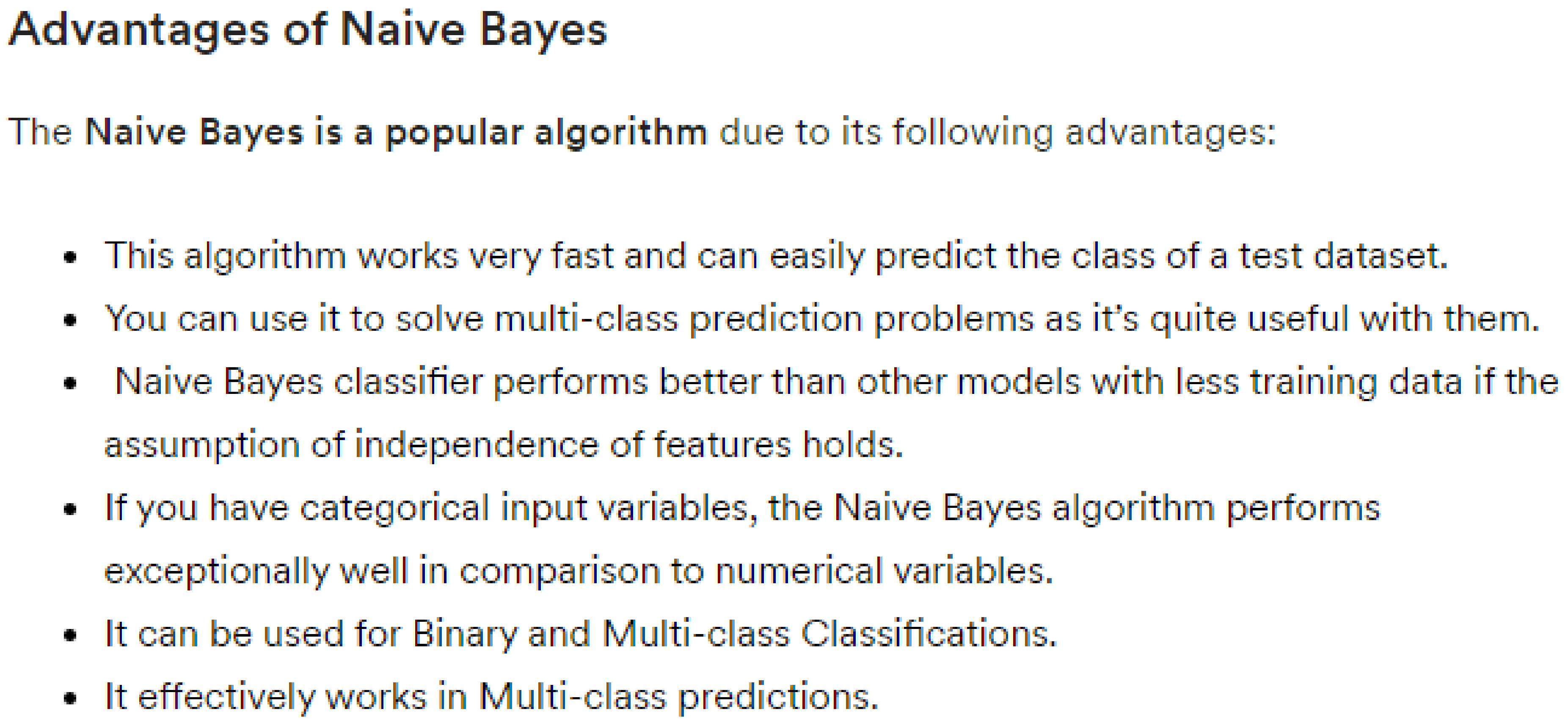
BİL 475 Örüntü Tanıma

Hafta-6:

Başarım Metrikleri

Doğrusal Regresyon

- Bayes Sınıflandırıcısı +/-



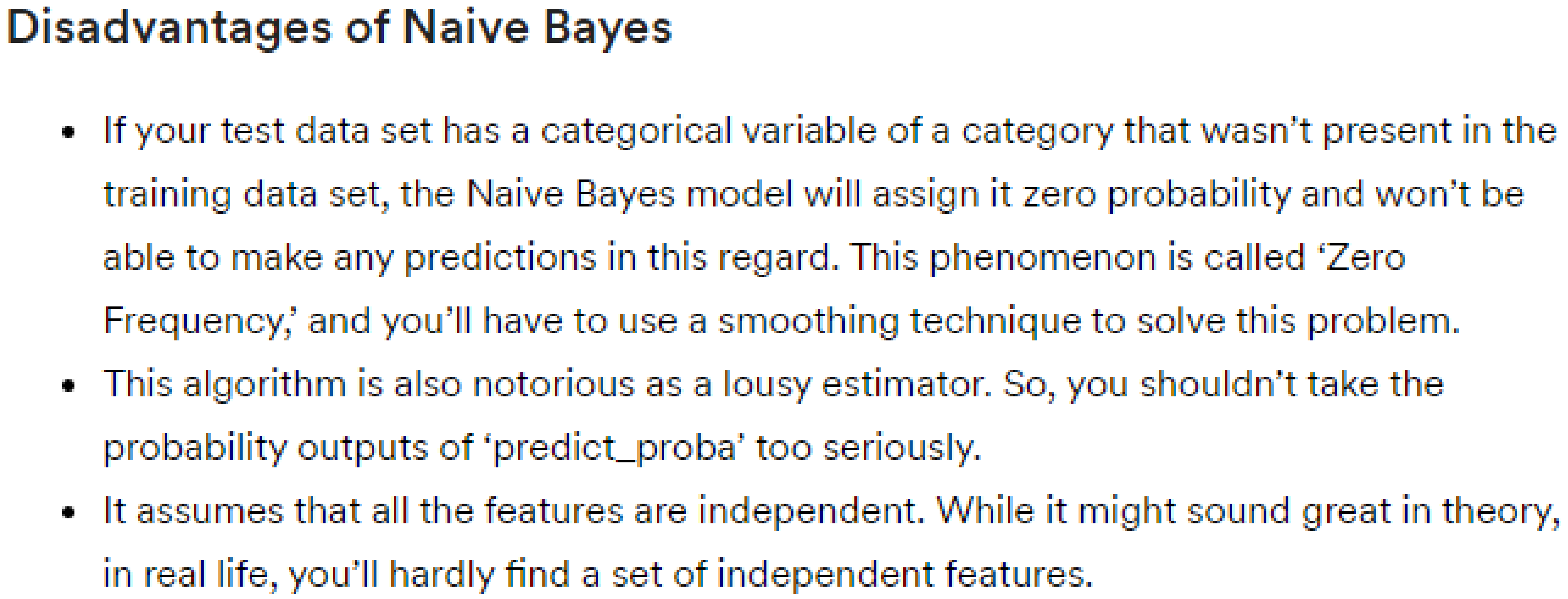
Advantages of Naive Bayes

The **Naive Bayes is a popular algorithm** due to its following advantages:

- This algorithm works very fast and can easily predict the class of a test dataset.
- You can use it to solve multi-class prediction problems as it's quite useful with them.
- Naive Bayes classifier performs better than other models with less training data if the assumption of independence of features holds.
- If you have categorical input variables, the Naive Bayes algorithm performs exceptionally well in comparison to numerical variables.
- It can be used for Binary and Multi-class Classifications.
- It effectively works in Multi-class predictions.

https://www.upgrad.com/blog/naive-bayes-classifier/

- Bayes Sınıflandırıcısı +/-



Disadvantages of Naive Bayes

- If your test data set has a categorical variable of a category that wasn't present in the training data set, the Naive Bayes model will assign it zero probability and won't be able to make any predictions in this regard. This phenomenon is called 'Zero Frequency,' and you'll have to use a smoothing technique to solve this problem.
- This algorithm is also notorious as a lousy estimator. So, you shouldn't take the probability outputs of 'predict_proba' too seriously.
- It assumes that all the features are independent. While it might sound great in theory, in real life, you'll hardly find a set of independent features.

https://www.upgrad.com/blog/naive-bayes-classifier/

x1	x2	x3	C
1	0	4	0
2	0	2	0
1	1	1	0
1	4	2	0
2	3	4	0
3	2	3	0
2	1	2	0
1	2	0	1
2	3	0	1
3	4	1	1
3	3	2	1
2	1	3	1
2	3	2	1
3	2	1	1

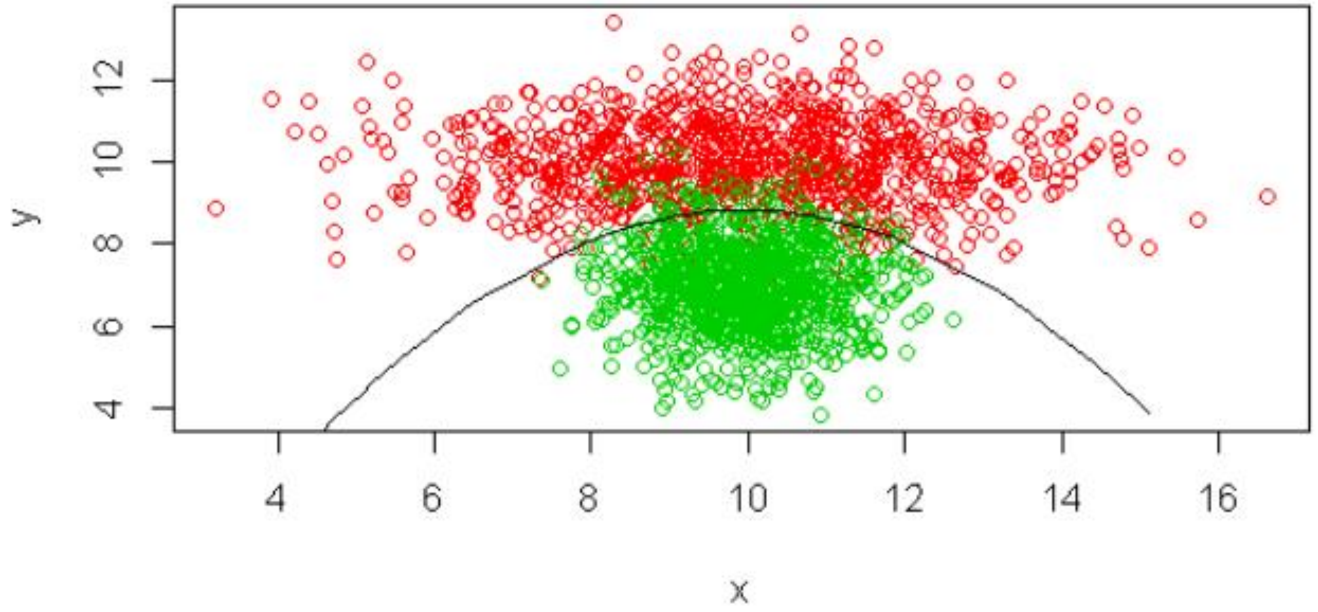
Algorithm

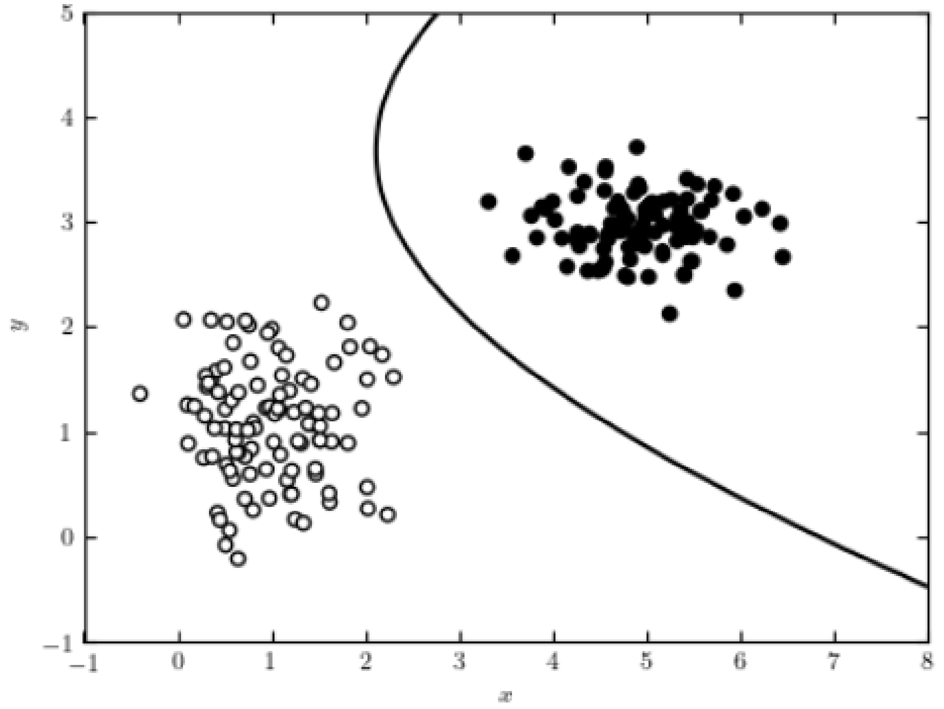
Bayes theorem provides a way of calculating the posterior probability, $P(c/x)$, from $P(c)$, $P(x)$, and $P(x/c)$. Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.

