Lecture 6

Decision Trees

By: Nazerke Sultanova

Decision Trees

- Decision tree is one of the most popular machine learning algorithms.

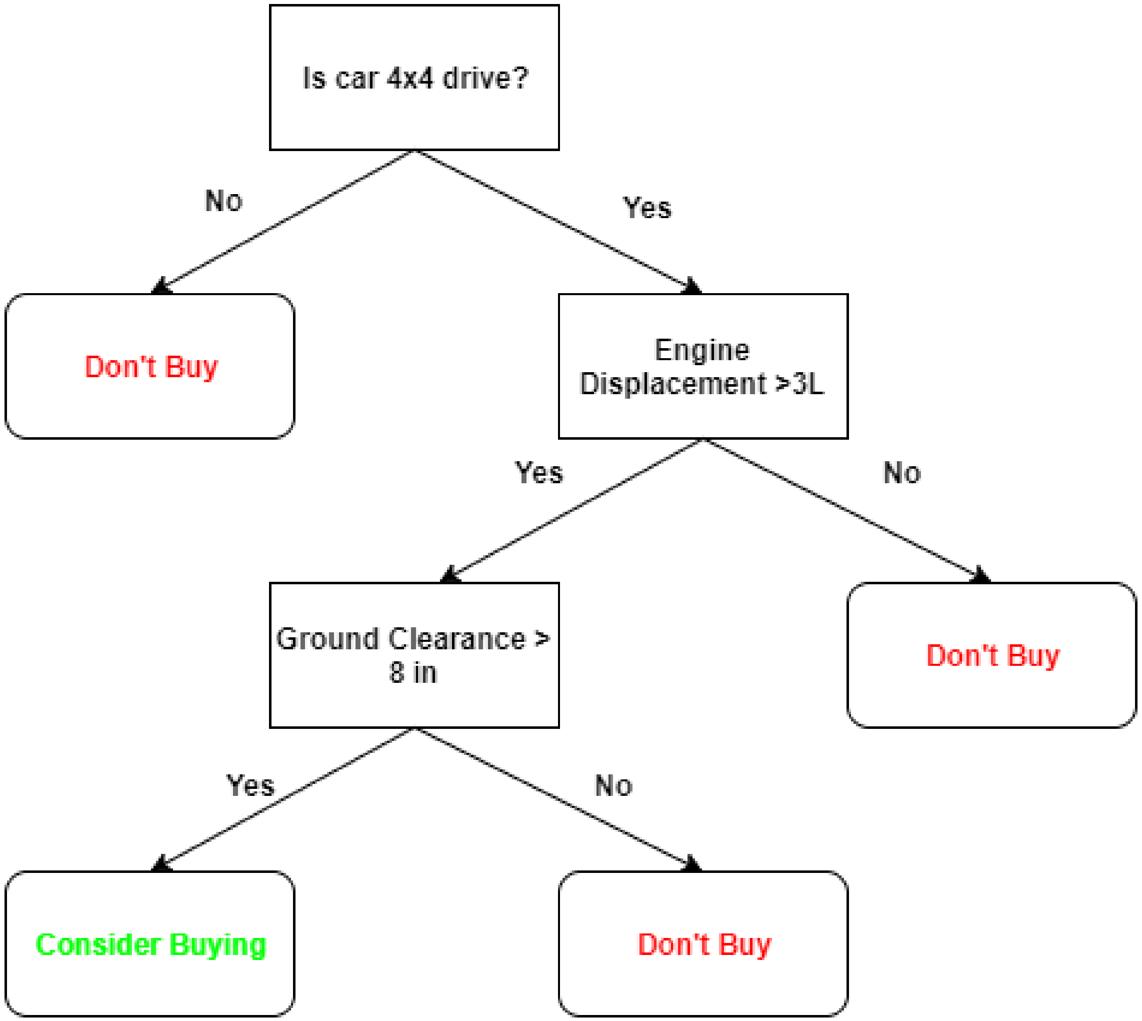
- Decision trees are used for both classification and regression problems.

