

# Supervised learning

Inteligencia Artificial en los Sistemas de Control Autónomo  
Máster en Ciencia y Tecnología desde el Espacio

Departamento de Automática

## Objectives

1. Extend supervised learning algorithms
2. Apply supervised learning to real-world problems

## Bibliography

- Müller, Andreas C., Guido, Sarah. Introduction to Machine Learning with Python. O'Reilly. 2016

All figures have been taken from

[https://github.com/amueller/introduction\\_to\\_ml\\_with\\_python/blob/master/02-supervised-learning.ipynb](https://github.com/amueller/introduction_to_ml_with_python/blob/master/02-supervised-learning.ipynb)

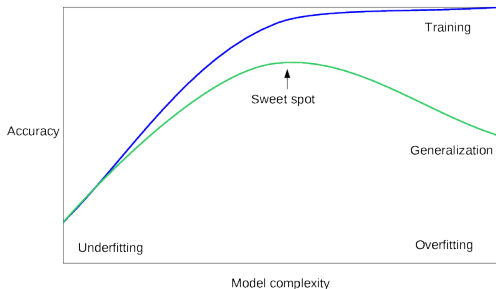
# Table of Contents

1. Generalization, overfitting and underfitting
2. k-Nearest Neighbors
  - k-NN classification
  - kNN regression
  - Summary
3. Linear models
  - Ordinary least squares
  - Linear regression
  - Regularized linear models
  - Ridge regression
  - Lasso regression
  - ElasticNet
  - Regularized linear models comparison
  - Linear models for classification
  - Summary
4. Decision Trees
  - Building decision trees
  - Controlling complexity of decision trees
  - Analyzing decision trees
  - Feature importance in trees
  - Decision trees in regression
  - Summary
5. Ensembles of Decision Trees
  - Ensembles
  - Random forests
  - Analyzing random forests
  - Summary: Random forest
  - Gradient boosted regression trees
  - Summary
6. Support Vector Machines
  - Kernelized Support Vector Machines
  - Summary
  - ARIMA

# Generalization, overfitting and underfitting

Generalization: accurate predictions on unseen data

- i.e. there is no overfitting neither underfitting
- Depends on model complexity and data variability



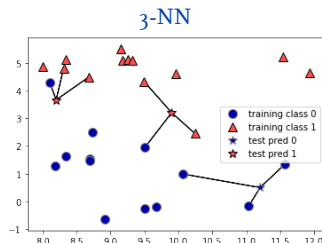
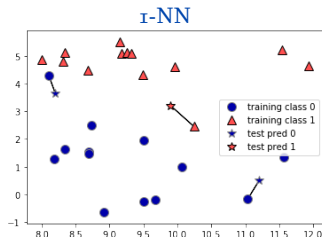
(Source)

# k-Nearest Neighbors

## k-NN classification (I)

k-NN (k-Nearest Neighbors): Likely, the simplest classifier

- Given a data point, it takes its  $k$  closest neighbors
- Same prediction than its neighbors



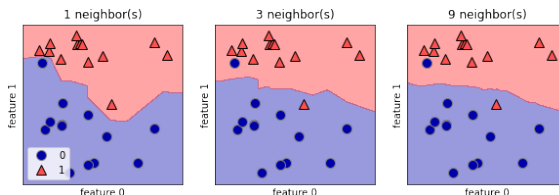
k-NN does not generate a model

- The whole dataset must be stored

$k$  uses to be an odd number (1-NN, 3-NN, 5-NN, ...)

# k-Nearest Neighbors

## k-NN classification (II)



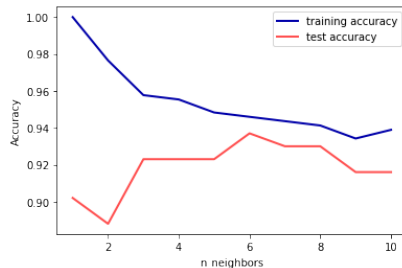
$k$  determines the model complexity

- Smoother boundaries in larger  $k$  values
- Model complexity decreases with  $k$
- If  $k$  equals the number of samples,  $k$ -NN always predicts the most frequent class

How to figure out the best  $k$ ?