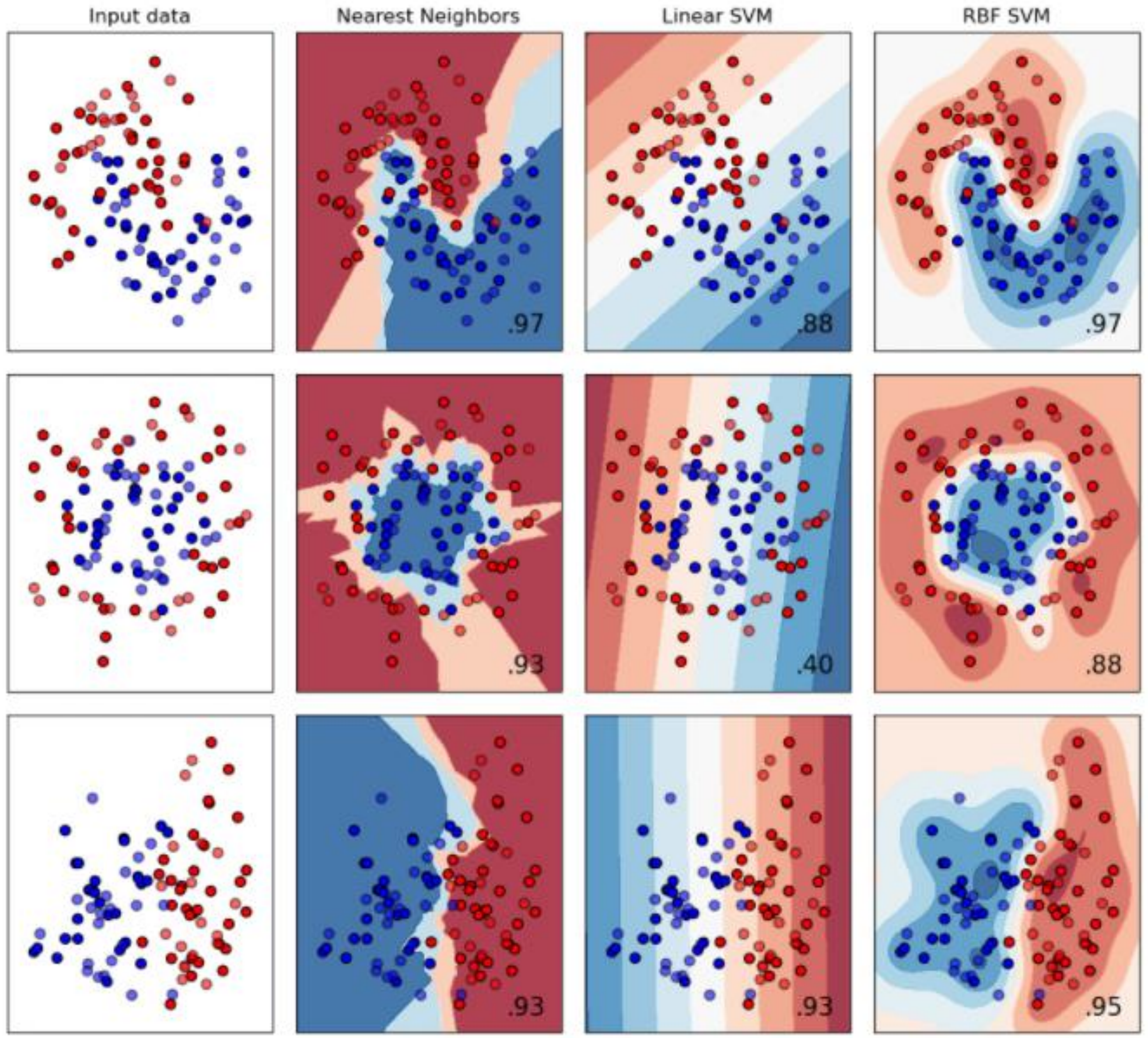
$$\underbrace{P(c\,|\,x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x\,|\,c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

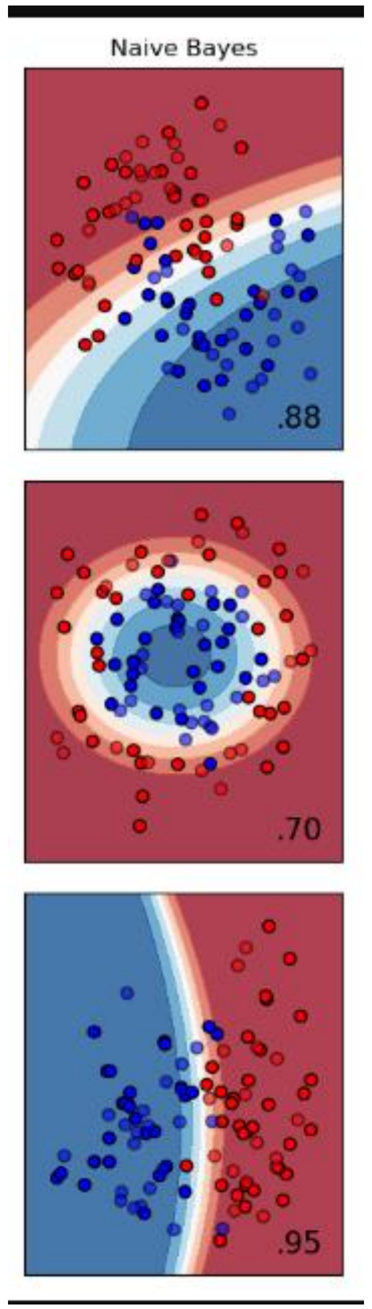
$$P(c\,|\,\mathrm{X}) = P(x_1\,|\,c)\times P(x_2\,|\,c)\times\cdots\times P(x_n\,|\,c)\times P(c)$$

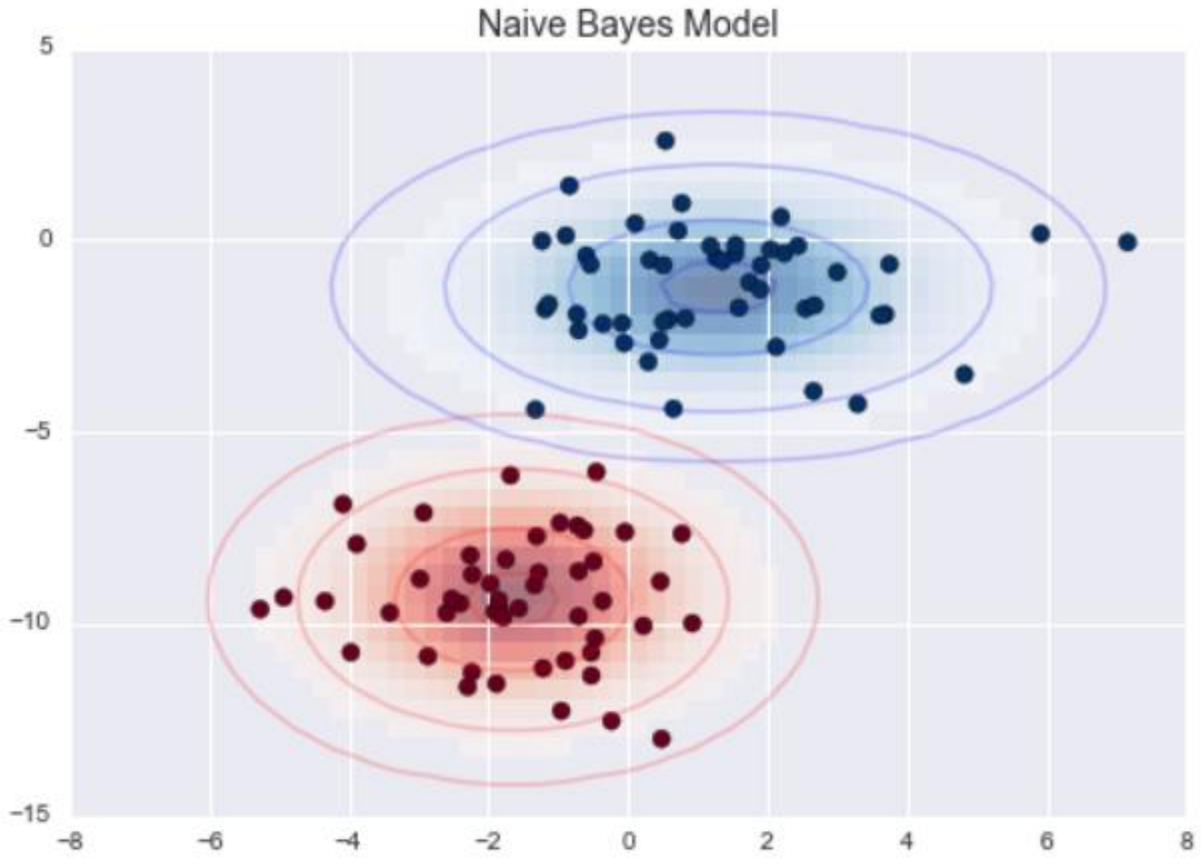
- $P(c/x)$ is the posterior probability of *class* (*target*) given *predictor* (*attribute*).
- $P(c)$ is the prior probability of *class*.
- $P(x/c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

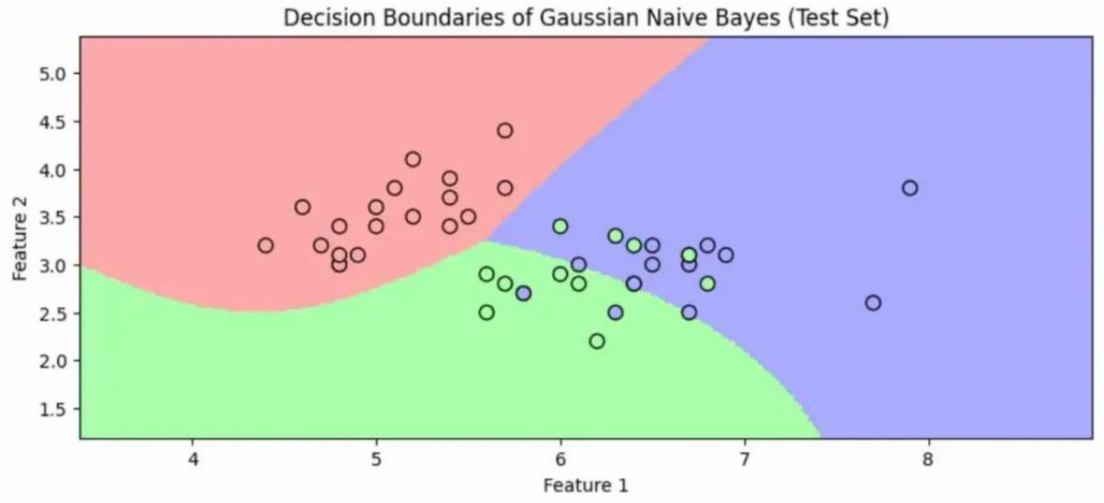

y
x
4
6
8
10
12
14
16


y
x
−1
0
1
2
3
4
5
6
7
8


Input data
Nearest Neighbors
Linear SVM
RBF SVM
.97
.88
.97
.93
.40
.88
.93
.93
.95

