

Introduction to supervised machine learning,
k-fold cross-validation, nearest neighbors, and
linear models

Toby Dylan Hocking

Supervised machine learning

- ▶ Goal is to learn a function $f(\mathbf{x}) = y$ where \mathbf{x} is an input/feature vector and y is an output/label.
- ▶ x = image of digit/clothing, $y \in \{0, \dots, 9\}$ (ten classes).
- ▶ x = vector of word counts in email, $y \in \{1, 0\}$ (spam or not).
- ▶ x = image of retina, y = risk score for heart disease.
- ▶ This week we will focus on a specific kind of supervised learning problem called binary classification, which means $y \in \{1, 0\}$.

Learning algorithm

- ▶ We want a learning algorithm `LEARN` which inputs a training data set and outputs a prediction function f .
- ▶ In math a training data set with n observations and p features is a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ with a label vector $\mathbf{y} \in \{0, 1\}^n$.
- ▶ On computers it is a CSV file with n rows and $p + 1$ columns.
- ▶ Want: $\text{LEARN}(\mathbf{X}, \mathbf{y}) \rightarrow f$.
- ▶ We will use three such data sets from Elements of Statistical Learning book by Hastie et al. (mixture slightly modified)

name	observations, n	inputs/features, p	outputs/labels
zip.test	images, 623	pixel intensities, 256	0/1 digits
spam	emails, 4601	word counts, 57	spam=1/not=0
mixture	people, 200	height/weight, 2	democratic/republican

<https://github.com/tdhock/cs570-spring-2022/tree/master/data>

<https://hastie.su.domains/ElemStatLearn/data.html>

Mixture data table

```
##           party height_in  weight_lb
## 0    democratic  71.741421  149.565034
## 1    democratic  69.582283  149.275446
## 2    democratic  69.983547  149.961470
## 3    democratic  69.908764  150.021178
## 4    democratic  69.195491  150.111237
## ..          ...          ...          ...
## 195 republican  69.472078  151.537588
## 196 republican  71.140501  149.409036
## 197 republican  70.517269  150.236183
## 198 republican  69.223459  151.486248
## 199 republican  69.019082  149.795387
##
## [200 rows x 3 columns]
```

Spam data table

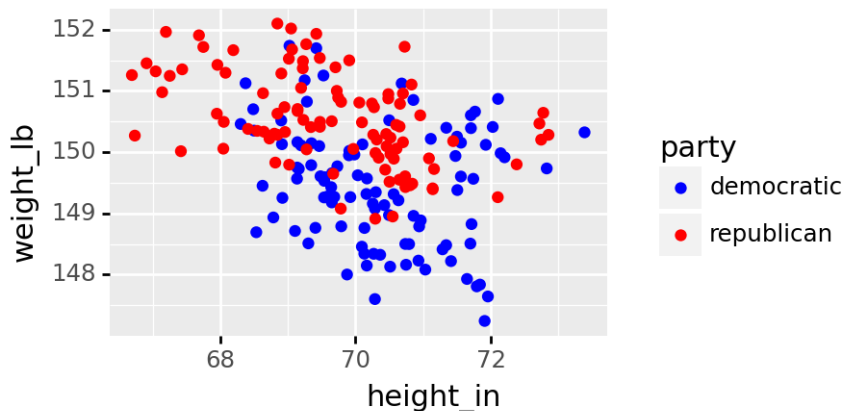
```
##           0           1           2           ...           55           56           57
## 0         0.00        0.64        0.64        ...           61          278           1
## 1         0.21        0.28        0.50        ...          101         1028           1
## 2         0.06        0.00        0.71        ...          485         2259           1
## 3         0.00        0.00        0.00        ...           40          191           1
## 4         0.00        0.00        0.00        ...           40          191           1
## ...         ...         ...         ...         ...         ...         ...         ..
## 4596       0.31        0.00        0.62        ...            3            88           0
## 4597       0.00        0.00        0.00        ...            4            14           0
## 4598       0.30        0.00        0.30        ...            6           118           0
## 4599       0.96        0.00        0.00        ...            5            78           0
## 4600       0.00        0.00        0.65        ...            5            40           0
##
## [4601 rows x 58 columns]
```