Intuition:

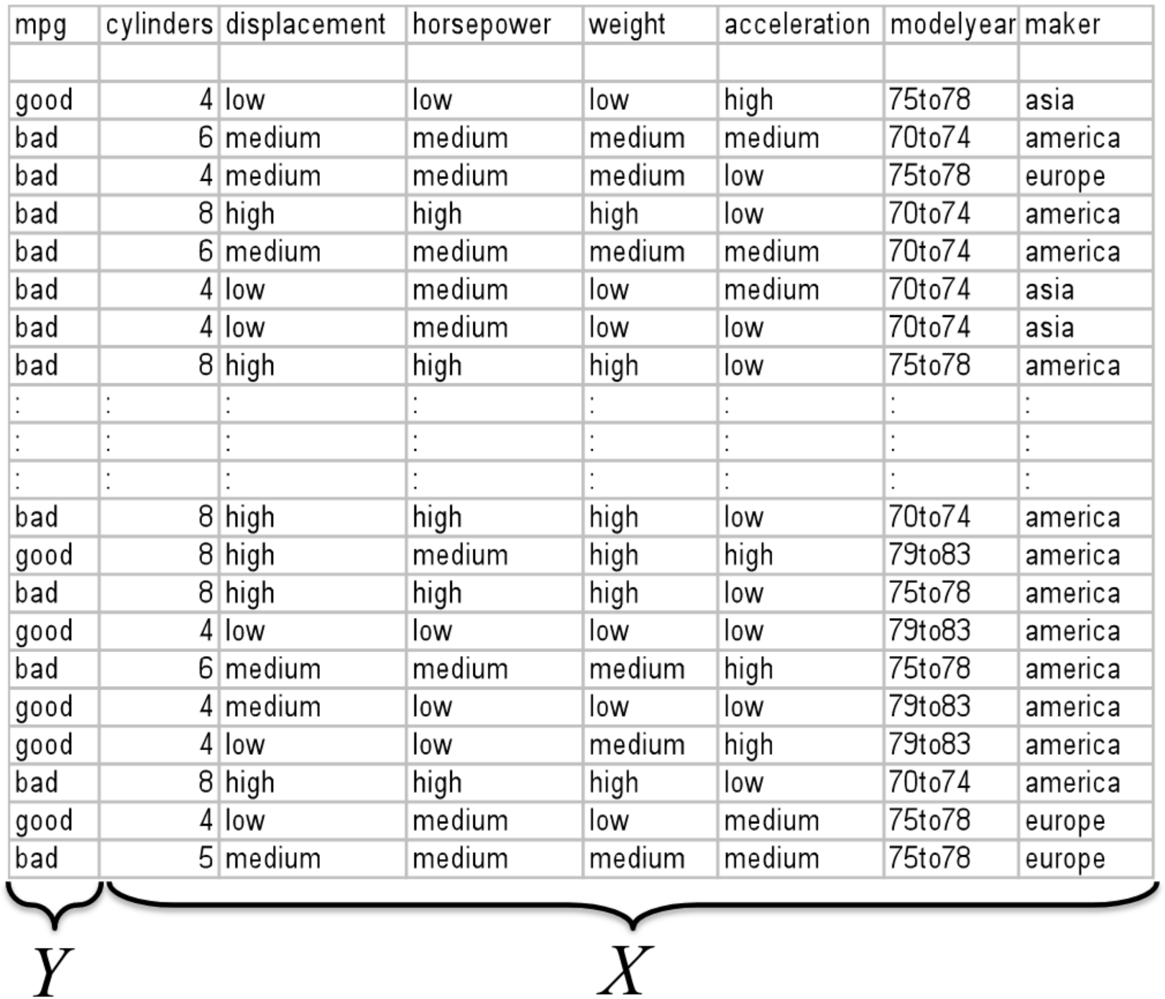
- 20 Questions Game

- Idea: Breaking down our data by making decisions based on asking series of questions

