Linear Regression

N data points

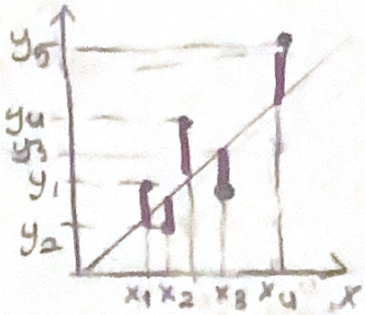
(x_1, y_1)
(x_2, y_2)
⋮
(x_N, y_N)

$\Big\}$ $y = \beta_0 + \beta_1 x_i + \varepsilon_i$

$E[\varepsilon_i] = 0$
$V[\varepsilon_i] = \sigma^2$

ε_i ← error term

Assume it's a RV, distributed normally.



↳ least squares

min $\sum \varepsilon_i^2$ by changing β_0 & β_1

* $\hat{\beta}_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = \frac{S_{xy}}{S_{xx}}$

$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$

* $S_{xy} = \sum x_i y_i - \frac{\sum x_i * \sum y_i}{N}$

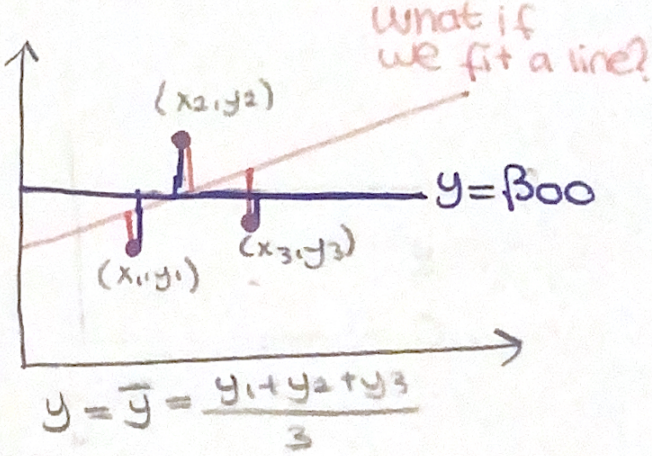
* $S_{xx} = \sum x_i^2 - \frac{(\sum x_i)^2}{N}$

↳ Do the fit!

Goodness of Fit

$y = \beta_0 + \beta_1 x$ → Our fit

$y = \beta_{00}$ → what else could be done?



* $SST = \sum (y_i - \bar{y})^2$

↓ sum of squares of total

* $SSE = \sum (y_i - \beta_0 - \beta_1 x_i)^2$ → $0 \leq \frac{SSE}{SST} \leq 1$

↳ Bad fit

↳ Burun hatası daha çok olmalı

$R^2 = 1 - \frac{SSE}{SST}$

Tree Based Models

Decision Trees

- Splits → GINI & entropy
  Uses greedy approach
- Methods to prevent overfitting
  - Pruning
  - Bagging

Pros → Explainable Cons → None

Ensembling

- Voting → Parallel Ensemble
  (Average for regression, majority vote for classification)
- Stacking → use serial models
  Serialize models to understand errors in previous models

Ensemble + Trees = Random Forest

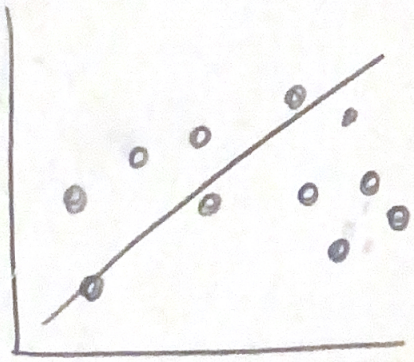
(Parallel)

- Uses Bagging, decreases variance, random sampling from original data.
  Uses randomness: random variables, random bags
  #trees ↗ overfitting ↗

Serial Ensemble + Trees = Gradient Boosting

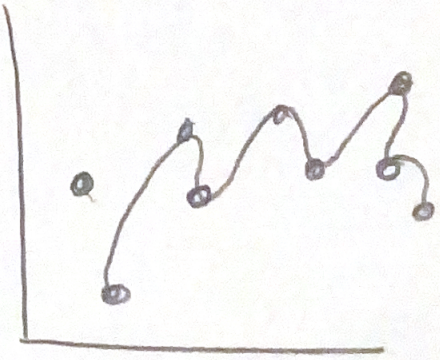
Light GBM, AdaBoost, XG Boost, Cat Boost