Zip.test data table

```
##          0      1      2      ...    254    255    256
## 0         9 -1.0 -1.0    ...   -1.0  -1.0  -1.0
## 1         6 -1.0 -1.0    ...   -1.0  -1.0  -1.0
## 2         3 -1.0 -1.0    ...   -1.0  -1.0  -1.0
## 3         6 -1.0 -1.0    ...   -1.0  -1.0  -1.0
## 4         6 -1.0 -1.0    ...   -1.0  -1.0  -1.0
## ...      ...    ...    ...    ...    ...    ...
## 2002       3 -1.0 -1.0    ...   -1.0  -1.0  -1.0
## 2003       9 -1.0 -1.0    ...   -1.0  -1.0  -1.0
## 2004       4 -1.0 -1.0    ...   -1.0  -1.0  -1.0
## 2005       0 -1.0 -1.0    ...   -1.0  -1.0  -1.0
## 2006       1 -1.0 -1.0    ...   -1.0  -1.0  -1.0
##
## [2007 rows x 257 columns]
```

Visualize mixture data set

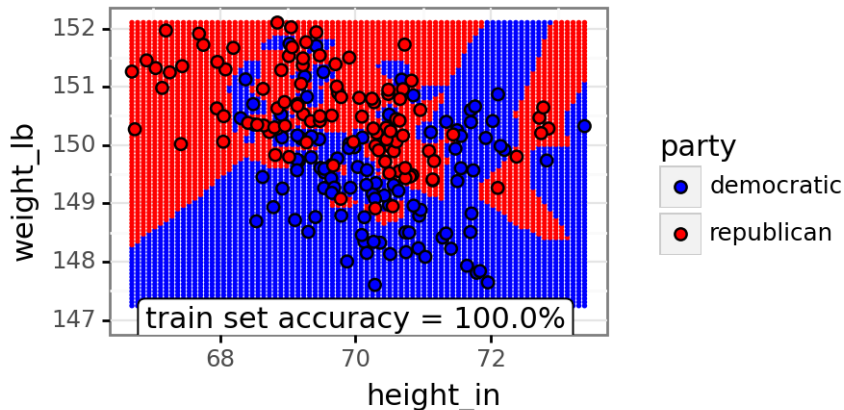
- ▶ Each axis represents one column of the \mathbf{X} matrix.
- ▶ Each point represents one row of the \mathbf{X} matrix.
- ▶ Color represents class label \mathbf{y} .



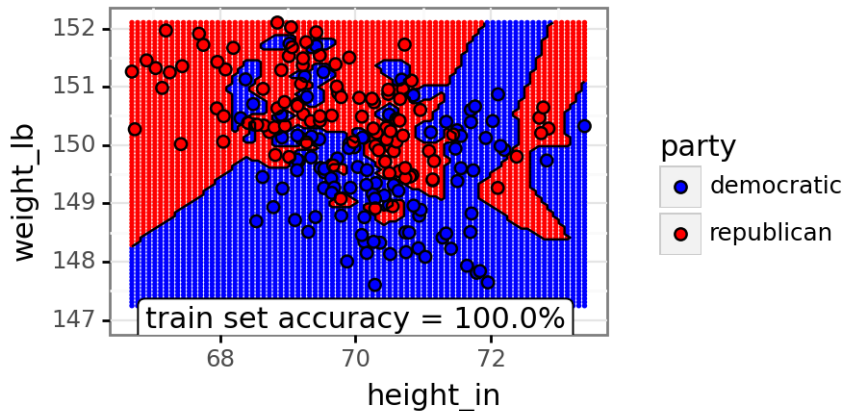
A basic machine learning algorithm

- ▶ Goal of supervised learning is to learn a function which predicts the label for new inputs $x \in \mathbb{R}^2$.
- ▶ K-Nearest neighbors: a simple non-linear algorithm.
- ▶ For any new data point, predict the average label of the K nearest neighbors.

Visualize predictions of 1-nearest neighbor algorithm



Also plot decision boundary in black

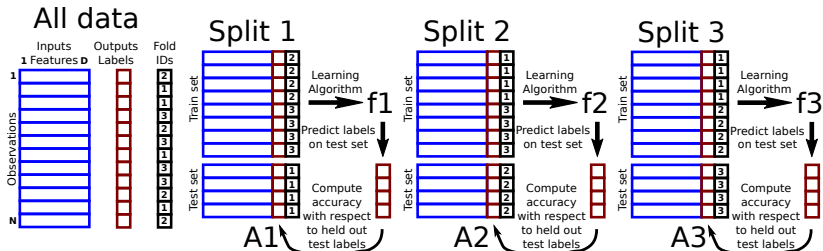


Is it good to have 100% accuracy on train data?