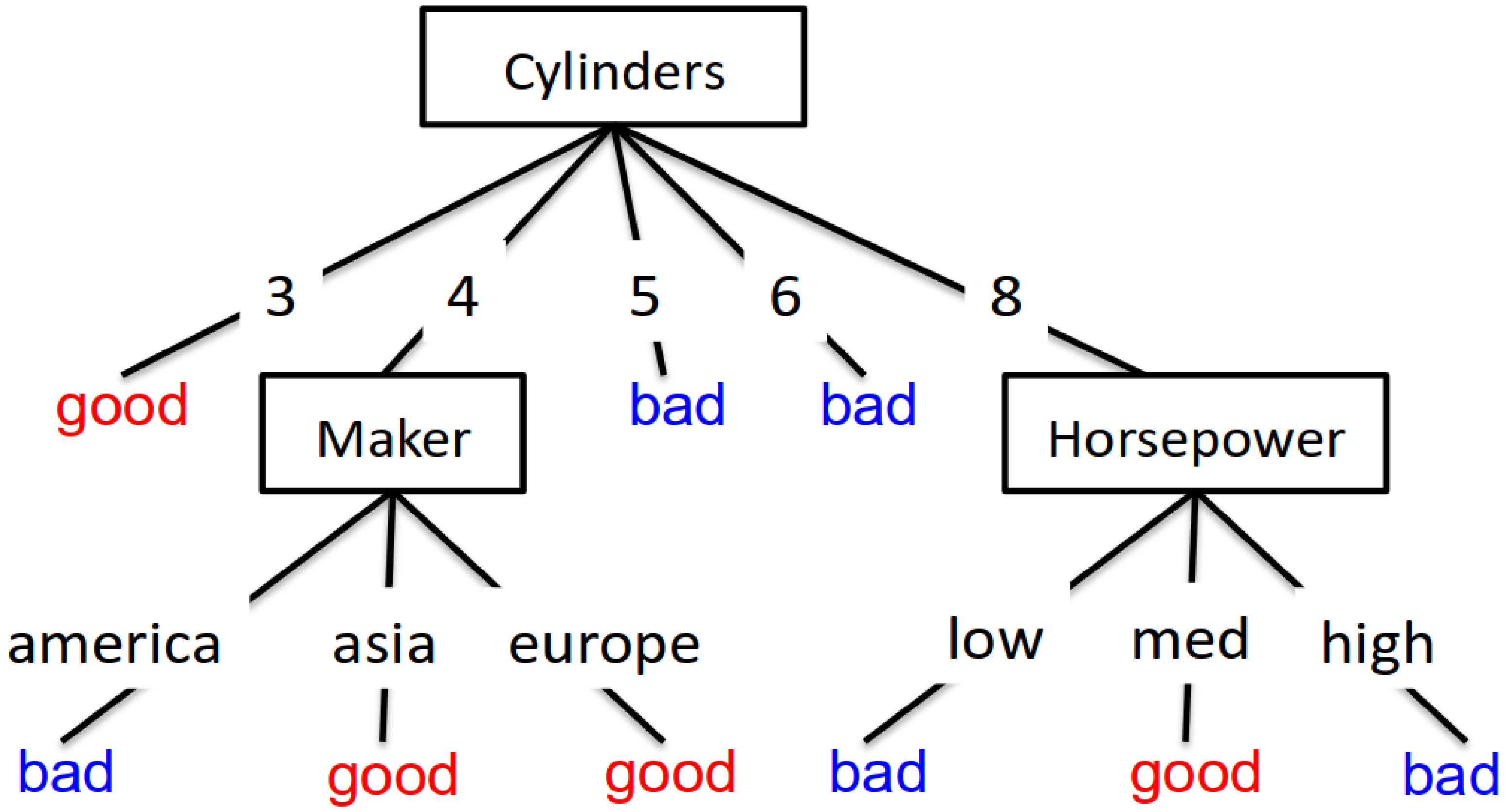
Decision Tree for Classification and Prediction



Example dataset:



mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe


Cylinders
3
4
5
6
8
good
Maker
bad
bad
Horsepower
america
asia
europe
low
med
high
bad
good
good
bad
good
bad

How to construct a tree??

