Model Valuation and Performance Metrics

Model Valuation and Performance Metrics

Kfold cross-validation

1. **Data Preparation**: Splitting the dataset into training, validation, and test sets.
2. **Model Fitting**: Creating multiple models.

LinReg/NonLin/KNN/SVR/DL/Reg tree

3. **Model Evaluation**:

- for Reg Problem: Mean Absolute error (MAE), Calculation of Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) on the validation set for both models.

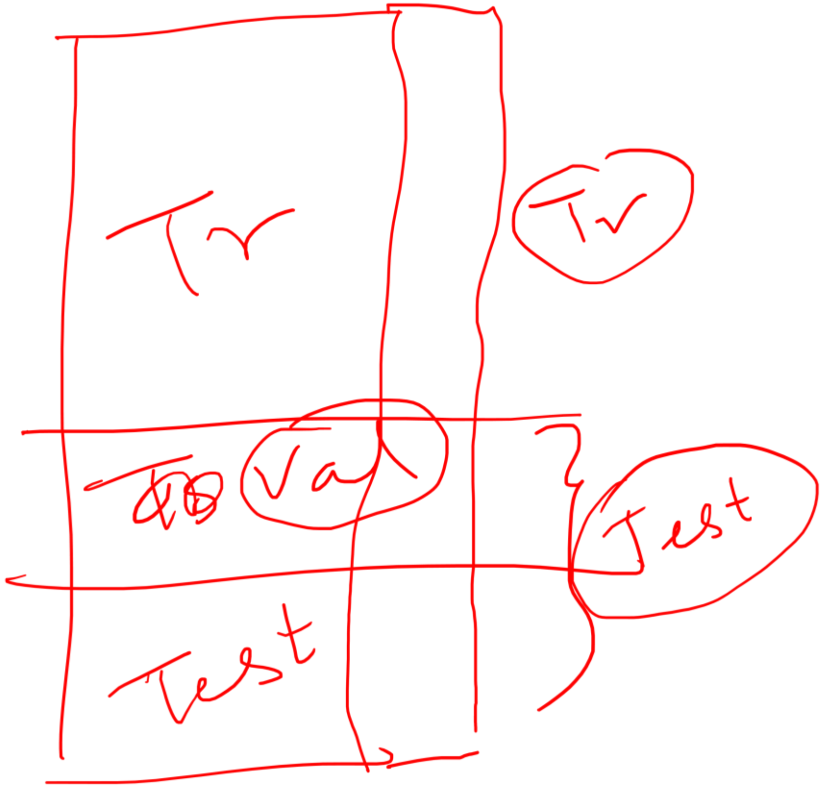
Log Reg/KNN/SVC/DT/DL

- for Classification Problem: Using confusion matrix, accuracy, precision, recall (sensitivity), specificity, F1-score etc on the validation set.

4. **Cross-Validation**: Performing cross-validation on the training set.

5. **Final Model Selection and Evaluation**: Applying the best model on the test set.