# k-Nearest Neighbors

## k-NN classification (III)



# k-Nearest Neighbors

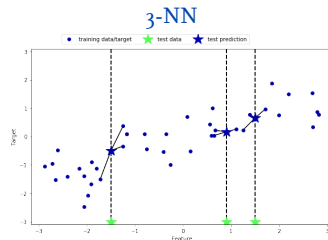
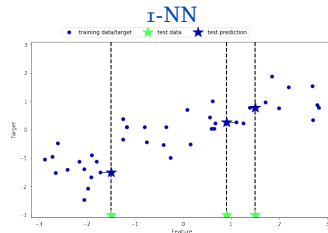
## kNN regression (I)

### k-NN regression

Given a data point

1. Take the  $k$  closest data points
2. Predict same target value (1-NN) or average target value (k-NN)

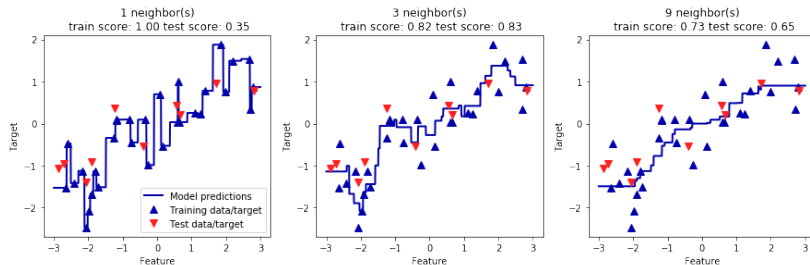
Performance is measured with a regression metric, by default,  $R^2$





# k-Nearest Neighbors

## kNN regression (II)



$k$  determines boundary smoothness

1. With  $k = 1$ , prediction visits all data points
2. With large  $k$  values, fit is worse

# k-Nearest Neighbors

## Summary

Hyperparameters	Advantages	Disadvantages
k	Simple	Slow with large datasets
Distance	Baseline	Bad performance with hundreds or more attributes
		No model
		Dataset must be stored in memory

# Linear models

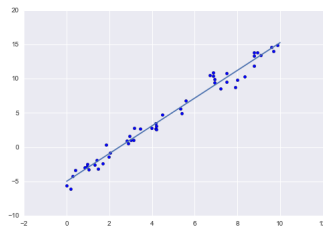
## Linear model (I)

### Linear model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

for a single feature  $y = \beta_0 + \beta_1 x_1$ , where

- $\beta_0$  is the intercept
- $\beta_1$  is the slope
- Interpretable model



Linear models assume a linear relationship among variables

- This limitation can be easily overcome
- Surprisingly good results in high dimensional spaces

# Linear models

## Linear regression

Different linear models for regression

- The difference lies in how  $\beta_i$  parameters are learned

Ordinary Least Squares (OLS): Minimizes mean squared error

- OLS does not have any hyperparameter
- No complexity control

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (x_i - y_i)^2$$

Linear regression can be used to fit non-linear models

- Just adding new attributes

# Linear models

## Regularized linear models

**Regularization:** Term that penalizes complexity

- Added to the cost function
- Linear models remain the same
- Train to minimize cost function and coefficients
- Intercepts are not part of regularization

Three regularizations

- $L_1$  (Lasso regression),  $L_2$  (Ridge regression) and ElasticNet ( $L_1$  and  $L_2$ )

**Lasso ( $L_1$ )**

$$\alpha \sum_j^n |\beta_j|$$

**Ridge ( $L_2$ )**

$$\frac{\alpha}{2} \sum_j^n \beta_j^2$$

**ElasticNet**

$$\alpha \left( \frac{\lambda}{2} \sum_j^n \beta_j^2 + (1 - \lambda) \sum_j^n |\beta_j| \right)$$

# Linear models

## Ridge regression

Ridge regression (or L2 regularization) adds a new term to cost function

$$\text{MSE} + \alpha \sum_{i=1}^n \beta_i^2$$

$\alpha$  controls the model complexity

- If  $\alpha = 0$  Ridge becomes a regular linear regression
- Optimal  $\alpha$  depends on the problem

Ridge by default

# Linear models

## Lasso regression (I)

Lasso regression (or  $L_1$  regularization) adds a new term to cost function

$$\text{MSE} + \alpha \frac{1}{2} \sum_{i=1}^n |\beta_i|$$

$\alpha$  controls the model complexity

- If  $\alpha = 0$  Ridge becomes a regular linear regression
- Optimal  $\alpha$  depends on the problem