- Resort to a greedy heurisPc:
  - –Start from empty decision tree
  - –Split on next best attribute (split the data on the feature that results in the largest information gain (IG))
  - - Recurse

Choosing best feature to split

- Objective function - maximizing information gain
- Information for symbol x_i: $l(x_i) = log_2 p(x_i)$

  where $p(x_i)$ is the probability of choosing this class
- Entropy: $H = -\sum_{i=1}^{c} p(x_i) log_2 p(x_i)$

where c is the number of classes, $p(x_i)$- fraction of examples in class i

Information Gain

- The change in information before and after the split is known as the information gain. When you know how to calculate the information gain, you can split your data across every feature to see which split gives you the highest information gain. The split with the highest information gain is your best option

information gain = entropy(parent) – (weighted average) entropy (children)

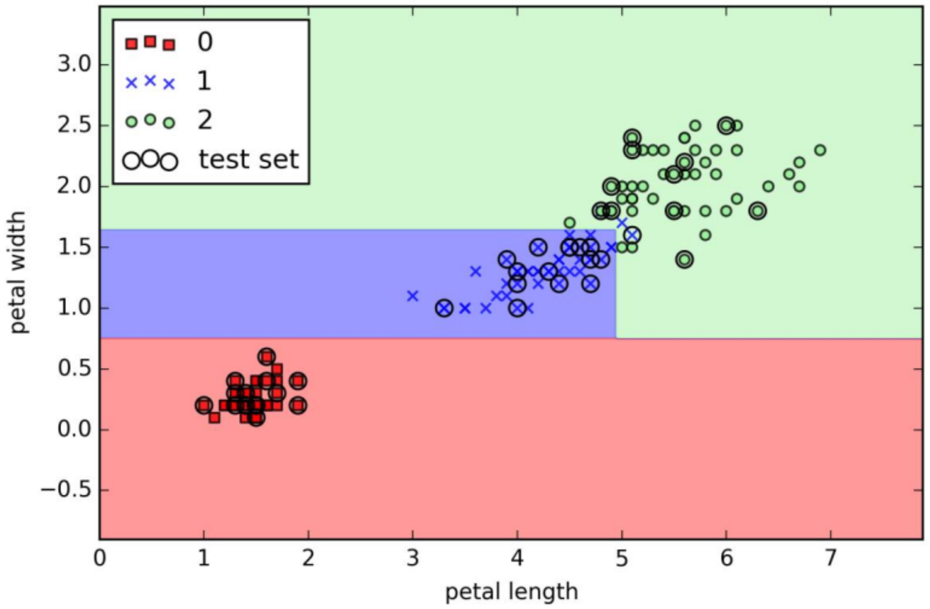
decision tree algorithm => maximize information gain