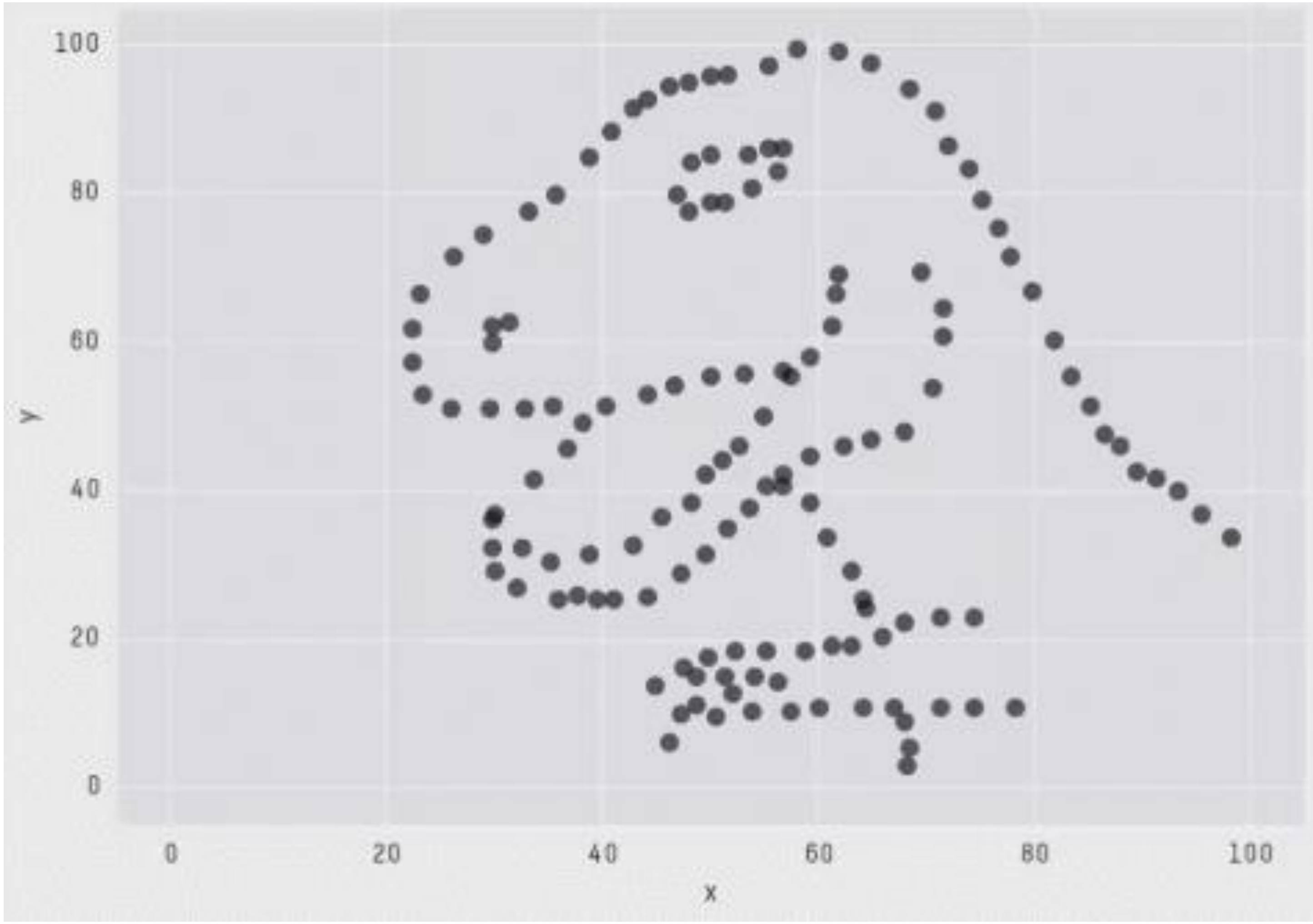

Naive Bayes
.88
.70
.95





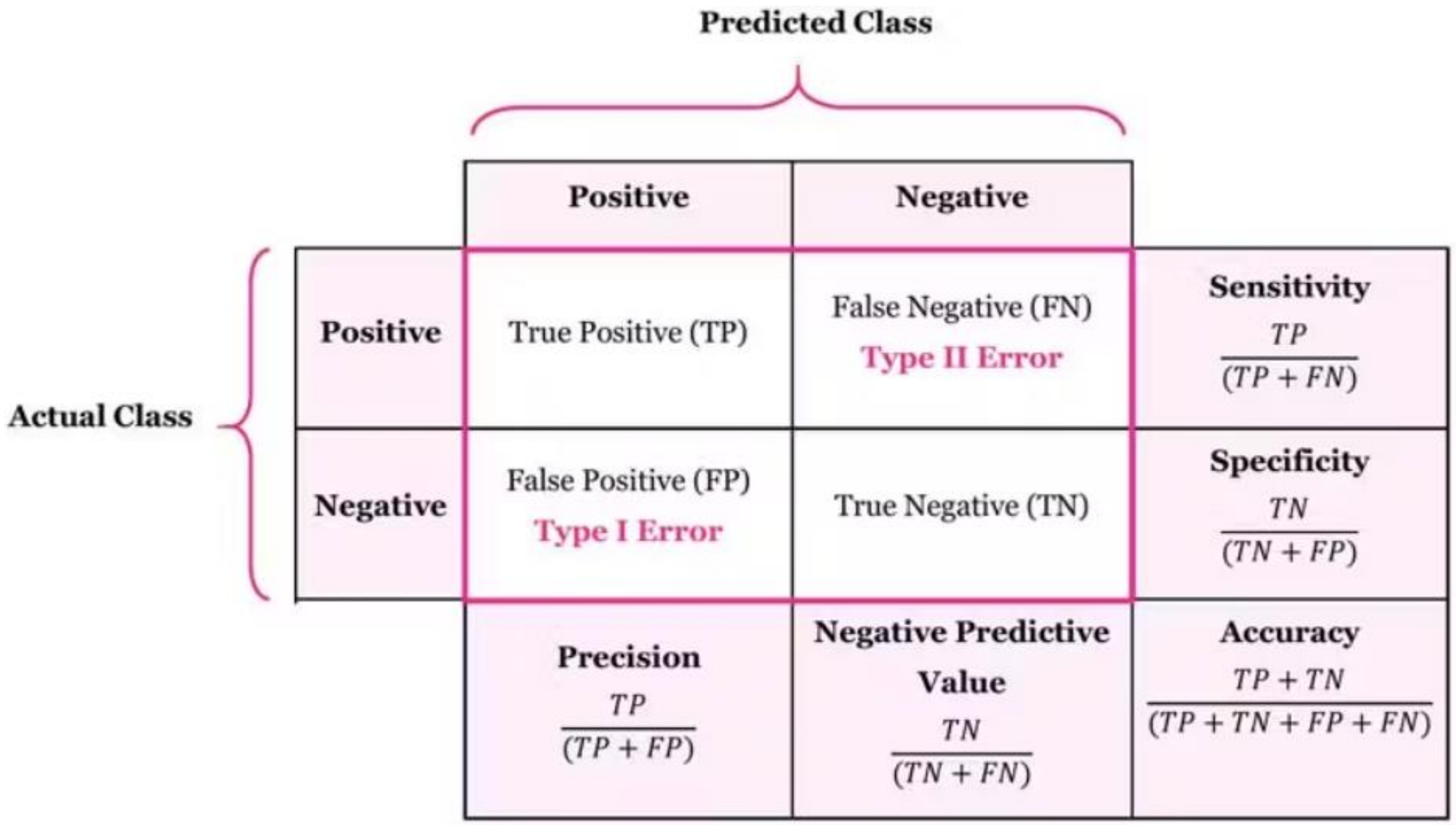
https://spotintelligence.com/2024/05/31/naive-bayes-classification/

https://jakevdp.github.io/PythonDataScienceHandbook/05.05-naive-bayes.html



X Mean: 54.2659224
Y Mean: 47.8313999
X SD : 16.7649829
Y SD : 26.9342120
Corr. : -0.0642526

https://blog.revolutionanalytics.com/2017/05/the-datasaurus-dozen.html



		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	**Negative Predictive Value** $\frac{TN}{(TN + FN)}$	**Accuracy** $\frac{TP + TN}{(TP + TN + FP + FN)}$

https://encord.com/glossary/confusion-matrix/

Sınıflandırma Metrikleri

Hata Matrisi (Confusion Matrix)

p = [1, 1, 1, 2, 2, 1, 2, 2, 2, 1]

t = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

$Acc = \frac{6}{10}$ %60

$$\begin{array}{c|c|c|c} & & \multicolumn{2}{c}{P} \\ & & 1 & 2 \\ \hline G & 1 & /// & // \\ & 2 & // & //// \end{array}$$

3	2
2	3

	1	2
1	TP	FN
2	FP	TN

65	2	1	0	8
0	46	6	30	0
1	0	20	0	2
1	1	0	60	0
30	5	2	1	80

- Hata Matrisi

Accuracy(Doğruluk): Sistemde doğru olarak yapılan tahminlerin tüm tahminlere oranıdır.

Accuracy = (TP+TN) / (TP+TN+FN+FP)

Precision(Kesinlik) : Pozitif olarak tahmin edilen bir durumdaki başarıyı gösteren durum.

Precision = TP / (TP + FP)

Recall(Hassasiyet) : Pozitif durumların ne kadar başarılı tahmin edildiğini gösterir.

Recall = TP / (TP + FN)

Sınıflandırma Metrikleri

Doğruluk (Accuracy) Kesinlik (Precision) Hassasiyet (Recal)