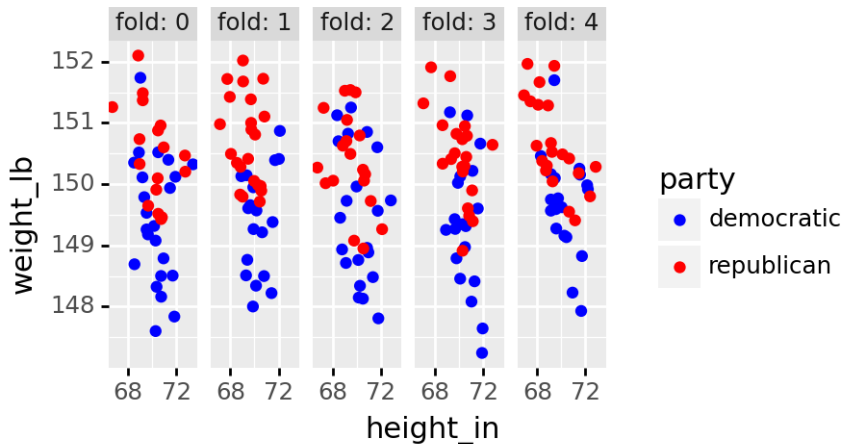
- ▶ Remember: goal is function f with accurate predictions on new inputs.
- ▶ What is a new input?
- ▶ We must assume that new/test inputs are similar to old/train inputs.
- ▶ In the statistical literature this is the iid (independent and identically distributed) assumption.
- ▶ We can therefore split the full data set into train/test sets.
- ▶ Train set is used to learn the prediction function f .
- ▶ Test set (simulated new inputs) is used to evaluate the accuracy of the function f (but can not be used to learn function f).

K-fold cross-validation for splitting data

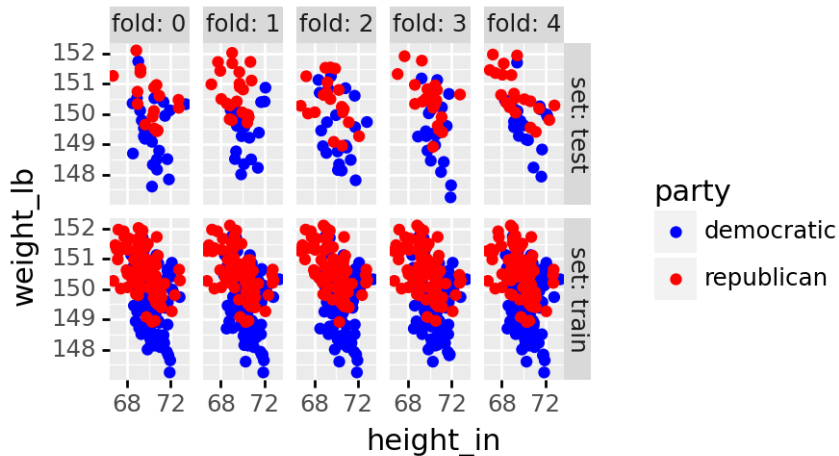
- ▶ One way to split is via K-fold cross-validation.
- ▶ Each row is assigned a fold ID number from 1 to K.
- ▶ For each for ID, those data are held out, and other data are kept.
- ▶ Popular relative to other splitting methods because of simplicity and fairness (each row is held out one time).



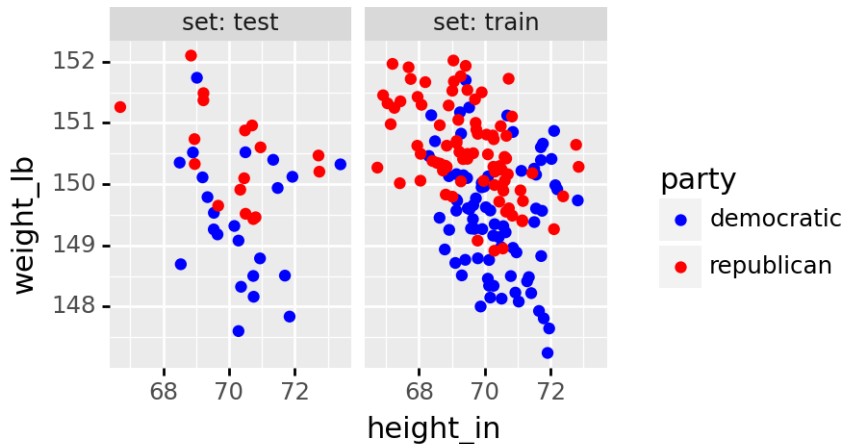
Visualization of fold IDs in input/feature space