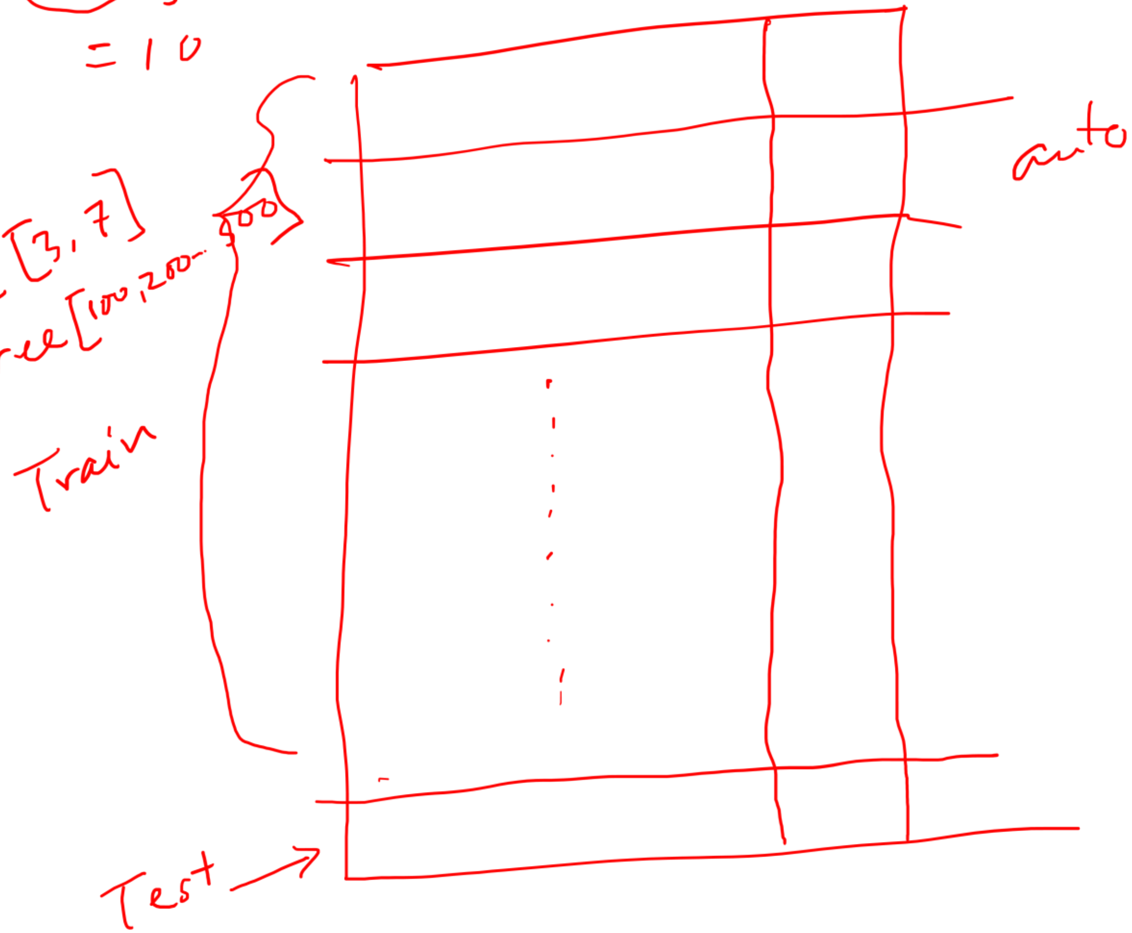
Validation →



K fold CV cross validation

= 10

DT
depth [3, 7]
tree [100, 200 ... 400]



KNN → K → how many neighbours

KMeans → K → how many clusters

K fold → K → how many folds

Model Valuation and Performance Metrics

- **Mean Squared Error (MSE)**
- MSE is a measure of the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. It's a common measure of the estimation accuracy of a predictive model in regression tasks.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- **Root Mean Squared Error (RMSE)**
- RMSE is the square root of the MSE. It's a widely used measure of the differences between values predicted by a model or an estimator and the values observed. The RMSE represents the sample standard deviation of the differences between predicted and observed values.

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}}$$

Model Valuation and Performance Metrics

- **Accuracy**
- Most commonly used metrics for evaluating classification models. It measures the proportion of total correct predictions (both true positives and true negatives) out of all predictions made.
- Accuracy=Number of Correct Predictions / Total Number of Predictions

Or, using the terms of the confusion matrix:

- Accuracy= (TP + TN) / (TP + FP + FN + TN)
- **Specificity**
- Specificity measures the proportion of actual negatives that are correctly identified as such (e.g., the percentage of healthy people who are correctly identified as not having the condition, in the medical context). It's a key metric when the cost of false positives is high. Specificity=True Negatives (TN)/ (True Negatives (TN) + False Positives (FP))
- **Recall (Sensitivity)**
- Recall, also known as sensitivity, is the ratio of true positive predictions to the total actual positives. It answers the question: "Of all the actual positive instances, how many did we correctly classify as positive?"