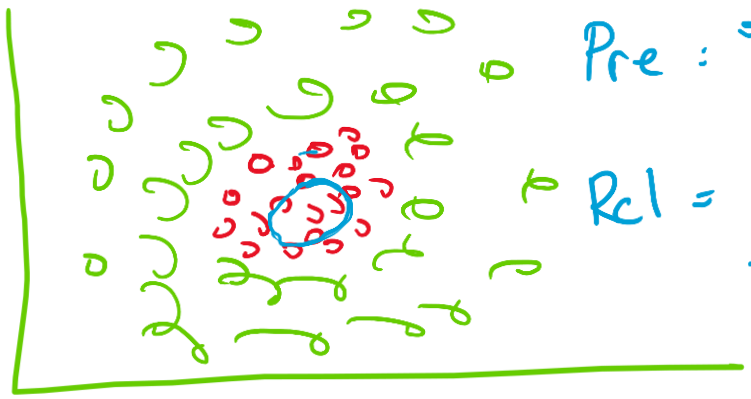
TP	FN
FP	TN

$$Acc = \frac{TP + TN}{TP + FN + FP + TN}$$

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

$$Rcl = \frac{TP}{TP + FN}$$

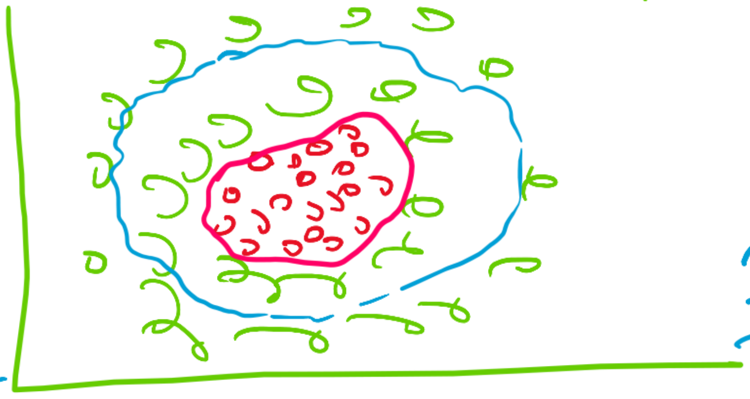
4 / 12	
0	



Pre : %100

$Rcl = \frac{4}{16} = \frac{1}{4}$

%25



20	0
10	

$Pre = \frac{20}{20}$

Rcl = %100

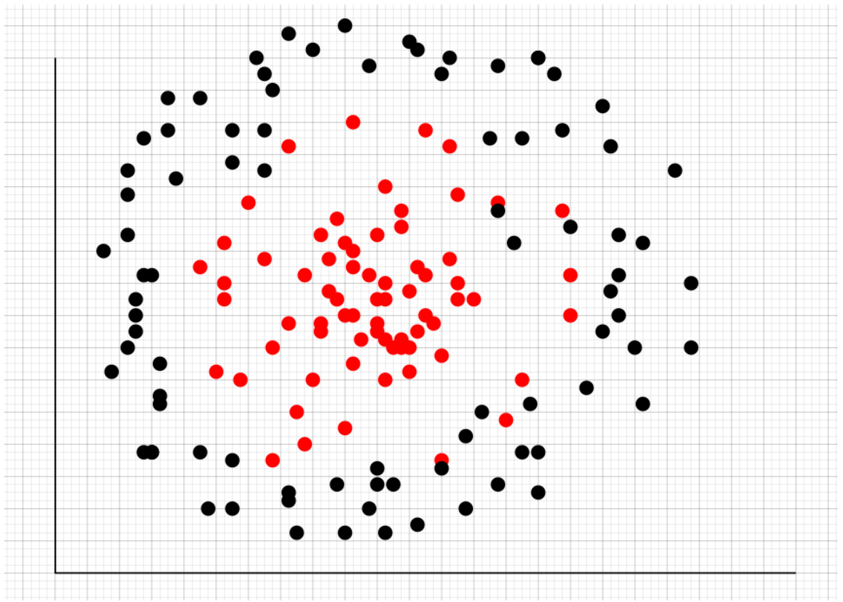
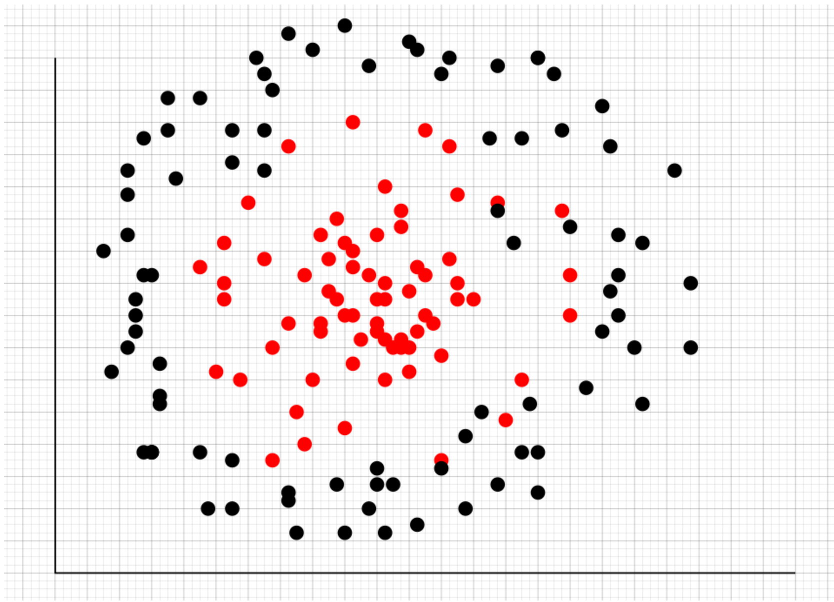
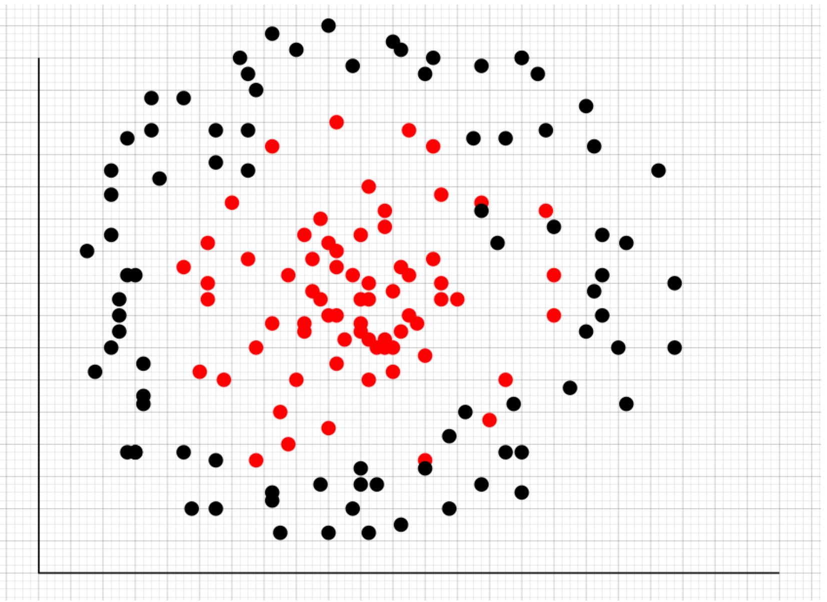
Sınıflandırma Metrikleri

F Skor

$$F = 2 \times \frac{Pre \otimes Rcl}{Pre + Rcl}$$

$$\frac{Pre + Rcl}{2}$$

$$\frac{\frac{1}{Pre} + \frac{1}{Rcl}}{2}$$

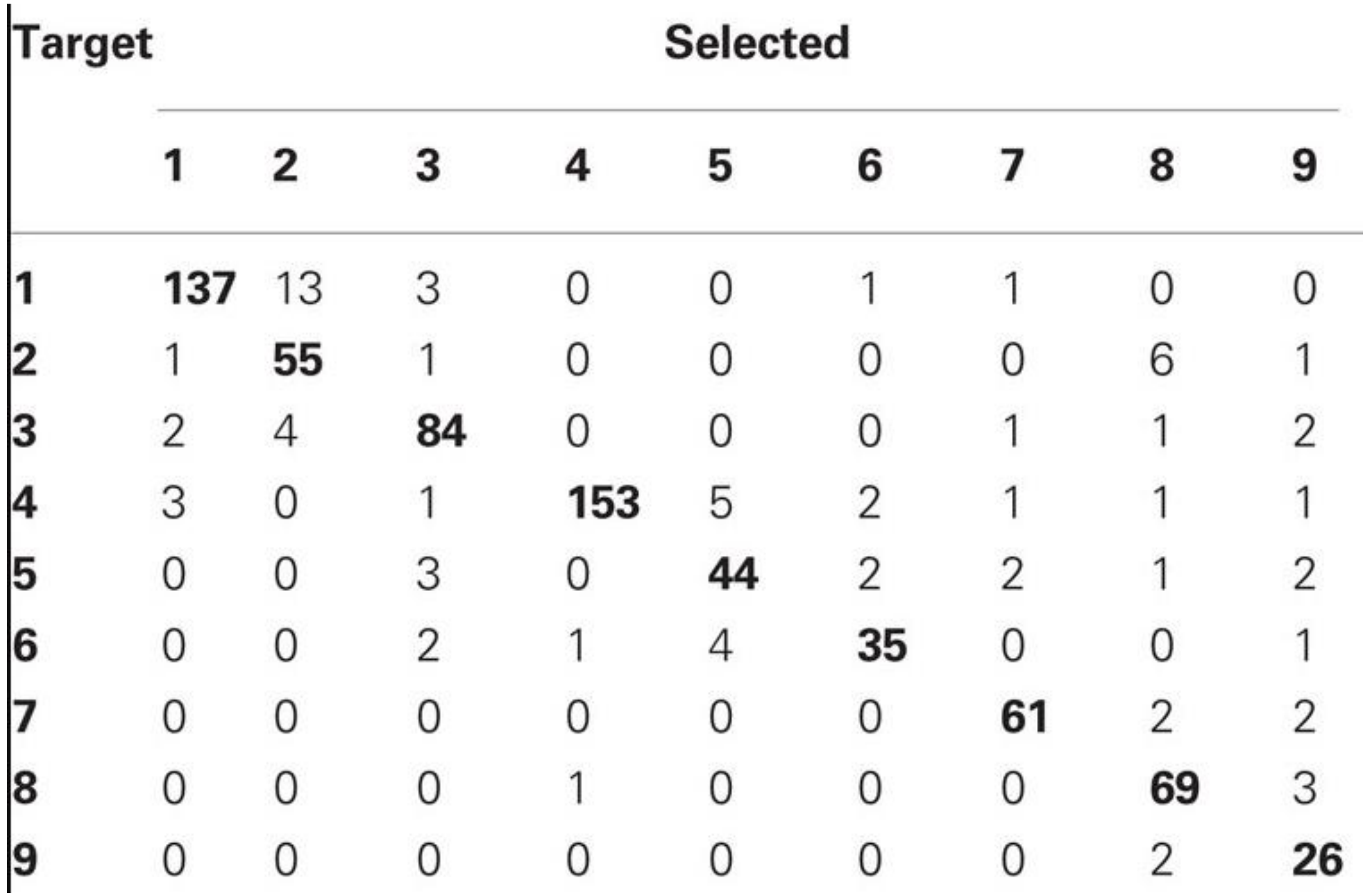




Dengesiz Dağılımlı Veri Seti (Imbalanced Dataset)