Visualization of splits/sets in input/feature space

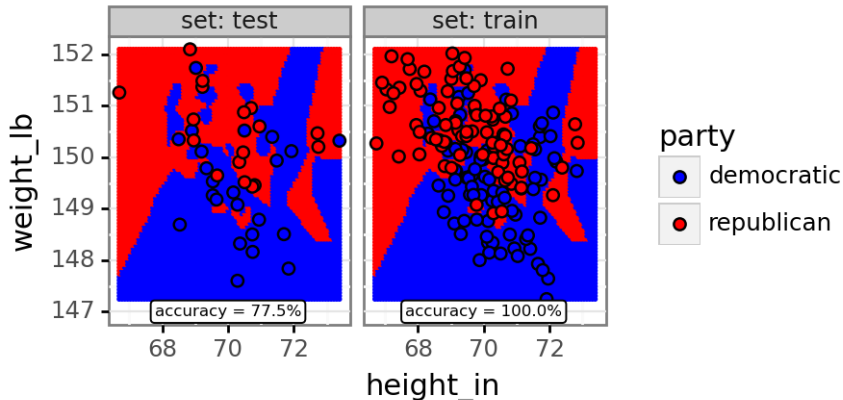


One split



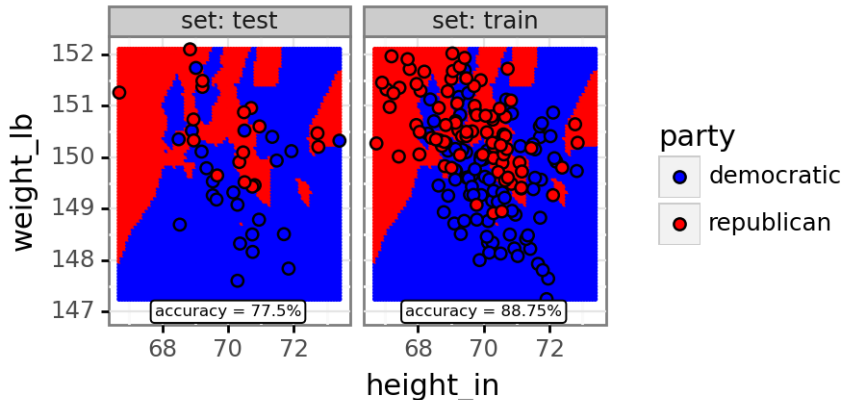
Nearest neighbor predictions

1 nearest neighbors



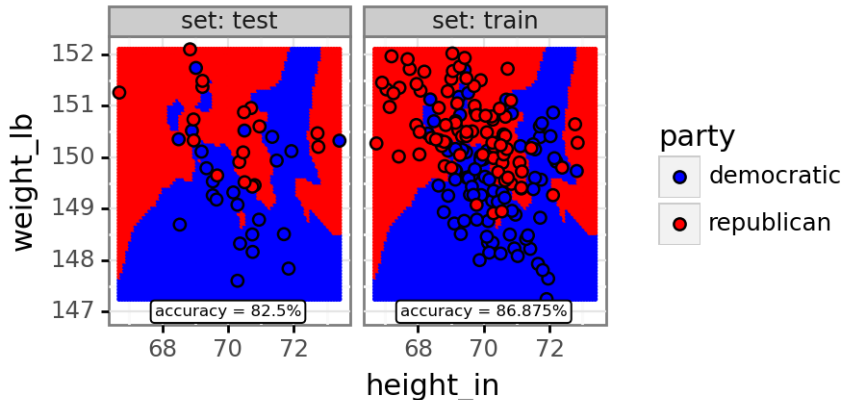
Nearest neighbor predictions

2 nearest neighbors



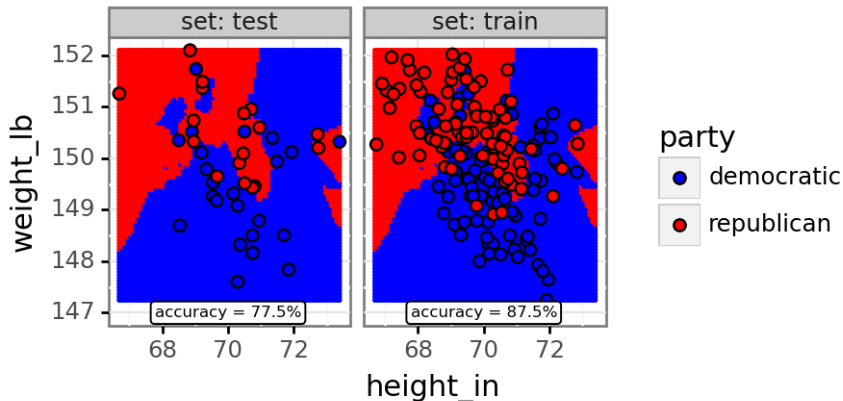
Nearest neighbor predictions

3 nearest neighbors



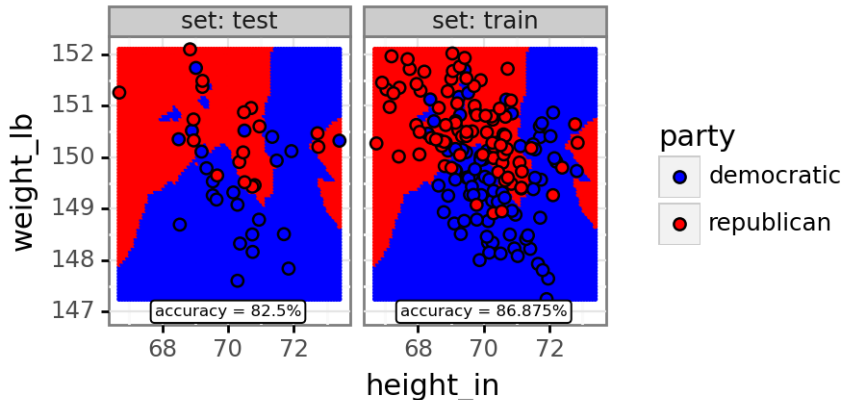
Nearest neighbor predictions

4 nearest neighbors



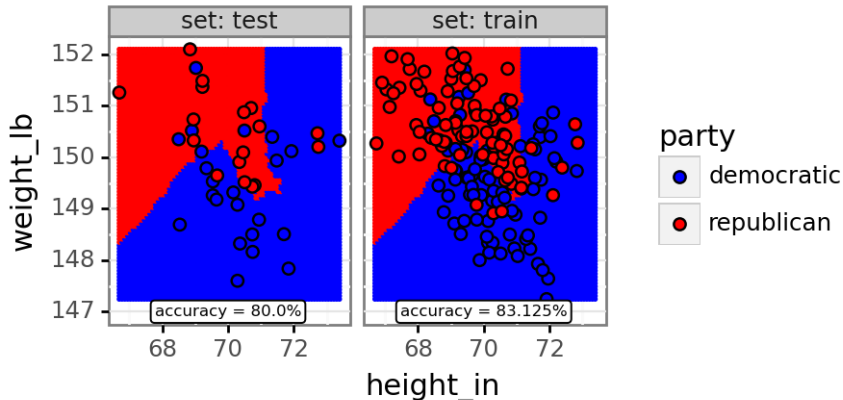