Practice information gain

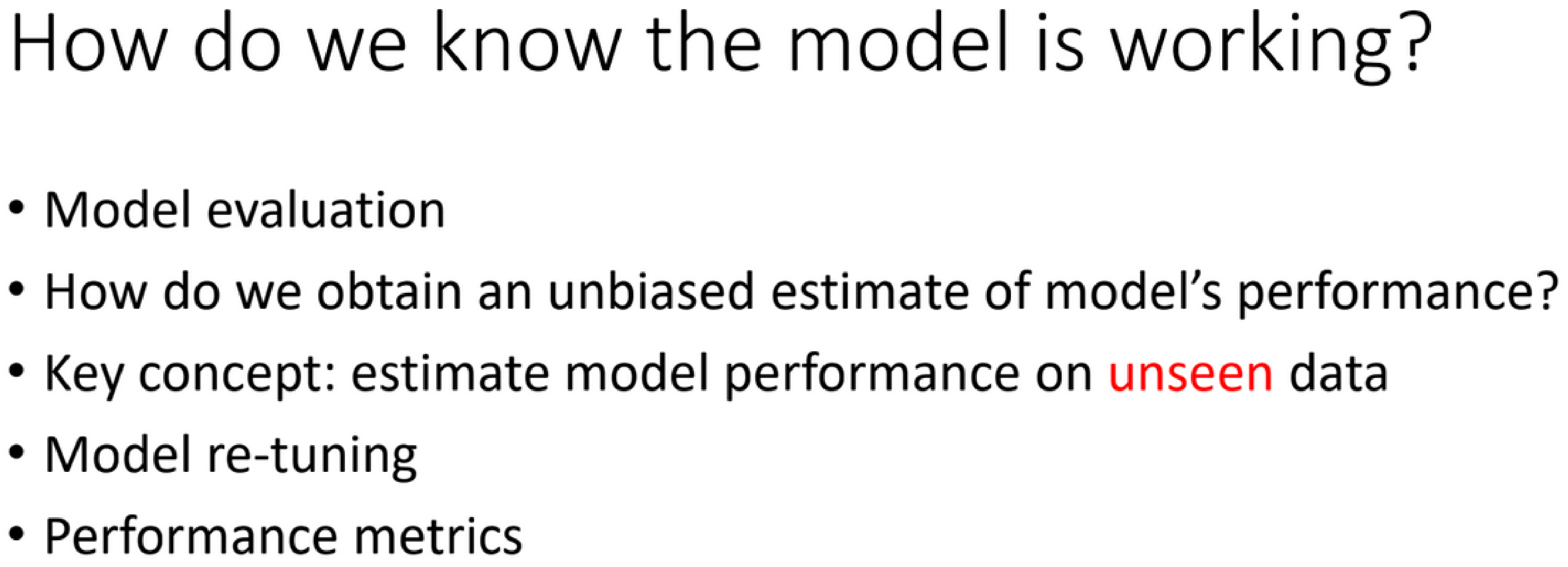
Grade	Bumpiness	Speed limit	Speed
Steep	Bumpy	Yes	Slow
Steep	Smooth	Yes	Slow
Flat	Bumpy	No	Fast
Steep	Smooth	No	Fast

Building a decision tree

```
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
tree.fit(X_train, y_train)

X_combined = np.vstack((X_train, X_test))
y_combined = np.hstack((y_train, y_test))
plot_decision_regions(X_combined, y_combined,
                      classifier=tree, test_idx=range(105, 150))

plt.xlabel('petal length [cm]')
plt.ylabel('petal width [cm]')
plt.legend(loc='upper left')
plt.tight_layout()
# plt.savefig('./figures/decision_tree_decision.png', dpi=300)
plt.show()
```



Evaluation of ML Algorithms:



How do we know the model is working?

- Model evaluation
- How do we obtain an unbiased estimate of model’s performance?
- Key concept: estimate model performance on unseen data
- Model re-tuning
- Performance metrics

Data Preprocessing

- Dealing with missing values:
  - Missing values while survey
  - Most tools unable to deal with missing values
  - Eliminating features or samples
  - Imputing missing values
    - Mean imputation
    - Most frequent imputation

Data Preprocessing

- Handling categorical data:
  - Data sometimes contains features with categorical data
    - Nominal features (colors)
    - Ordinal features (M, L, XL)
  - What if you train classifier with above data?

    Ex for handling:
    - Blue → 0
    - Red → 1
    - Green → 2

Feature Scaling

- Imagine we have two features
  - $1 < x_1 < 10$
  - $1 < x_2 < 100000$
- Algorithm will likely focus on optimizing w_2 as this will produce the largest changes in perceptron error
- KNN based on Euclidean distance will be dominated by x_2
- Two common approaches
  - Normalization
  - Standartization

Normalization

- Normalization refers to the rescaling of the features to a range of [0, 1]. To normalize the data, we apply the min-max scaling to each feature column, where the new value $x_{norm}^{(i)}$ of a sample $x^{(i)}$ is calculated as follows:

$$x_{norm}^{(i)} = \frac{x^{(i)} - x_{min}}{x_{max} - x_{min}}$$

- Here, $x^{(i)}$ is a particular sample, x_{min} is the smallest value in a feature column, and x_{max} the largest value, respectively.