  Recall=True Positives (TP) / (True Positives (TP) + False Negatives (FN))

Model Valuation and Performance Metrics

- **Precision**
- Precision is the ratio of true positive predictions to the total positive predictions (including both true positives and false positives). It answers the question: "Of all instances classified as positive, how many are actually positive?"

  Precision=True Positives (TP)/(True Positives (TP) + False Positives (FP))

- **F1-Score**
- The F1-score is the harmonic mean of precision and recall. It provides a single score that balances both the precision and recall. It's particularly useful when you need to balance both precision and recall, such as in imbalanced datasets.

  F1-score=2×(Precision×Recall) / (Precision+Recall)

Model Valuation and Performance Metrics

		True class		
		Positive	Negative	Measures
Predicted class	Positive	True positive *TP*	False positive *FP*	Positive predictive value (PPV) $\frac{TP}{TP+FP}$
	Negative	False negative *FN*	True negative *TN*	Negative predictive value (NPV) $\frac{TN}{FN+TN}$
Measures		Sensitivity $\frac{TP}{TP+FN}$	Specificity $\frac{TN}{FP+TN}$	Accuracy $\frac{TP+TN}{TP+FP+FN+TN}$

		Actual Value	
		Positive	Negative
Result Obtained	Positive	**True Positive** **(1- β)**	False Positive Type-I Error (α)
	Negative	False Negative Type-II Error (β)	**True Negative**

		Predicted class	
		positive	negative
True class	Positive (P)	**T**rue **P**ositive (TP)	**F**alse **N**egative (FN)
	Negative (N)	**F**alse **P**ositive (FP)	**T**rue **N**egative (TN)

Row summary

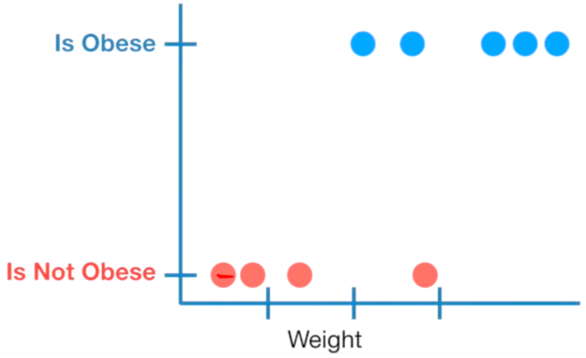
$TPR = \frac{TP}{P}$	$FNR = \frac{FN}{P}$
$TNR = \frac{TN}{N}$	$FPR = \frac{FP}{N}$

$PPV = \frac{TP}{TP + FP}$	$NPR = \frac{TN}{TN + FN}$
$FDR = \frac{FP}{TP + FP}$	$FOR = \frac{FN}{TN + FN}$