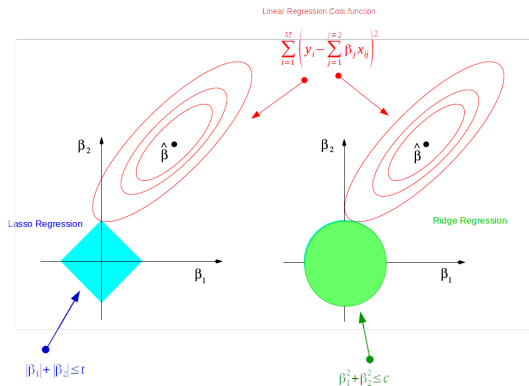
Some coefficients may be exactly zero

- Implicit feature selection
- Easier interpretation
- Better with large number of attributes

# Linear models

## Lasso regression (II)

### Dimension Reduction of Feature Space with LASSO



(Source)



# Linear models

## ElasticNet

Lasso and Ridge can be combined

$$\text{MSE} + \alpha \left( \lambda \frac{1}{2} \sum_{i=1}^n |\beta_i| + (1 - \lambda) \sum_{i=1}^n \beta_i^2 \right)$$

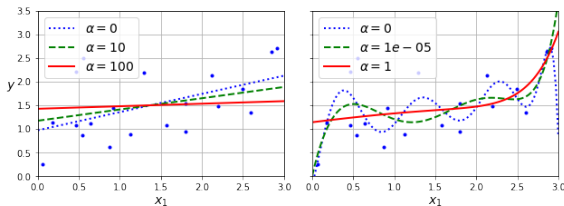
Two hyperparameters

- $\alpha$  controls the model complexity
- $\lambda$  balances between  $L_1$  and  $L_2$

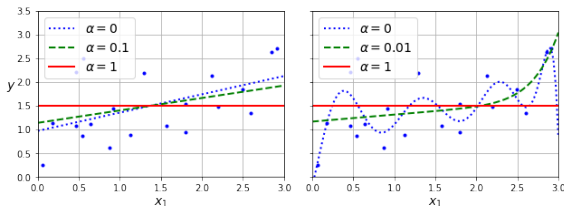
# Linear models

## Regularized linear models comparison

### Ridge - L2



### Lasso - L1



# Linear models

## Linear models for classification (I)

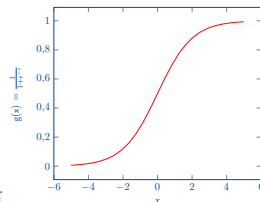
A linear regression can be used as classifier

- Just compare the prediction with a threshold (0, for instance)
  - If  $\hat{y} > 0$ , assign class 1
  - If  $\hat{y} \leq 0$ , assign class -1
- The decision boundary for any binary linear classifier is a line, plane or hyperplane

A **logistic regression** is a generalization of a linear regression

- It is a binary classifier
- Its output is a probability

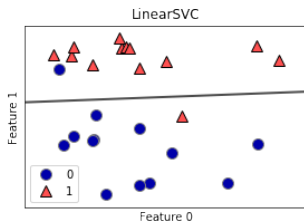
$$\hat{p} = \sigma \left( \beta_0 + \sum_{i=1}^n \beta_i x_i \right),$$



where  $\sigma(t)$  is the logistic function, defined as  $\sigma(t) = \frac{1}{1+e^t}$

# Linear models

## Linear models for classification (II)

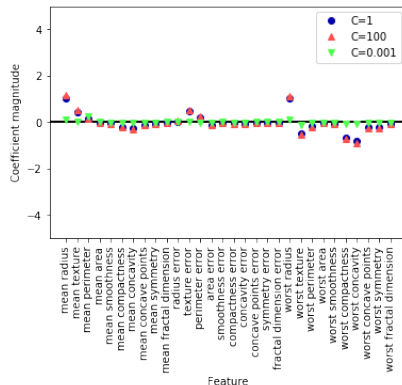


# Linear models

## Linear models for classification (III)

The model can be regularized with  $L_1$ ,  $L_2$  and ElasticNet

- In Scikit-Learn, regularization strength is given by  $C$
- Lower values of  $C$  correspond to smaller regularization strength



# Linear models

## Summary

Hyperparameters	Advantages	Disadvantages
-	Fast train and predict	No complexity tuning
$\alpha$ (L1, L2, ElasticNet)	Scales well to large data-sets	Limited in low dimensional spaces
l1_ratio (ElasticNet)	Better in high dimensional spaces Few hyperparameters Interpretable	