Standartization

- Normalization gives us values in a bounded interval
- Standartization can be more practical
- Many ML algorithms initialize the weights to zero
- Standartization centers the columns at mean = 0 and std = 1
- So feature columns take the form of a normal distribution
- This makes it easer to learn the weights
- Standartization encodes useful info about outliers
- Vs. normalization which scales the data to a fixed range

- The procedure of standardization can be expressed by the following equation:

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x}$$

- Here, μ_x is the sample mean of a particular feature column and σ_z the corresponding standard deviation, respectively.
- Example of using normalization and standardization

Splitting dataset into train and test

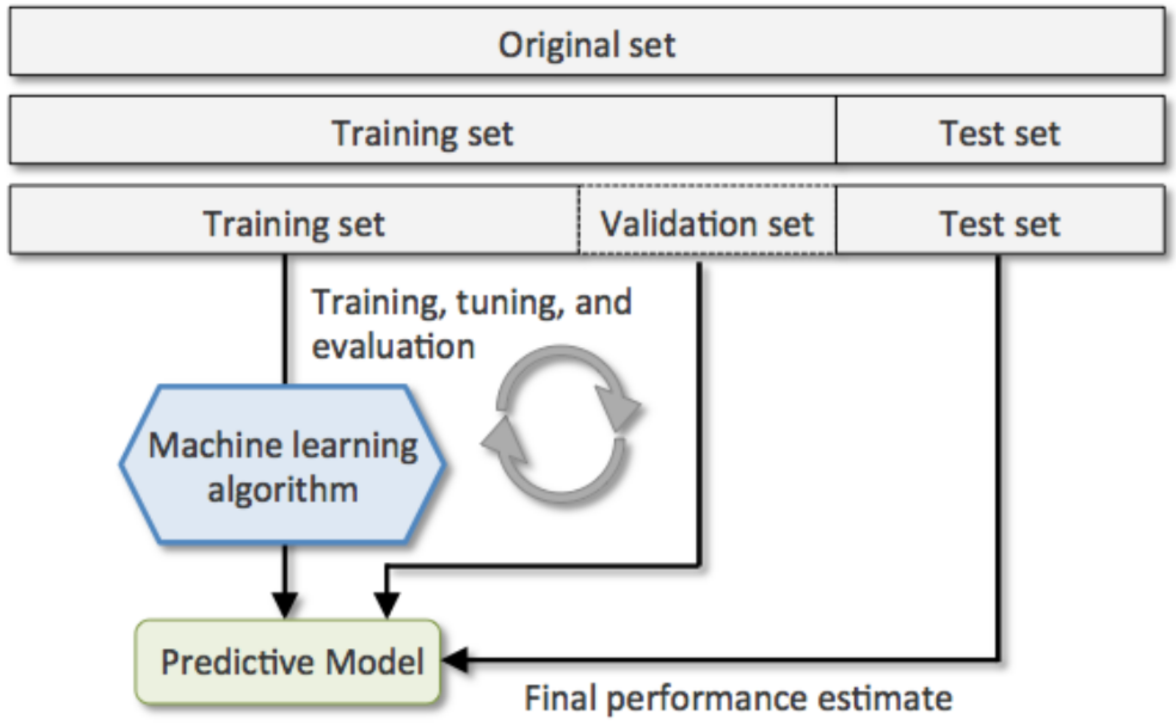
- Cannot train and test on the same dataset
- Train and test split ratio tradeoff
  - 70:30 and 80:20
  - 90:10 and 99:1 for very large datasets

  - Sklearn’s built in functions

The holdout method

- Split data into training and test datasets
- Need to tune the model to further improve the   performance
- Select optimal values of hyperparameters
- This step is known as model selection
- A better approach: training set + validation set + test set
- Validation set is used for model selection