	Pos	Neg
Pos	0	10
Neg	0	90

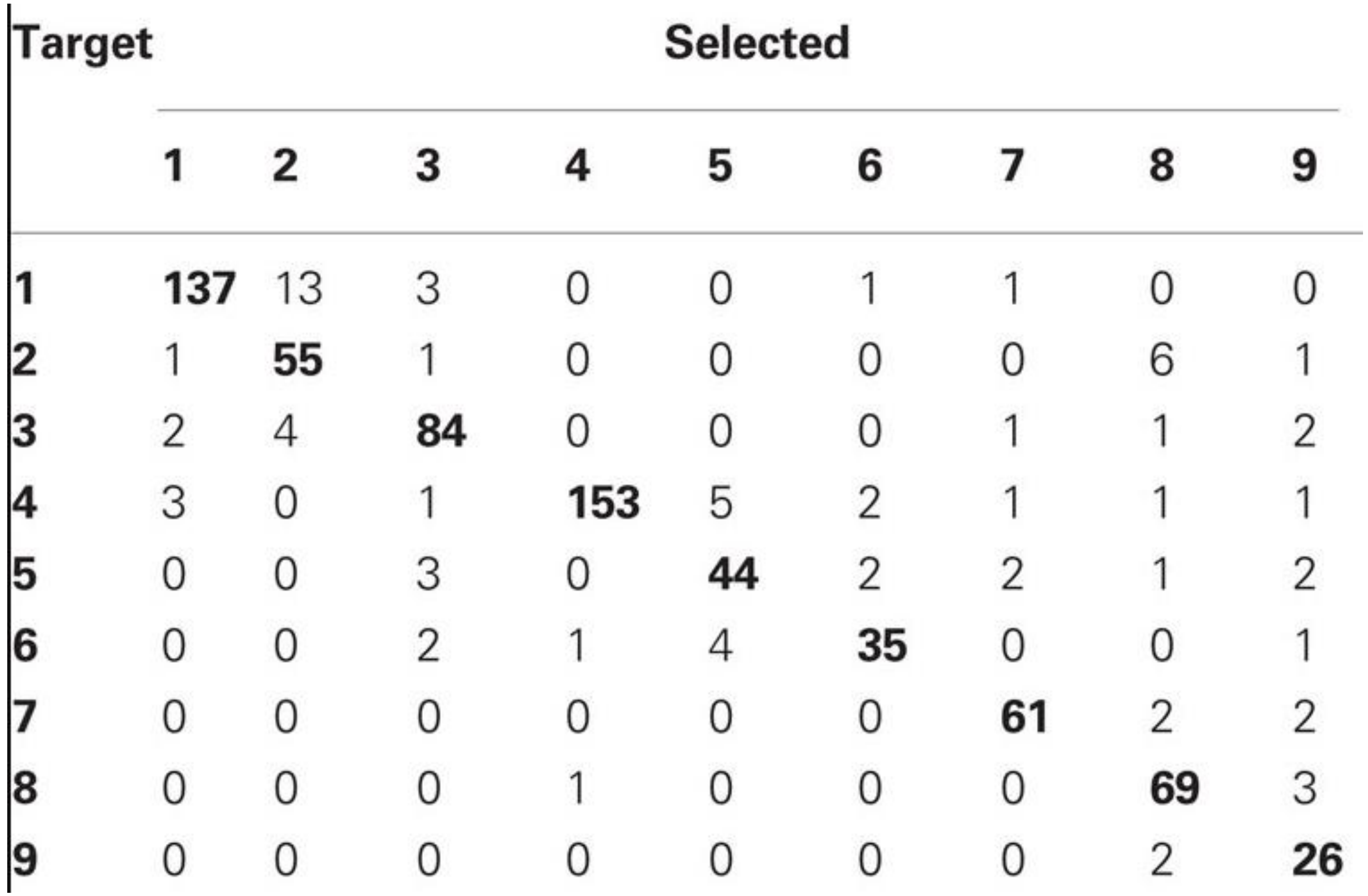
	Pos	Neg
Pos	10	0
Neg	80	10

Çok Sınıflı Veri Kümeleri

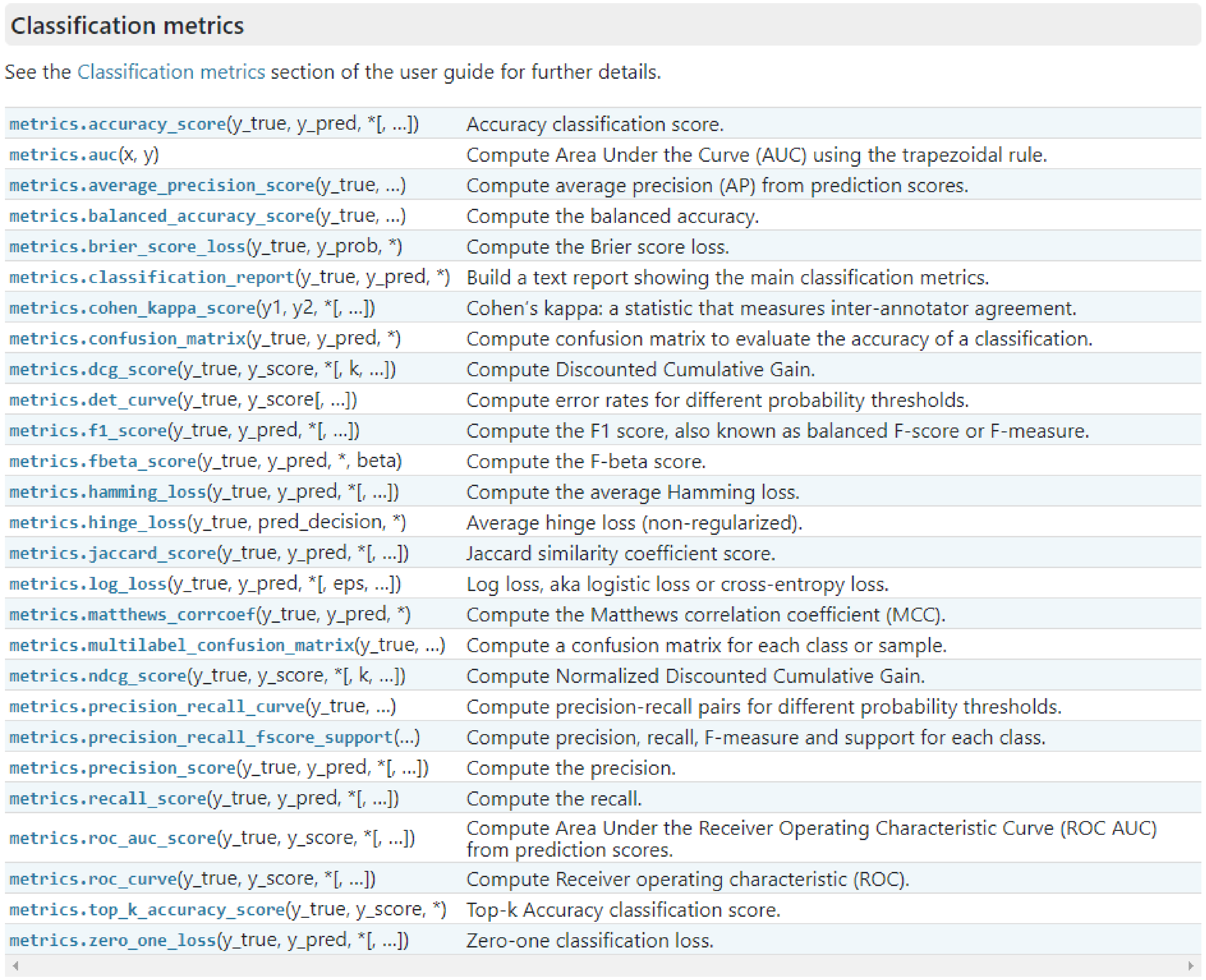


Target	**Selected**								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	**137**	13	3	0	0	1	1	0	0
2	1	**55**	1	0	0	0	0	6	1
3	2	4	**84**	0	0	0	1	1	2
4	3	0	1	**153**	5	2	1	1	1
5	0	0	3	0	**44**	2	2	1	2
6	0	0	2	1	4	**35**	0	0	1
7	0	0	0	0	0	0	**61**	2	2
8	0	0	0	1	0	0	0	**69**	3
9	0	0	0	0	0	0	0	2	**26**

Çok Sınıflı Veri Kümeleri



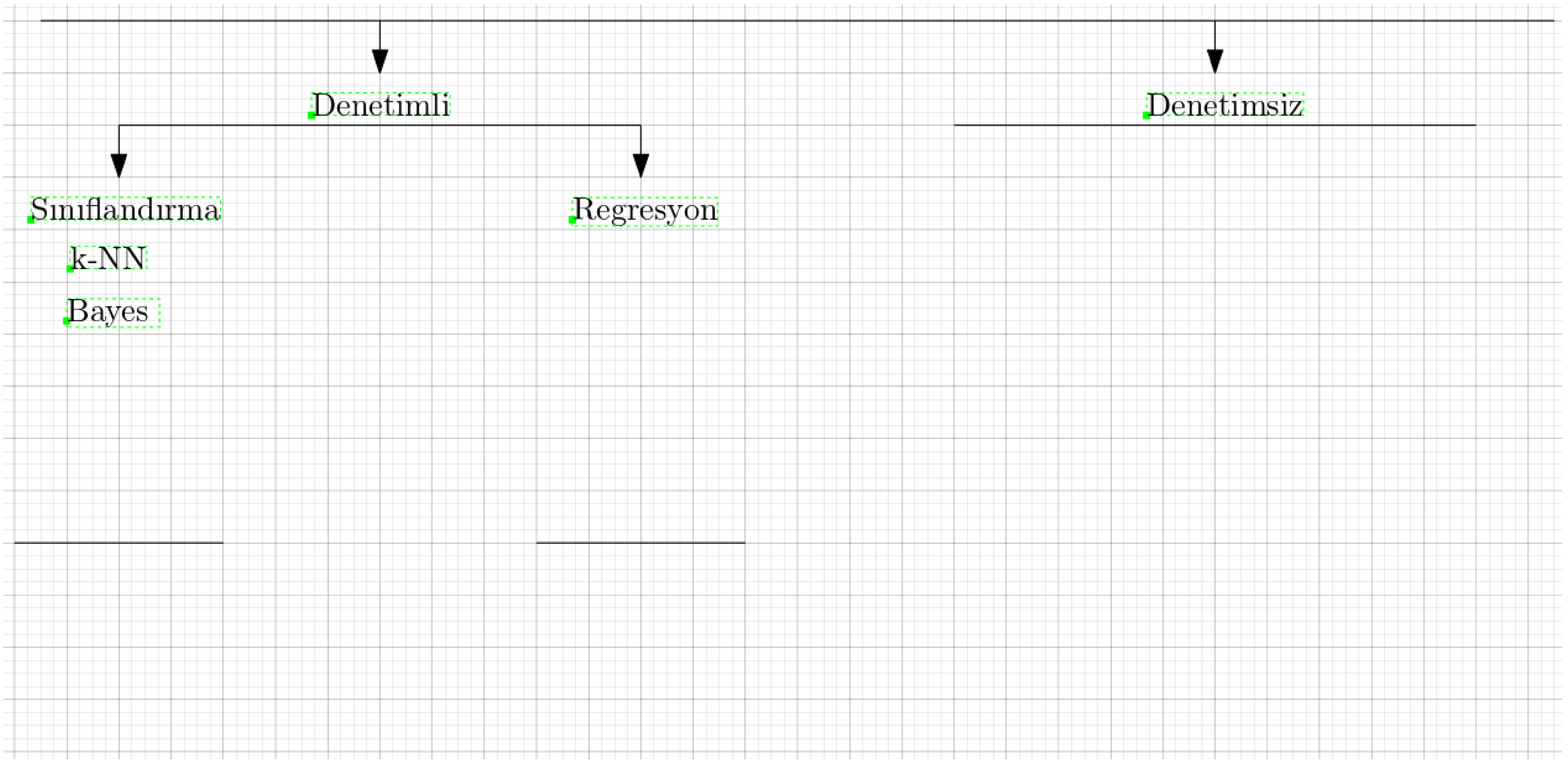
Target	Selected								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	**137**	13	3	0	0	1	1	0	0
2	1	**55**	1	0	0	0	0	6	1
3	2	4	**84**	0	0	0	1	1	2
4	3	0	1	**153**	5	2	1	1	1
5	0	0	3	0	**44**	2	2	1	2
6	0	0	2	1	4	**35**	0	0	1
7	0	0	0	0	0	0	**61**	2	2
8	0	0	0	1	0	0	0	**69**	3
9	0	0	0	0	0	0	0	2	**26**



Classification metrics

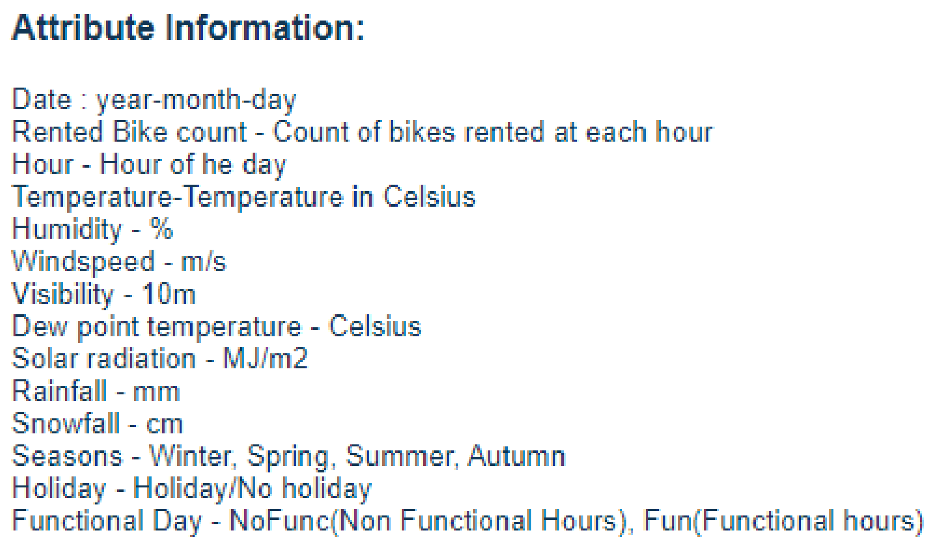
See the Classification metrics section of the user guide for further details.

`metrics.accuracy_score`(y_true, y_pred, *[, ...])	Accuracy classification score.
`metrics.auc`(x, y)	Compute Area Under the Curve (AUC) using the trapezoidal rule.
`metrics.average_precision_score`(y_true, ...)	Compute average precision (AP) from prediction scores.
`metrics.balanced_accuracy_score`(y_true, ...)	Compute the balanced accuracy.
`metrics.brier_score_loss`(y_true, y_prob, *)	Compute the Brier score loss.
`metrics.classification_report`(y_true, y_pred, *)	Build a text report showing the main classification metrics.
`metrics.cohen_kappa_score`(y1, y2, *[, ...])	Cohen's kappa: a statistic that measures inter-annotator agreement.
`metrics.confusion_matrix`(y_true, y_pred, *)	Compute confusion matrix to evaluate the accuracy of a classification.
`metrics.dcg_score`(y_true, y_score, *[, k, ...])	Compute Discounted Cumulative Gain.
`metrics.det_curve`(y_true, y_score[, ...])	Compute error rates for different probability thresholds.
`metrics.f1_score`(y_true, y_pred, *[, ...])	Compute the F1 score, also known as balanced F-score or F-measure.
`metrics.fbeta_score`(y_true, y_pred, *, beta)	Compute the F-beta score.
`metrics.hamming_loss`(y_true, y_pred, *[, ...])	Compute the average Hamming loss.
`metrics.hinge_loss`(y_true, pred_decision, *)	Average hinge loss (non-regularized).
`metrics.jaccard_score`(y_true, y_pred, *[, ...])	Jaccard similarity coefficient score.
`metrics.log_loss`(y_true, y_pred, *[, eps, ...])	Log loss, aka logistic loss or cross-entropy loss.
`metrics.matthews_corrcoef`(y_true, y_pred, *)	Compute the Matthews correlation coefficient (MCC).
`metrics.multilabel_confusion_matrix`(y_true, ...)	Compute a confusion matrix for each class or sample.
`metrics.ndcg_score`(y_true, y_score, *[, k, ...])	Compute Normalized Discounted Cumulative Gain.
`metrics.precision_recall_curve`(y_true, ...)	Compute precision-recall pairs for different probability thresholds.
`metrics.precision_recall_fscore_support`(...)	Compute precision, recall, F-measure and support for each class.
`metrics.precision_score`(y_true, y_pred, *[, ...])	Compute the precision.
`metrics.recall_score`(y_true, y_pred, *[, ...])	Compute the recall.
`metrics.roc_auc_score`(y_true, y_score, *[, ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
`metrics.roc_curve`(y_true, y_score, *[, ...])	Compute Receiver operating characteristic (ROC).
`metrics.top_k_accuracy_score`(y_true, y_score, *)	Top-k Accuracy classification score.
`metrics.zero_one_loss`(y_true, y_pred, *[, ...])	Zero-one classification loss.