Better when the number of features is large compared to the number of samples

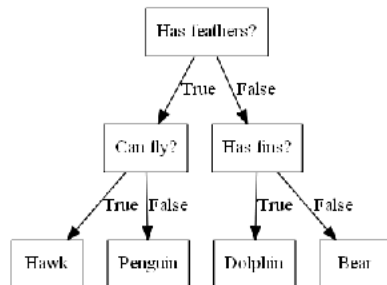
# Decision Trees

Decision trees are a family of algorithms for classification and regression

- They learn a tree data structure
- Hierarchy of if/else questions (test, or node)
- Decision (terminal node or leaf)

Usually, datasets does not contain binary attributes

- Continuous features
- Is feature  $i$  larger than value  $a$ ?



# Decision Trees

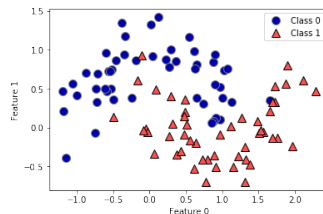
## Building decision trees (I)

### Tree learning algorithm

1. Begin with the root node
2. Searches all possible tests (according to a purity measure)
3. The most informative test is taken
4. Repeat recursively

### Prediction of a new data point

- Classification: Majority class in the partition
- Regression: Average value of target values in the partition

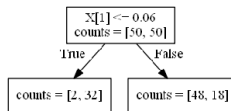
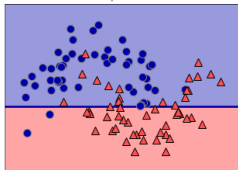




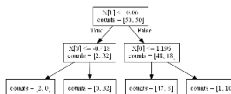
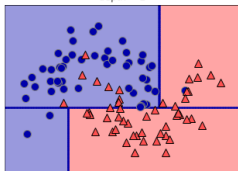
# Decision Trees

## Building decision trees (II)

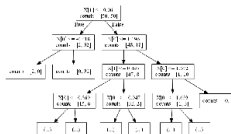
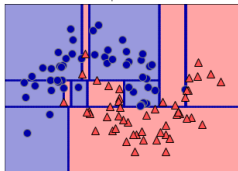
depth = 1



depth = 2



depth = 9



# Decision Trees

## Building decision trees (III)

Let  $p_{mk}$  be the proportion of class  $k$  in node  $m$ , and  $Q_m$  the data in node  $m$

Gini

$$G(Q_m) = \sum_k p_{mk}(1 - p_{mk})$$

Log Loss or Entropy

$$H(Q_m) = - \sum_k p_{mk} \log(p_{mk})$$

# Decision Trees

## Controlling complexity of decision trees

Trees tend to grow until all leaves are pure

- Very big trees in real problems
- Big trees use to be overfitted models

Two strategies to prevent overfitting

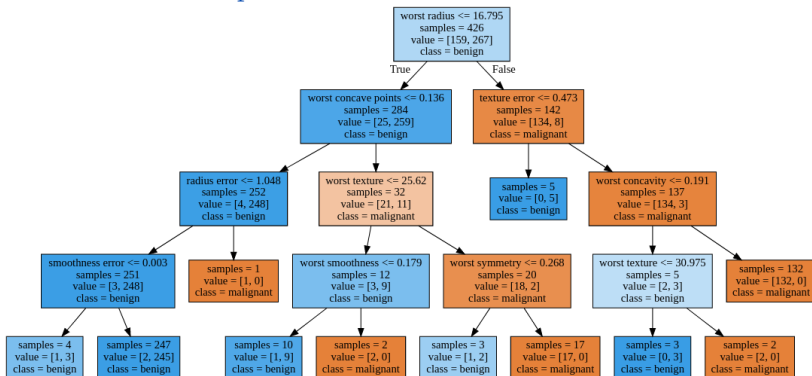
- Pre-pruning: Stop the creation of the tree early according to some criteria
  - Maximum depth, number of leaves, minimum number of points in a node, ...
  - Implemented in Scikit-Learn
- Post-pruning: Build the tree and then remove nodes with little information

# Decision Trees

## Analyzing decision trees

Decision trees is easily explained to nonexperts

- Interpretable models
- Deep trees are overwhelming
- Trick: Observe the path with most data