Nearest neighbor predictions

5 nearest neighbors



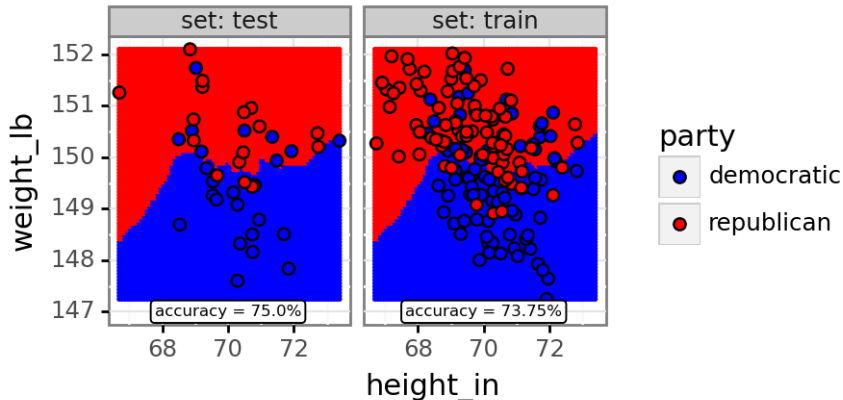
Nearest neighbor predictions

10 nearest neighbors



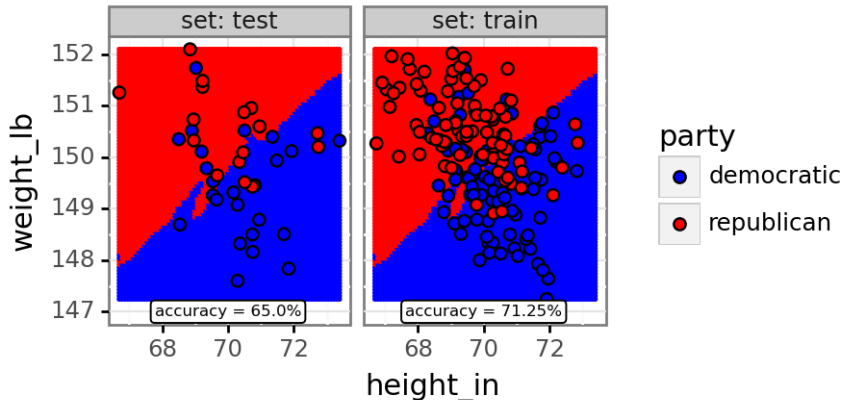
Nearest neighbor predictions

50 nearest neighbors

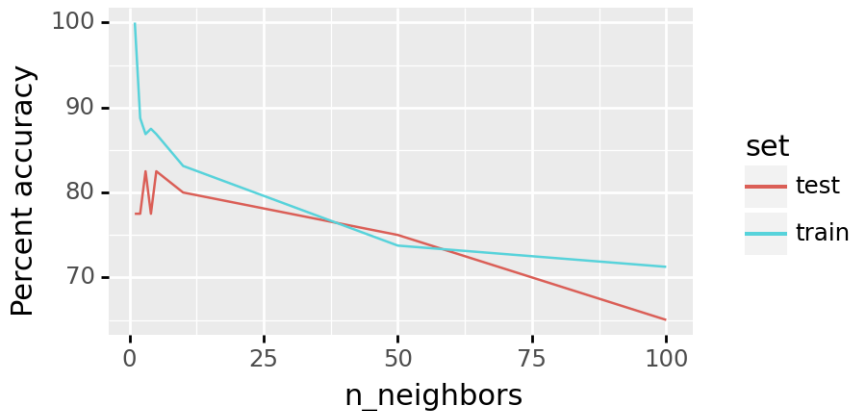


Nearest neighbor predictions

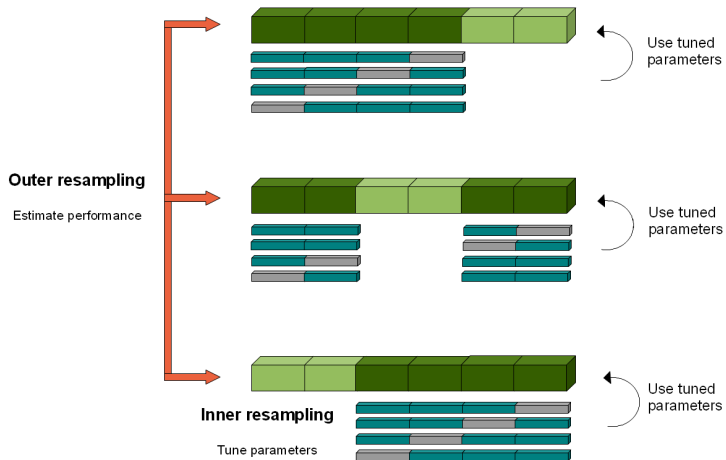
100 nearest neighbors



Accuracy for each model size




Two kinds of splits



 Training set
outer resampling

 Test set
outer resampling

 Subtrain set
inner resampling

 Validation set
inner resampling

Implementing splits in python

- ▶ Full data into train/test ->
`sklearn.model_selection.KFold`. For evaluating prediction accuracy and comparing different algorithms.
- ▶ Train into subtrain/validation ->
`sklearn.model_selection.GridSearchCV`. For learning hyper-parameters such as `n_neighbors` which must be fixed before running the learning algorithm / computing predictions.

Basic idea of linear model

How to fairly compare linear model with nearest neighbors?