Denetimli
Denetimsiz
Sınıflandırma
Regresyon
k-NN
Bayes

Regresyon: Nerelerde?

- Kategorik sınıf olmayan sürekli çıkış tahmini
- Örnek:
  - Seoul Bike Sharing Demand Data Set

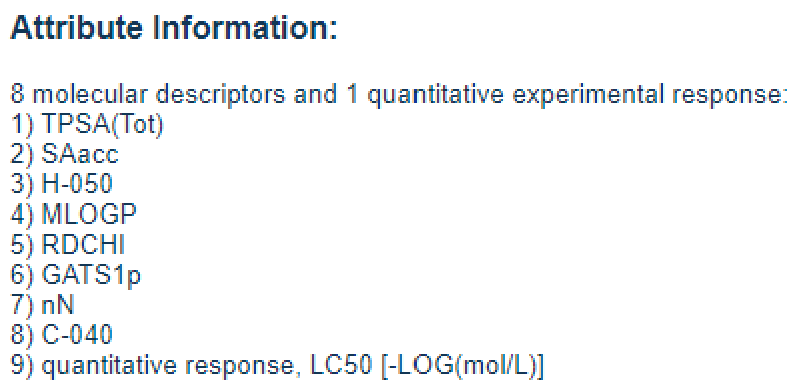


Attribute Information:

Date : year-month-day
Rented Bike count - Count of bikes rented at each hour
Hour - Hour of he day
Temperature-Temperature in Celsius
Humidity - %
Windspeed - m/s
Visibility - 10m
Dew point temperature - Celsius
Solar radiation - MJ/m2
Rainfall - mm
Snowfall - cm
Seasons - Winter, Spring, Summer, Autumn
Holiday - Holiday/No holiday
Functional Day - NoFunc(Non Functional Hours), Fun(Functional hours)

Regresyon: Nerelerde?

- Kategorik sınıf olmayan sürekli çıkış tahmini
- Örnek:
  - QSAR aquatic toxicity Data Set



Attribute Information:

8 molecular descriptors and 1 quantitative experimental response:
1) TPSA(Tot)
2) SAacc
3) H-050
4) MLOGP
5) RDCHI
6) GATS1p
7) nN
8) C-040
9) quantitative response, LC50 [-LOG(mol/L)]

Regresyon: Nerelerde?

- Kategorik sınıf olmayan sürekli çıkış tahmini
- Örnek:
    - Indoor Localization

Regresyon: Nerelerde?

- Kategorik sınıf olmayan sürekli çıkış tahmini
- Örnek:
  - Object Detection

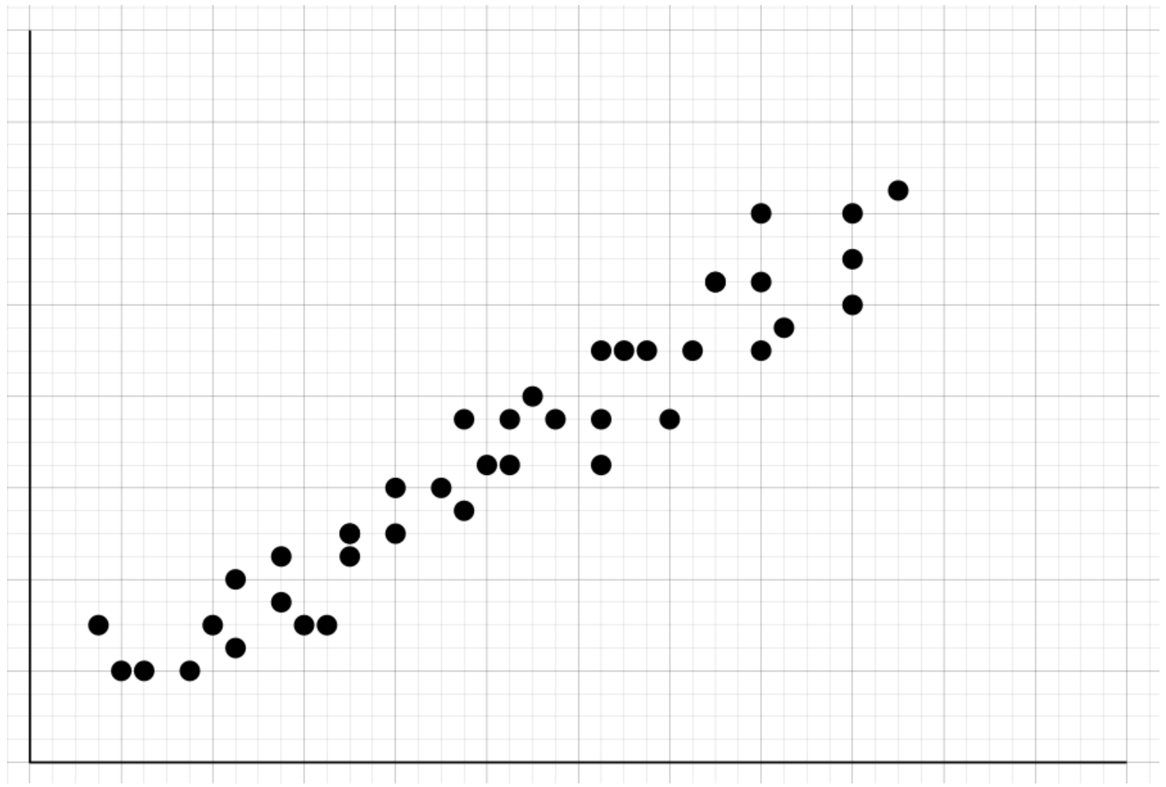
Regresyon: Nerelerde?

- Kategorik sınıf olmayan sürekli çıkış tahmini
- Örnek:
    - Antenna Parameter Estimation

Regresyon: Nerelerde?

- Kategorik sınıf olmayan sürekli çıkış tahmini
- Örnek:
  - Fizik Yasaları

Doğrusal Regresyon: Matematiksel Tanım



Doğr

Partial deriative w.r.t. $\hat{b_0}$: $\dfrac{\partial Z}{\partial \hat{b_0}} = \sum_i^n -2\left(Y_i - \hat{b_0} - \hat{b_1}X_i\right)$

Setting the derivative to 0 and solving, we have: $\hat{b_0} = \dfrac{1}{n}\sum_i^n Y_i - \dfrac{1}{n}\sum_i^n \hat{b_1}X_i$

$$\implies \hat{b_0} = \bar{Y} - \hat{b_1}\bar{X}$$

Partial deriative w.r.t. $\hat{b_1}$: $\dfrac{\partial Z}{\partial \hat{b_1}} = \sum_i^n -2X_i\left(Y_i - \hat{b_0} - \hat{b_1}X_i\right)$

Setting the derivative to 0 and substituting $\hat{b_1}$, we have:

$$\sum_i^n X_iY_i - \sum_i^n(\bar{Y} - \hat{b_1}\bar{X})X_i - \sum_i^n \hat{b_1}X_i^2 = 0$$

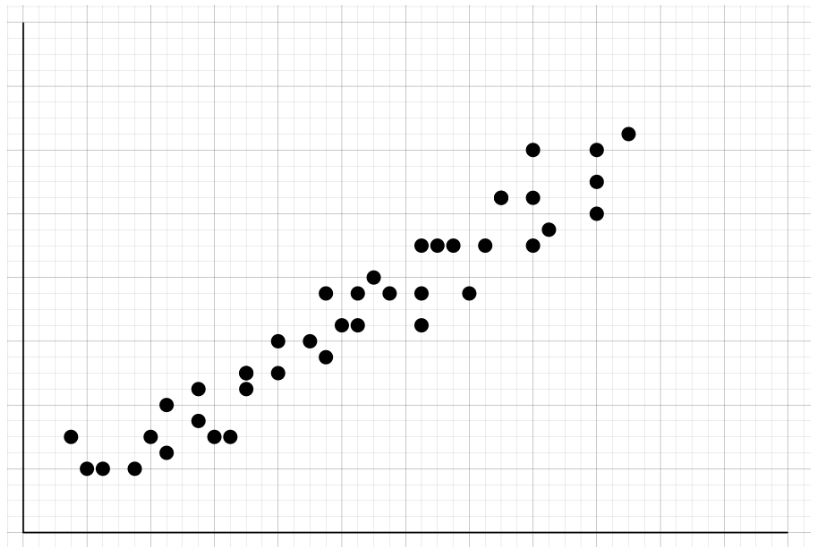
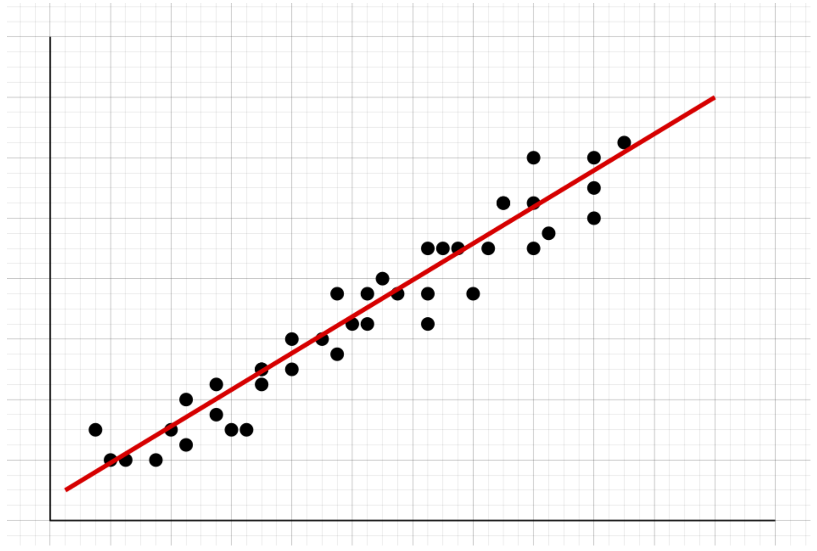
$$\sum_i^n X_iY_i - \bar{Y}\sum_i^n X_i + \hat{b_1}\left(\bar{X}\sum_i^n X_i - \sum_i^n X_i^2\right) = 0$$

$$\hat{b_1} = \frac{\sum_i^n X_iY_i - \bar{Y}\sum_i^n X_i}{\sum_i^n X_i^2 - \bar{X}\sum_i^n X_i}$$

$$= \frac{\sum_i^n X_iY_i - n\bar{X}\bar{Y}}{\sum_i^n X_i^2 - n\bar{X}^2}$$

simplifying: $\hat{b_1} = \dfrac{\sum_i^n(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_i^n(X_i - \bar{X})^2}$

Doğrusal Regresyon: Matematiksel Tanım (**Vektörel Form**)