Variance - Bias Tradeoff



High Bias
Underfitting

US



High Variance
Overfitting

Subset Selection

Forward Subset Selection
- Start with most explaining variable
- Add variables until it doesn't improve
- Prevents overfitting

Backward Subset Selection
- Start with all variables
- Remove until no improvement
- Expected to perform better than forward (worse when noise is too much)

Forward & Backward
- Add or remove in any of steps

Mix Stepwise Selection
- Let the algorithm try every method til it finds a local optimal variable base

Advanced Selections
- Ensemble selection
- Use multiple algorithms
- Use a starting pool

Subset Selection Penalty Metrics
- C_p → find min
- BIC → find min
- AIC → uses MLE
- Adjusted R^2 → larger the better

Shrinkage
- Fit a model with p predictors and shrink their coefficients to zero.
- Shrinking reduces variance (prevents overfitting)

Ridge (L2)
$\text{Penalty} = RSS + \lambda \sum_{j=1}^{p} \beta_j^2$

λ: tuning param for regularization

small when $\beta_j \approx 0$

shrinkage penalty

Disadvantage → none of predictor coefficients go zero

Lasso (L1)
$\text{Penalty} = RSS + \lambda \sum_{j=1}^{p} |\beta_j|$

Some coefficients go completely zero (like subset selection) if λ is sufficiently large

Yields sparse models

ElasticNet (L1+L2)
Lasso + Ridge

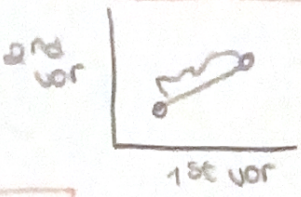
Can be given weights

→ No free lunch!

Unsupervised Learning

Clustering → Flat (Centroid, k-means)
↳ Hierarchical (Bottom up, agglomerative)
Top down, divisive
↓
Density (Dense regions, DBScan)

Distance Metric: Euclidean Distance



- Variables might be on different scales
- Convert to z score or standardize

$$\frac{\#\text{vars with matching value for samples}}{\#\text{vars}}$$

Matching coefficient
- Used when clustering observations are 0 or 1
- Count number of variables with matching values

Jaccard's Coefficient
measure doesn't count matching zero entries

Centroid Algorithms

* K-means
1. Pick a k
2. Initialize k points (means)
3. Categorize each point to closest mean
4. Repeat until clusters don't change

- Can't handle outliers (use k-medoids)
- Features are needed to be scaled

* K-medoids
Uses real data points instead of means

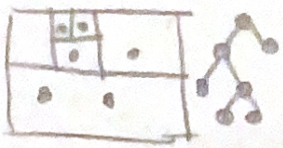
Centroid based ones cannot handle arbitrary shaped clusters

Density-Based

* DBSCAN
- Can handle arbitrary shaped clusters
- Algorithm checks number of points near a point
* If #points > threshold, accept the point as core point
* Core points and border points form a cluster together

Hierarchical-Clustering

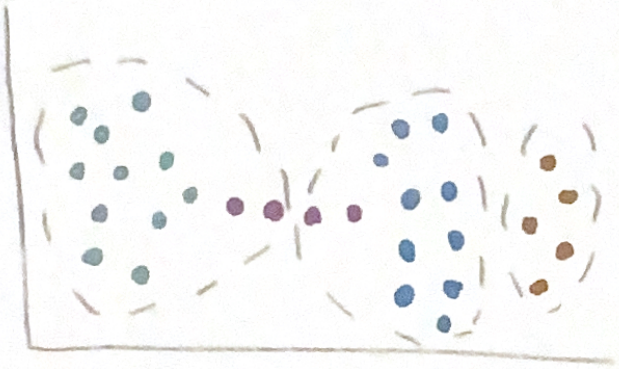
- Agglomerative
- Divisive
- Isolation Forest