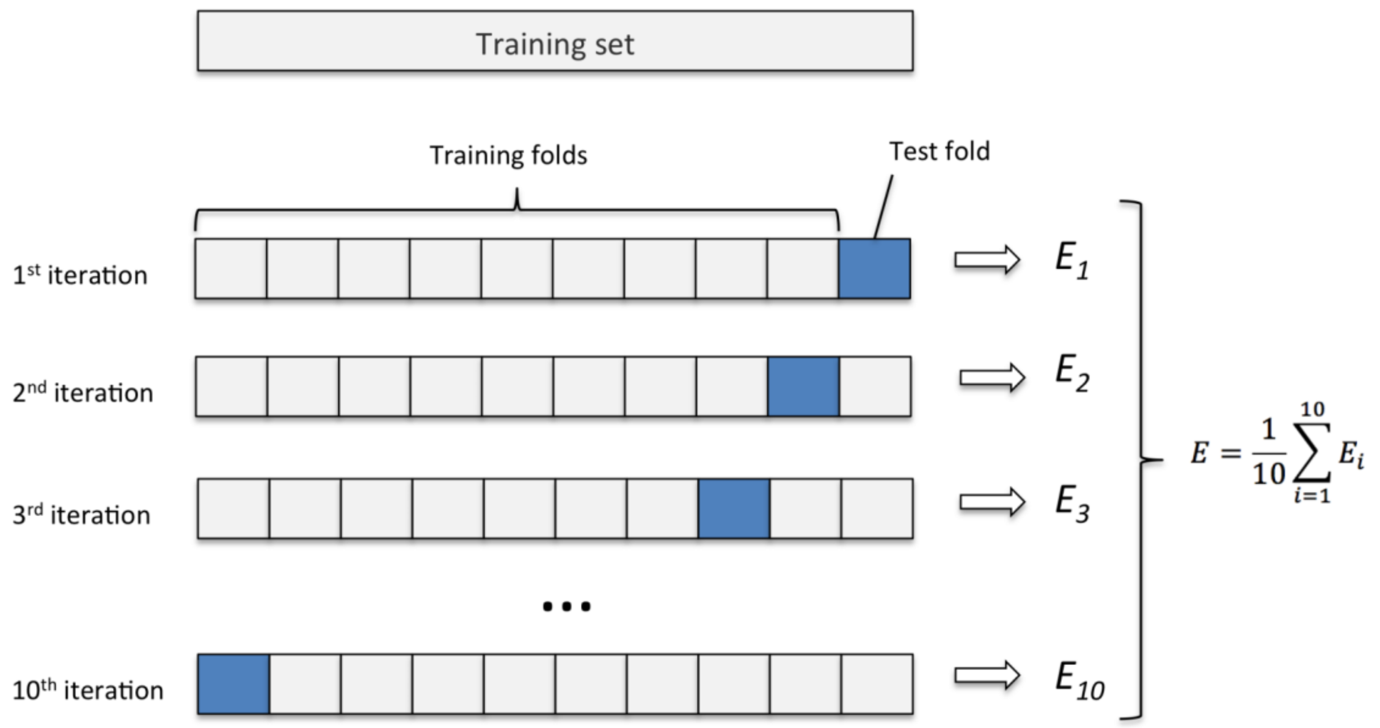
The holdout method


Original set
Training set
Test set
Training set
Validation set
Test set
Training, tuning, and evaluation
Machine learning algorithm
Predictive Model
Final performance estimate

K-fold cross-validation

- Disadvantage of the holdout method: sensitive to partitioning
- Randomly split the training dataset into k folds
- Of these, k - 1 folds are used for training and one for testing
- Repeat this procedure k times and average across k folds
- Each sample will be part of train and test sets
- Lower-variance estimate of the model performance (than holdout)