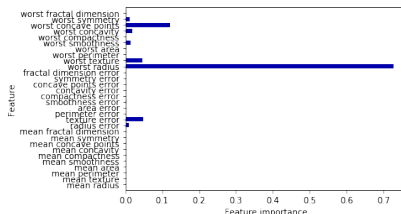


# Decision Trees

## Analyzing decision trees

**Feature importance** is a metric that summarizes features

- Number between 0 (not used at all) and 1 (perfect prediction)
- Feature importances sum to one
- Useful for feature selection and model interpretation



Some considerations

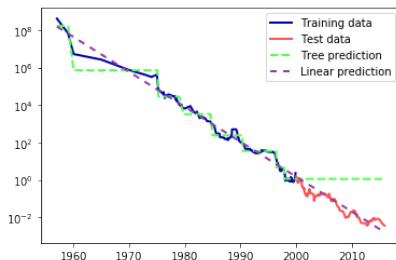
- It does not inform about the relationship between attribute and target
- It quantifies the importance in the tree
  - Correlated attributes may score low importance

# Decision Trees

## Decision trees in regression

Decision trees are not able to extrapolate

- i. e. to predict outside of the range of the training data
- It is specially important in regression problems



# Decision Trees

## Summary

Hyperparameters	Advantages	Disadvantages
max_depth	Visualization	Tend to overfit
max_leaf_nodes	Interpretable by non-experts	Poor generalization
min_samples_leaf	Invariant to scale	
criterion	Mix of categorical and numerical data	

# Ensembles of Decision Trees

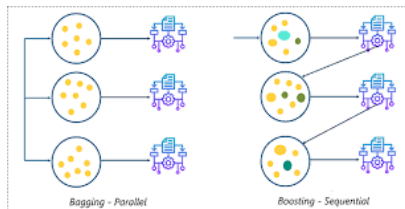
## Ensembles

**Ensembles**, in ML, refers to the combination of several models

- For instance, an ensemble of three classifiers voting

Two common aprochaes to build ensembles

- **Bagging** (or bootstrap) samples the dataset with replacement
  - The ensemble make prediction by aggregating its predictors
- **Boosting** trains models to correct previous models



(Source)



# Ensembles of Decision Trees

## Random forests

Trees have poor generalization

- A tree is good doing his job, but does not generalize well
- Different trees could overfit in different ways
- Idea: Using many trees and averaging their result

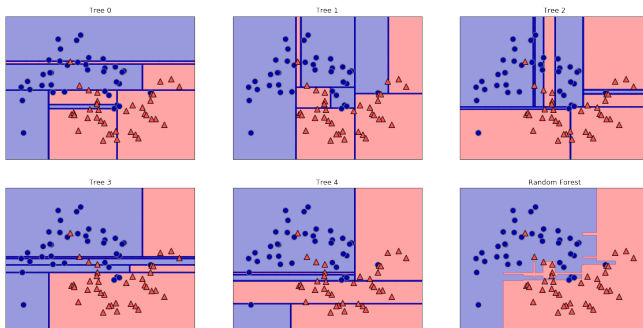
**Random forest** is an algorithm that trains different trees injecting randomness

- Selecting data - bootstrap
- Selecting features in each test
  - It does not look for the best test
  - It looks for the best test involving a random subset of features
  - The size of the features subset is a critical hyperparameter

# Ensembles of Decision Trees

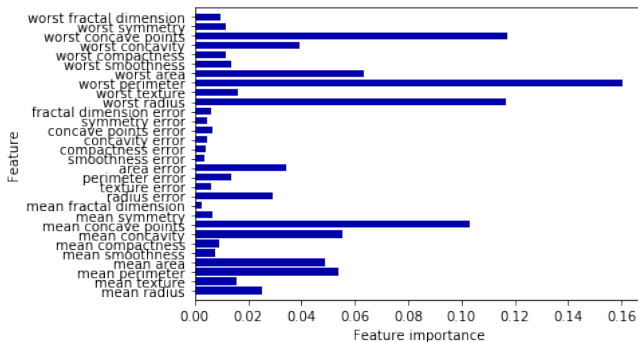
## Analyzing random forests (I)