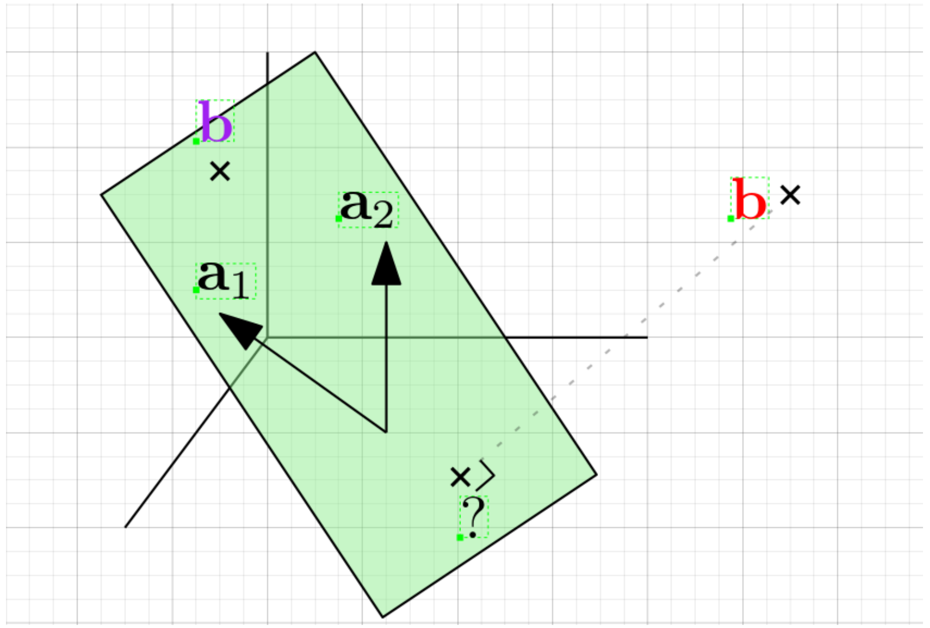
Doğrusal Regresyon: Matematiksel Tanım

Least Mean Square (LMS) Error

Doğrusal Regresyon: Matematiksel Tanım

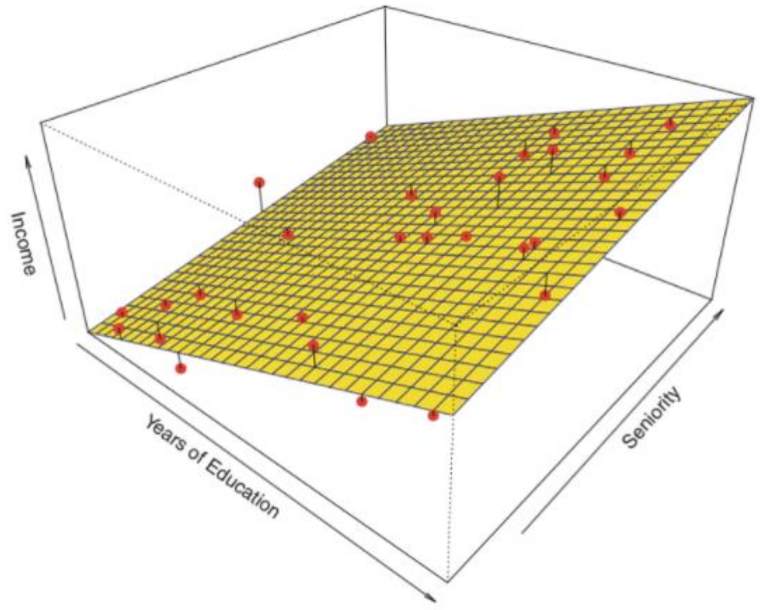
Least Mean Square (LMS) Error

$$\begin{bmatrix} x_{11} & 1 \\ x_{21} & 1 \\ \cdot & \cdot \\ \cdot & \cdot \\ x_{N1} & 1 \end{bmatrix} \boldsymbol{w} = \boldsymbol{y}$$

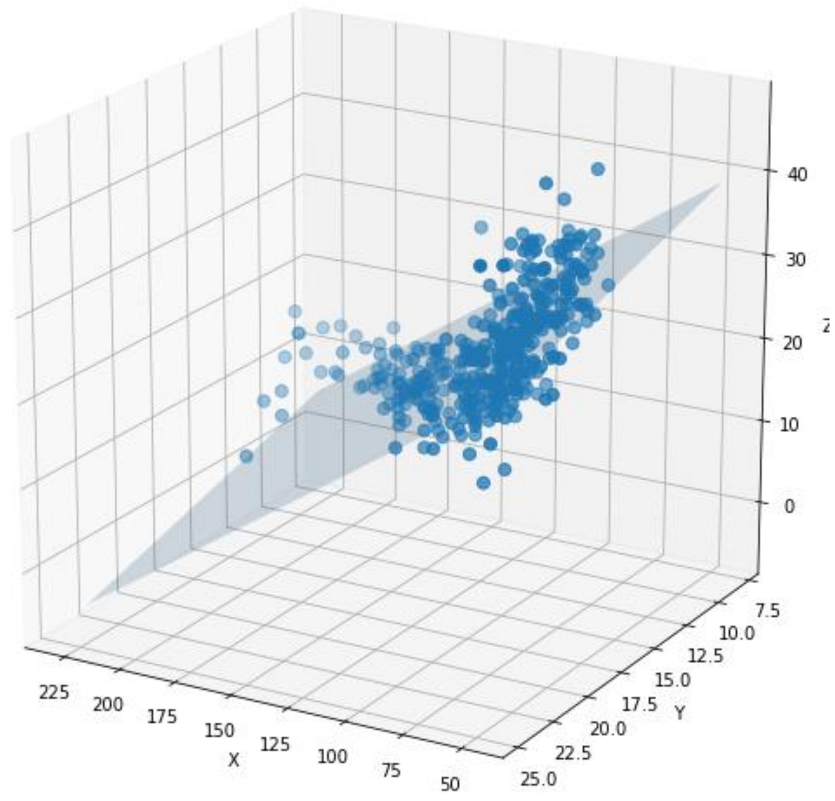


Doğrusal Regresyon: Çözüm Algoritması

Doğrusal Regresyon: Çözüm Algoritması

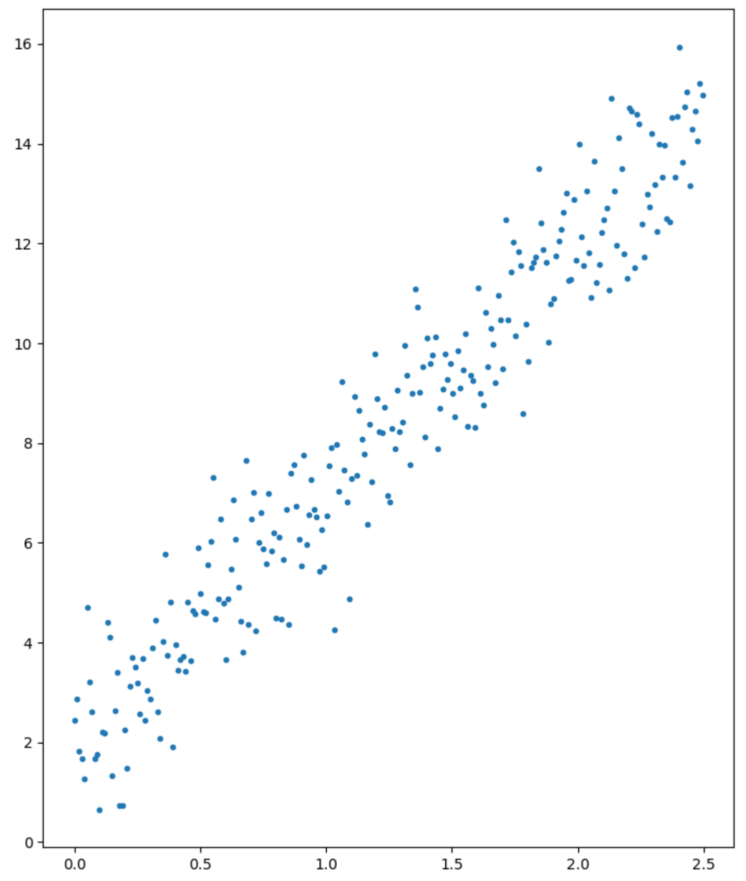


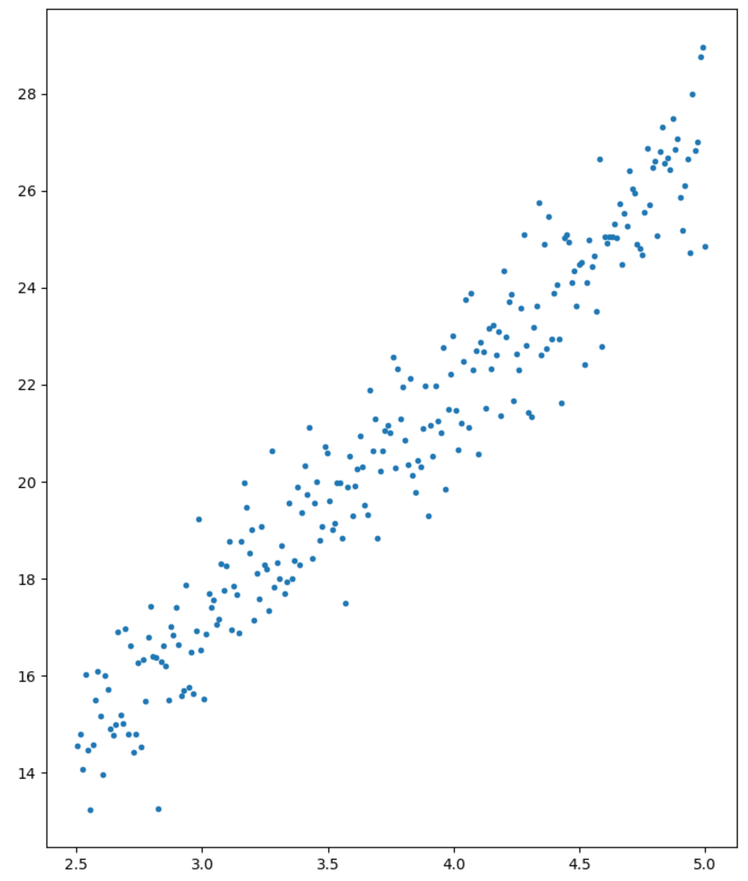
Source: James et al. *Introduction to Statistical Learning* (Springer 2013)



https://i.stack.imgur.com/O5036.png

Doğrusal Regresyon: Eğitim, Test ve Skor

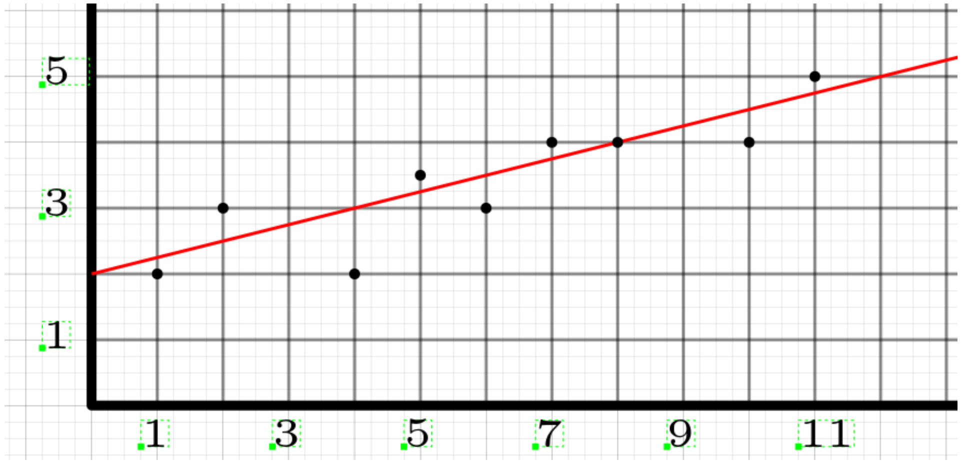




Doğrusal Regresyon: Eğitim, Test ve Skor

Skor Metrikleri:

Doğrusal Regresyon: Taylor, Fourier Serileri vd



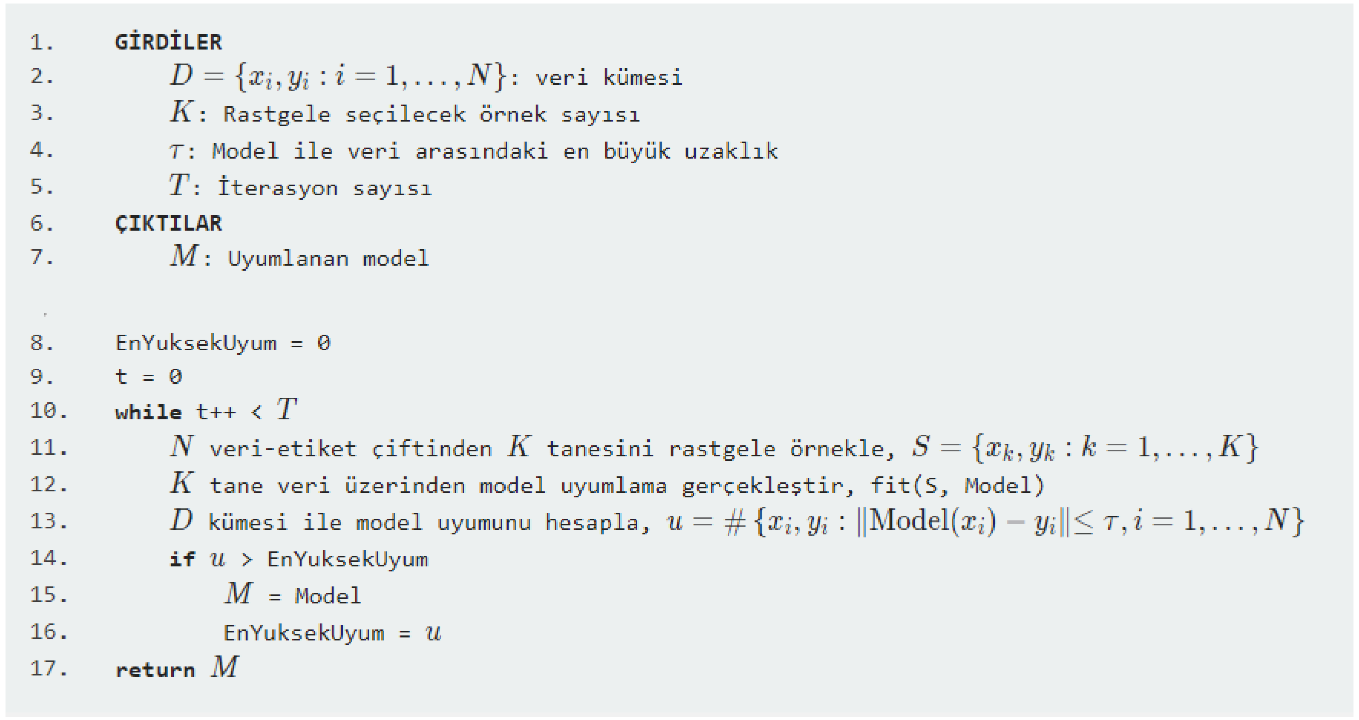
x	y
1	2
2	3
4	2
5	3.5
6	3
7	4
8	6
10	4
11	5

Doğrusal Regresyon: Aykırı Örnekler ve RANSAC Algoritması

RANSAC: Random Sample Consensus

Doğrusal Regresyon: Aykırı Örnekler ve RANSAC Algoritması

RANSAC: Random Sample Consensus



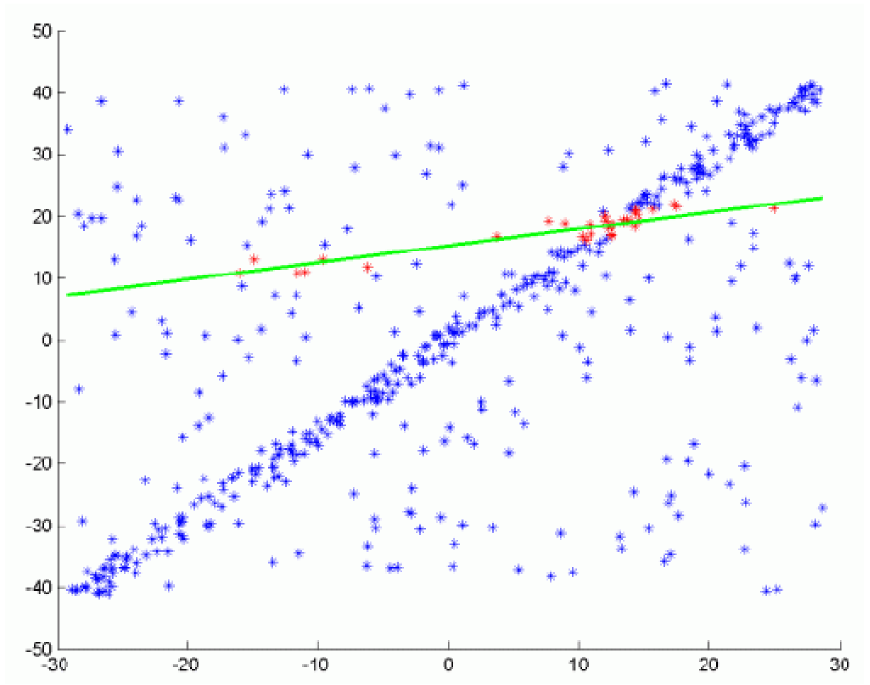
```
GİRDİLER
    $D = \{x_i, y_i : i = 1, \dots, N\}$: veri kümesi
    $K$: Rastgele seçilecek örnek sayısı
    $\tau$: Model ile veri arasındaki en büyük uzaklık
    $T$: İterasyon sayısı
ÇIKTILAR
    $M$: Uyumlanan model

EnYuksekUyum = 0
t = 0
while t++ < $T$
    $N$ veri-etiket çiftinden $K$ tanesini rastgele örnekle, $S = \{x_k, y_k : k = 1, \dots, K\}$
    $K$ tane veri üzerinden model uyumlama gerçekleştir, fit(S, Model)
    $D$ kümesi ile model uyumunu hesapla, $u = \#\{x_i, y_i : \|\text{Model}(x_i) - y_i\| \leq \tau, i = 1, \dots, N\}$
    if $u$ > EnYuksekUyum
        $M$ = Model
        EnYuksekUyum = $u$
return $M$
```

https://imlab.io/2020/08/23/ransac-algorithm/

Doğrusal Regresyon: Aykırı Örnekler ve RANSAC Algoritması

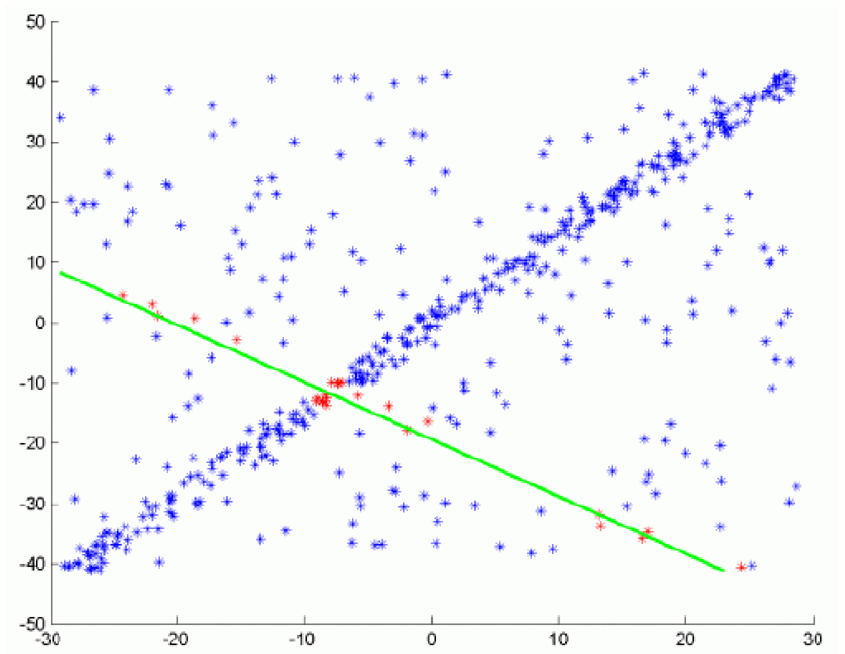
RANSAC: Random Sample Consensus



https://www.analyticsvidhya.com/blog/2021/02/new-approach-for-regression-analysis-ransac-and-mlesac/

Doğrusal Regresyon: Aykırı Örnekler ve RANSAC Algoritması

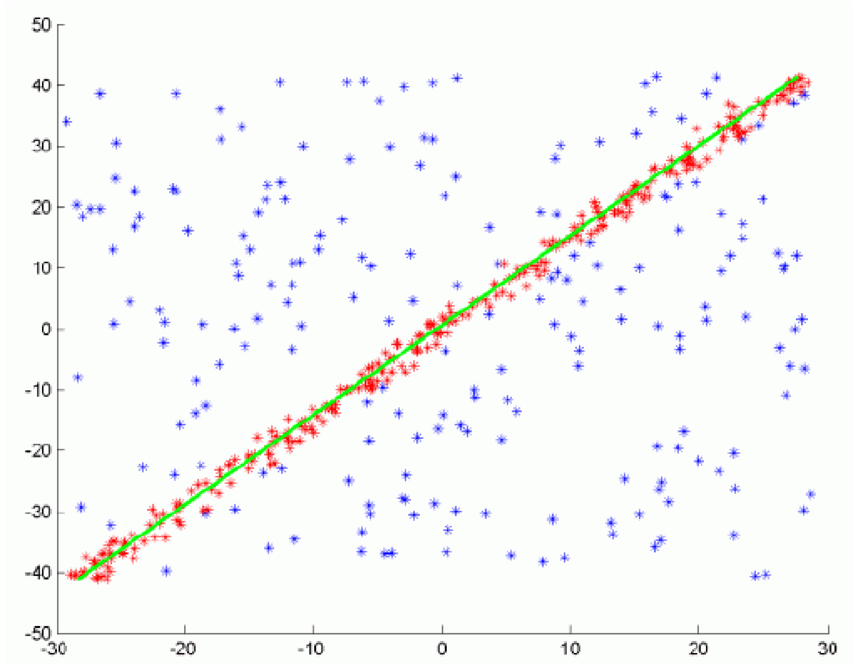
RANSAC: Random Sample Consensus



https://www.analyticsvidhya.com/blog/2021/02/new-approach-for-regression-analysis-ransac-and-mlesac/

Doğrusal Regresyon: Aykırı Örnekler ve RANSAC Algoritması

RANSAC: Random Sample Consensus



https://www.analyticsvidhya.com/blog/2021/02/new-approach-for-regression-analysis-ransac-and-mlesac/

sklearn.linear_model: Linear Models

The sklearn.linear_model module implements a variety of linear models.

User guide: See the Linear Models section for further details.

The following subsections are only rough guidelines: the same estimator can fall into multiple categories, depending on its parameters.

Linear classifiers

linear_model.LogisticRegression([penalty, ...])	Logistic Regression (aka logit, MaxEnt) classifier.
linear_model.LogisticRegressionCV(*[, Cs, ...])	Logistic Regression CV (aka logit, MaxEnt) classifier.
linear_model.PassiveAggressiveClassifier(*)	Passive Aggressive Classifier.
linear_model.Perceptron(*[, penalty, alpha, ...])	Linear perceptron classifier.
linear_model.RidgeClassifier([alpha, ...])	Classifier using Ridge regression.
linear_model.RidgeClassifierCV([alphas, ...])	Ridge classifier with built-in cross-validation.
linear_model.SGDClassifier([loss, penalty, ...])	Linear classifiers (SVM, logistic regression, etc.) with SGD training.
linear_model.SGDOneClassSVM([nu, ...])	Solves linear One-Class SVM using Stochastic Gradient Descent.

Classical linear regressors

linear_model.LinearRegression(*[, ...])	Ordinary least squares Linear Regression.
linear_model.Ridge([alpha, fit_intercept, ...])	Linear least squares with l2 regularization.
linear_model.RidgeCV([alphas, ...])	Ridge regression with built-in cross-validation.
linear_model.SGDRegressor([loss, penalty, ...])	Linear model fitted by minimizing a regularized empirical loss with SGD.

Regressors with variable selection

The following estimators have built-in variable selection fitting procedures, but any estimator using a L1 or elastic-net penalty also performs variable selection: typically SGDRegressor or SGDClassifier with an appropriate penalty.

linear_model.ElasticNet([alpha, l1_ratio, ...])	Linear regression with combined L1 and L2 priors as regularizer.
linear_model.ElasticNetCV(*[, l1_ratio, ...])	Elastic Net model with iterative fitting along a regularization path.
linear_model.Lars(*[, fit_intercept, ...])	Least Angle Regression model a.k.a.
linear_model.LarsCV(*[, fit_intercept, ...])	Cross-validated Least Angle Regression model.
linear_model.Lasso([alpha, fit_intercept, ...])	Linear Model trained with L1 prior as regularizer (aka the Lasso).

Outlier-robust regressors

Any estimator using the Huber loss would also be robust to outliers, e.g. `SGDRegressor` with `loss='huber'`.

`linear_model.HuberRegressor`(*[, epsilon, ...])	Linear regression model that is robust to outliers.
`linear_model.QuantileRegressor`(*[, ...])	Linear regression model that predicts conditional quantiles.
`linear_model.RANSACRegressor`([...])	RANSAC (RANdom SAmple Consensus) algorithm.
`linear_model.TheilSenRegressor`(*[, ...])	Theil-Sen Estimator: robust multivariate regression model.