K-fold cross-validation



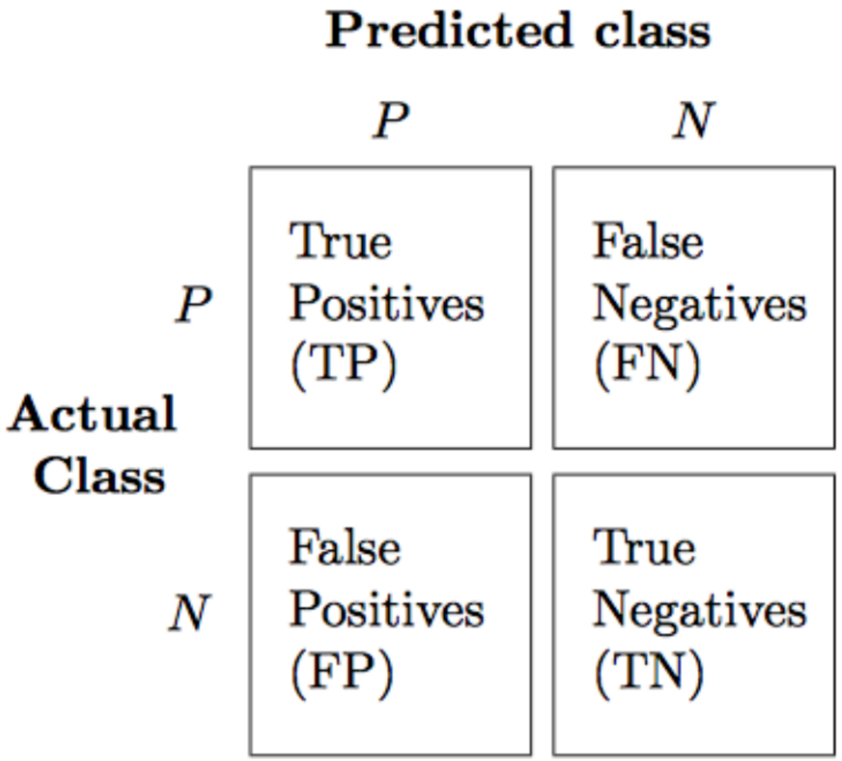
Performance metrics

- Accuracy = (number of correctly predicted)/total

- What are some shortcomings of accuracy?

- Need ways to compute the performance for a specific class
- Confusion matrix helps visualize different types of errors a classier can make by reporting the counts of these errors
- I.e. true positive (TP), true negative (TN), false positive (FP), false negative (FN) predictions

Confusion matrix



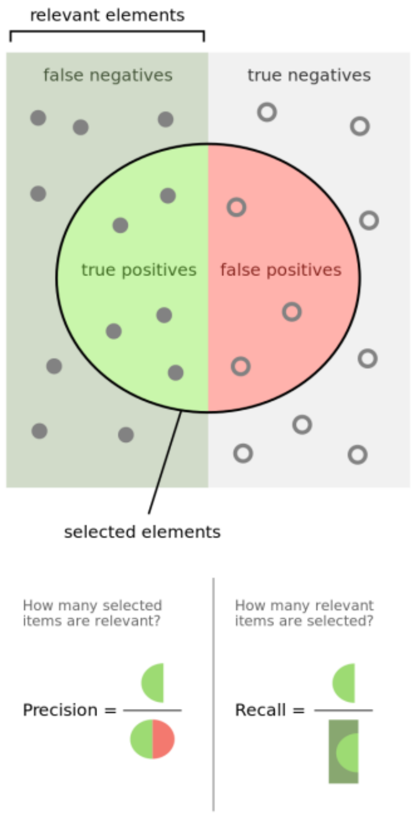
		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Precision, recall

Imagine a scenario when you have M apples and N oranges in a basket. You have a robot whose task is to look into basket and pick out all the apples, leaving the oranges behind.

Precision: number of apples picket out of the basket out of all fruits picked out.

Recall: number of apples picket out of the basket out of all apples (M) that were initially in basket.



Precision, recall and F1 score

Precision:

$$P = \frac{TP}{TP + FP}$$

Recall:

$$R = \frac{TP}{P} = \frac{TP}{FN+TP}$$

F1 score:

$$F1 = 2 \times \frac{P \times R}{P + R}$$