Isolates anomalies

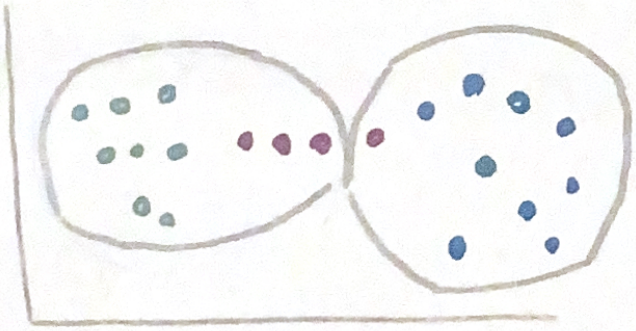
Measuring Distance Between Two Clusters

Single Link: Distance between two clusters is the distance between two closest data points in the two clusters. It can find arbitrary shaped clusters, but it may cause a chain effect with noise.



→ Disadvantage of Single Link

Complete Link: Distance between two clusters is the distance between two furthest data points in two clusters.

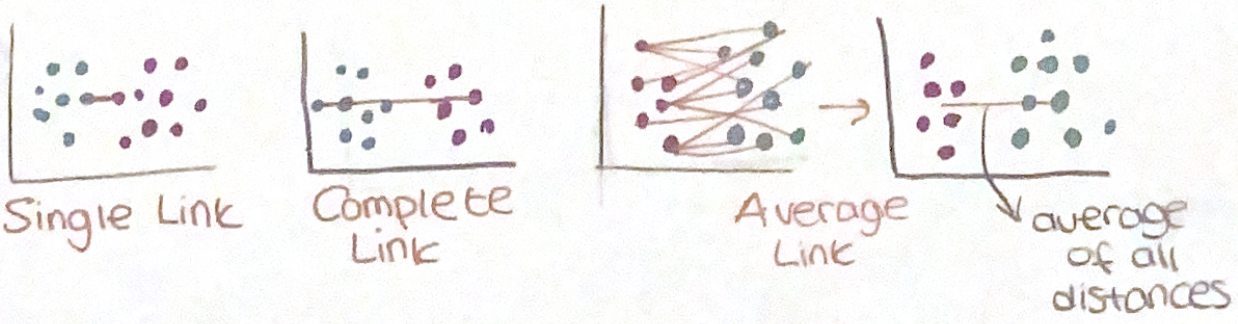


Average Link: Provides a trade-off between two of the above.

→ Complete link is sensitive to outliers

→ Single link forms chains do not correspond to clusters

* Distance between two clusters is the average distance of all pairwise distance between data points in two clusters.



K-Means Algorithm

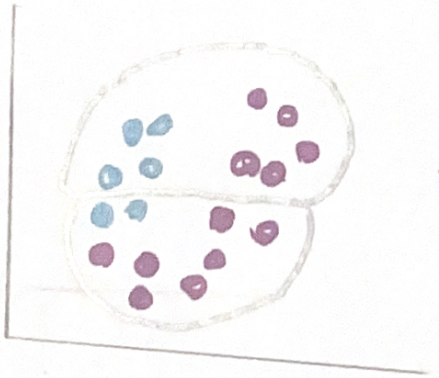
1) Randomly choose k data points as initial centroids
2) Assign each data point to the closest centroid
3) Recompute centroids
4) Do this (2-4) until convergence criterion is met

* Time complexity: $O(tkn)$
  - t = #iterations
  - k = #clusters
  - t & k is so small that this is considered linear

* Algorithm is sensitive to outliers

Sensitive to outliers

Sensitive to initial seeds



→ Performs well on hyperspheres but not on clusters with no hyperspheres

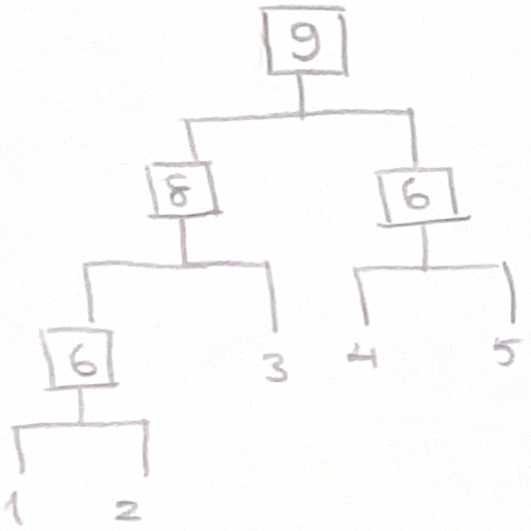
Use hierarchical clustering!

