}$ is a vector of all the feature values (excluding the label) of the i^{th} instance in the dataset, and $y^{(i)}$ is its label (the desired output value for that instance).

Géron, Aurélien. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow* O'Reilly Media, Inc.

Regression Performance Measures

$\mathbf{x}^{(i)}$ $y^{(i)}$

— For example, if the first district in the dataset is located at longitude –118.29°, latitude 33.91°, and it has 1,416 inhabitants with a median income of \$38,372, and the median house value is \$156,400 (ignoring the other features for now), then:

$$\mathbf{x}^{(1)} = \begin{pmatrix} -118.29 \\ 33.91 \\ 1,416 \\ 38,372 \end{pmatrix}$$

and:

$$y^{(1)} = 156,400$$

Regression Performance Measures

$\mathbf{X}$

- $\mathbf{X}$ is a matrix containing all the feature values (excluding labels) of all instances in the dataset. There is one row per instance and the i^{th} row is equal to the transpose of $\mathbf{x}^{(i)}$, noted $(\mathbf{x}^{(i)})^T$.[6]
  - — For example, if the first district is as just described, then the matrix $\mathbf{X}$ looks like this:

$$\mathbf{X} = \begin{pmatrix} \left(\mathbf{x}^{(1)}\right)^T \\ \left(\mathbf{x}^{(2)}\right)^T \\ \vdots \\ \left(\mathbf{x}^{(1999)}\right)^T \\ \left(\mathbf{x}^{(2000)}\right)^T \end{pmatrix} = \begin{pmatrix} -118.29 & 33.91 & 1,416 & 38,372 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Regression Performance Measures

h

- h is your system's prediction function, also called a *hypothesis*. When your system is given an instance's feature vector $\mathbf{x}^{(i)}$, it outputs a predicted value $\hat{y}^{(i)} = h(\mathbf{x}^{(i)})$ for that instance ($\hat{y}$ is pronounced "y-hat").
  - — For example, if your system predicts that the median housing price in the first district is \$158,400, then $\hat{y}^{(1)} = h(\mathbf{x}^{(1)}) = 158,400$. The prediction error for this district is $\hat{y}^{(1)} - y^{(1)} = 2,000$.

$\text{RMSE}(\mathbf{X},h)$

- $\text{RMSE}(\mathbf{X},h)$ is the cost function measured on the set of examples using your hypothesis h.

- Computing the root of a sum of squares (RMSE) corresponds to the *Euclidian norm*: it is the notion of distance you are familiar with. It is also called the ℓ_2 *norm*, noted $\| \cdot \|_2$ (or just $\| \cdot \|$).

Géron, Aurélien. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow* O'Reilly Media, Inc.

Regression Performance Measures

Equation 2-2. Mean Absolute Error

$$\mathrm{MAE}(\mathbf{X}, h) = \frac{1}{m} \sum_{i=1}^{m} \left| h\left(\mathbf{x}^{(i)}\right) - y^{(i)} \right|$$

m

- m is the number of instances in the dataset you are measuring the RMSE on.

$\mathbf{x}^{(i)}$ $y^{(i)}$

- $x^{(i)}$ is a vector of all the feature values (excluding the label) of the *i*th instance in the dataset, and $y^{(i)}$ is its label (the desired output value for that instance).

- Computing the sum of absolutes (MAE) corresponds to the ℓ_1 *norm*, noted $\| \cdot \|_1$. It is sometimes called the *Manhattan norm* because it measures the distance between two points in a city if you can only travel along orthogonal city blocks.

Regression Performance Measures

RMSE is more sensitive to outliers than the MAE

- when outliers are exponentially rare (like in a bell-shaped curve), the RMSE performs very well and is generally preferred

Upcoming Work

- **DataCamp for June 9**
  - Cleaning Data in Python
  - Pre-processing for Machine Learning in Python course
  - Model Validation in Python course
- **Videos for June 13 (password: data71200)**
  - K-nearest Neighbors: https://vimeo.com/400660692
  - Linear Models: https://vimeo.com/403004687
- **Reading for June 13**
  - Ch 2: “Supervised Learning” in Guido, Sarah and Andreas C. Muller. (2016). *Introduction to Machine Learning with Python*, O’Reilly Media, Inc. 27–70
- **Project 1 due on June 13**