### Random forest with five trees



# Ensembles of Decision Trees

## Analyzing random forests (II)



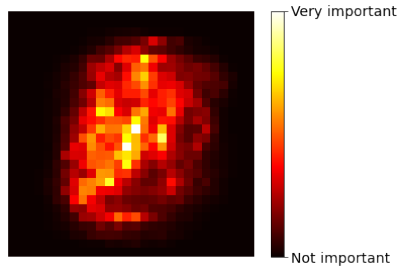
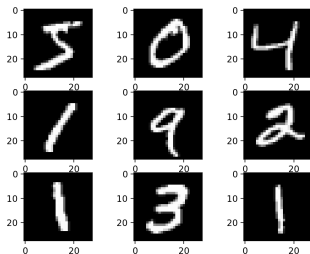
Feature importance can be aggregated

- More informative than single trees
- The algorithms must consider many possible explanations

# Ensembles of Decision Trees

## Analyzing random forests (III)

Random forest classifier with MNIST dataset



# Ensembles of Decision Trees

Summary: Random forest

Hyperparameters	Advantages	Disadvantages
Same than trees	Same than trees	Interpretation
Number of trees	High performance	High dimensional data
Pre-pruning	Robust	Sparse data
	Widely used	Memory and CPU
	Parallelized	

# Ensembles of Decision Trees

## Gradient boosted regression trees (I)

**Gradient boosting** is an ensemble of classification and regression trees

- Based on boosting, builds trees in a serial manner
- One tree corrects the mistakes of the previous one

A set of *weak learners* is used

- Shallow trees (by default, 3 in Sklearn)
- No data randomization
- Strong pre-pruning

A new hyperparameter: learning rate

- How strongly each tree tries to correct
- High learning rate makes stronger corrections
  - More complex models
- More trees also adds more complexity

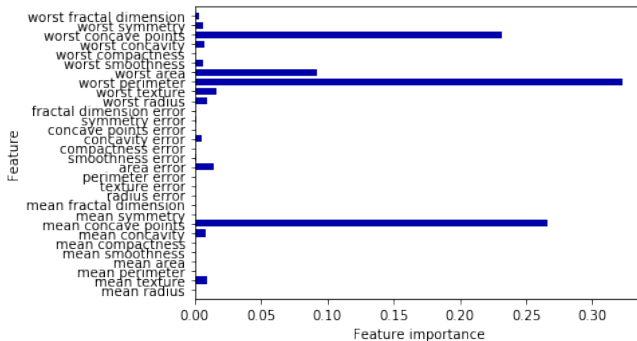
State of the art results

- Widely adopted by industry
- Comparable in performance with deep neural networks

# Ensembles of Decision Trees

## Gradient boosted regression trees (II)

Feature importances with cancer dataset



The (XGBoost) package provides a high performance implementation of gradient boosted trees

# Ensembles of Decision Trees

## Summary

Hyperparameters	Advantages	Disadvantages
Same than trees	Very high performance	Slow
Number of trees	Invariant to scale	High dimensional data
Learning rate	Mix of categorial and numerical data	Tricky hyperparameter tuning
		Overfitting



# Support Vector Machines

TODO

# Support Vector Machines

## Kernelized Support Vector Machines

TODO

# Support Vector Machines

## Summary

---

Hyperparameters	Advantages	Disadvantages
-----------------	------------	---------------

---

# Algorithms

## ARIMA (I)

### AR: Autoregressive model

- Current observation depends on the last  $p$  observations
- Long term memory

AR( $p$ )

$$X_t = c + \sum_{i=1}^p \phi_i X_{t-1} + \epsilon_t$$

### MA: Moving Average model

- Current observation linearly depends on the last  $q$  innovations
- Short term memory

MA( $q$ )

$$X_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \dots + \theta_q \epsilon_{t-q}$$

ARMA model = AR + MA

- ARMA( $p, q$ ): Two hyperparameters,  $p$  and  $q$

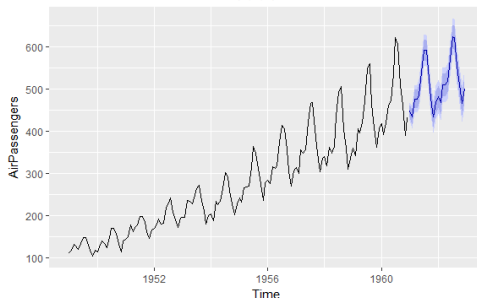
# Algorithms

## ARIMA (II)

ARIMA = AR + i + MA (AR integrated MA)

- ARIMA(p, d, q)
- Three integer parameters: p, q and d (in practice, low order models)

Forecasts from STL + ARIMA(1,1,1) with drift



(Source)

**autoarima:** search over p, q and d