Unsupervised Learning

Hierarchical Clustering

Agglomerative (bottom up): Build the dendrogram (tree) from bottom level, merge most similar pair of clusters. Stop when all data points are merged into a single cluster

Divisive (Top Down): Start with all data points in one cluster Split it into child clusters Divide them recursively. Stop when each cluster is only one data point.



Algorithm (AGglomerative) (D)

Make each data point D a cluster.

Compute all pairwise distances $x_1, x_2 \dots, x_n$

Repeat:

Find two clusters that are nearest

Merge them to new cluster c

Compute distance from c to all other clusters

Until

There's only one cluster

Distance Functions

Minkowski Distance

$$dist(x_i, x_j) = ((x_{i1} - x_{j1})^n + x_{i2} - x_{j2})^n + \ldots)^{\frac{1}{n}}$$

(x_i, x_j: vectors)

if $n = 1 \rightarrow$ Manhattan Distance
$n = 2 \rightarrow$ Euclidean Distance

Distance Functions for Binary Attributes (& Nominal)

Use a confusion matrix
ith & jth data points be x_i & x_j (vectors)

Data point i \ Data point j	1	0	
1	a	b	a+b
0	c	d	c+d
	a+c	b+d	a+b+c+d

a: # Number of attributes with value 1 for both data points

b: # attributes for x_i $f=1$ & x_j $f=0$ (value of fth attribute in data point x_i)

c: # attributes for x_i $f=0$ x_j $f=1$

The distance becomes:

$$dist(x_i, x_j) = \frac{b+c}{a+b+c+d}$$

Nominal Attributes:

Colors: Red, Blue, Yellow

1 2 3 ← Simply assigning numbers

1 → 2 → Distance } this is wrong

Determine the number of values that will match in vectors.

Data points → x_i x_j

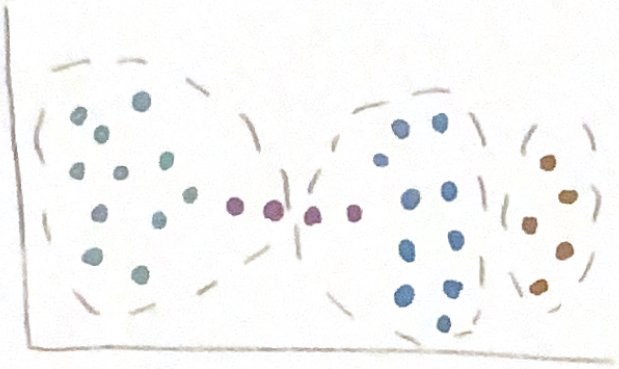
#attributes = r

values that match in x_i & x_j = q

$$Distance(x_i, x_j) = \frac{r-q}{q}$$

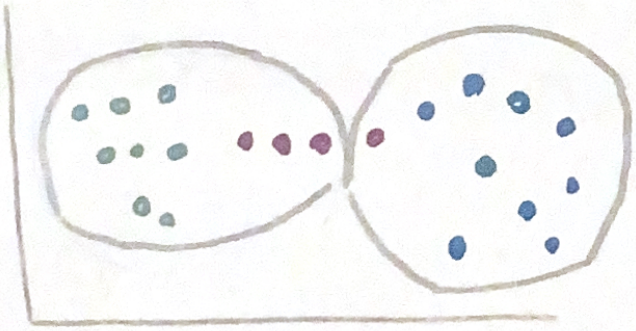
Measuring Distance Between Two Clusters

Single Link: Distance between two clusters is the distance between two closest data points in the two clusters. It can find arbitrary shaped clusters, but it may cause a chain effect with noise.



→ Disadvantage of Single Link

Complete Link: Distance between two clusters is the distance between two furthest data points in two clusters.



Average Link: Provides a trade-off between two of the above.

→ Complete link is sensitive to outliers

→ Single link forms chains do not correspond to clusters

* Distance between two clusters is the average distance of all pairwise distance between data points in two clusters.