Column summary

Log Reg training and Thresholds

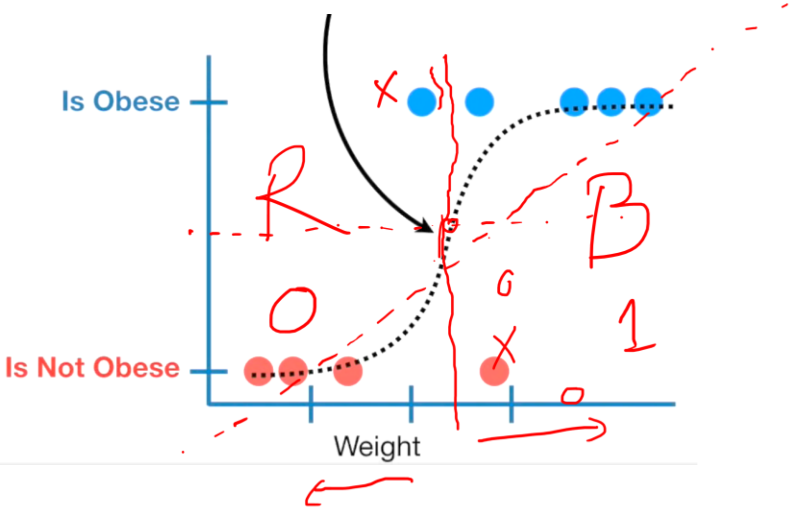
treshold prob = 0.5



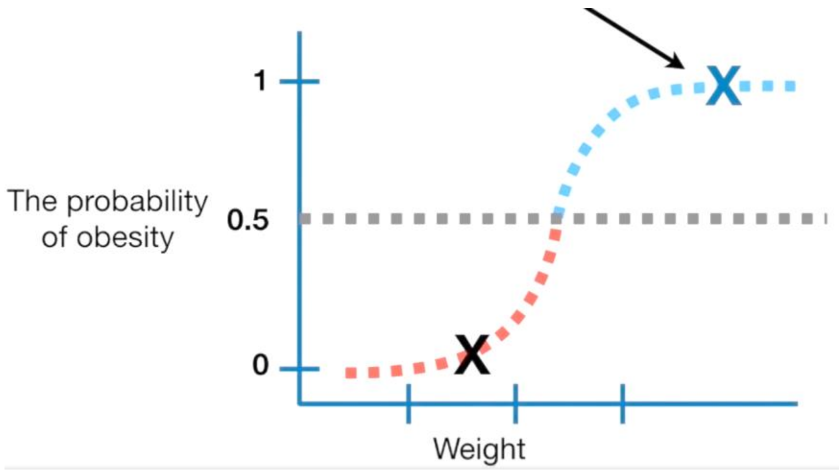
$$y = \frac{e^{\beta_0 + \beta_1 x_1}}{1 + e^{\beta_0 + \beta_1 x_1}}$$

Sigmoid f^n

Log Reg Curve



Predicted as obese

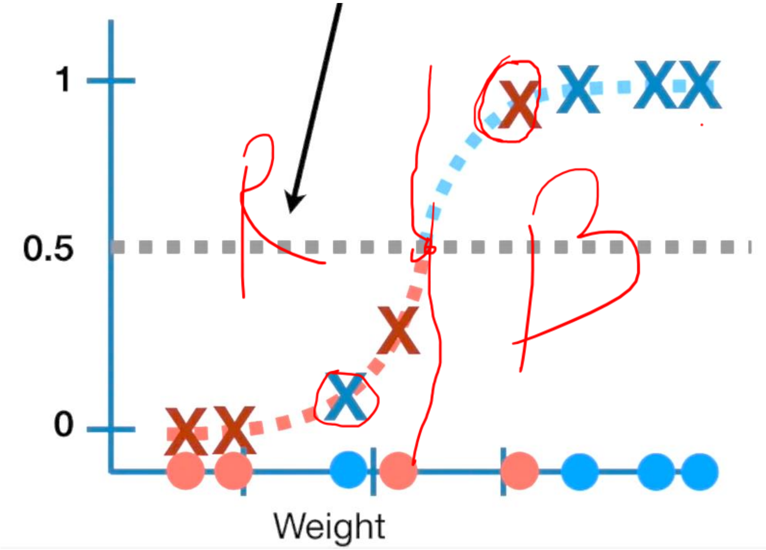


https://www.youtube.com/watch?v=4jRBRDbJemM&t=936s

Perf of Log Reg on test data with diff Thresholds

x | y

Predictions on test data with tp = 0.5



		Actual	
		Is Obese	Is Not Obese
Predicted	Is Obese	3	1
	Is Not Obese	1	3
		3/4	3/4

Sensitivity = 75%

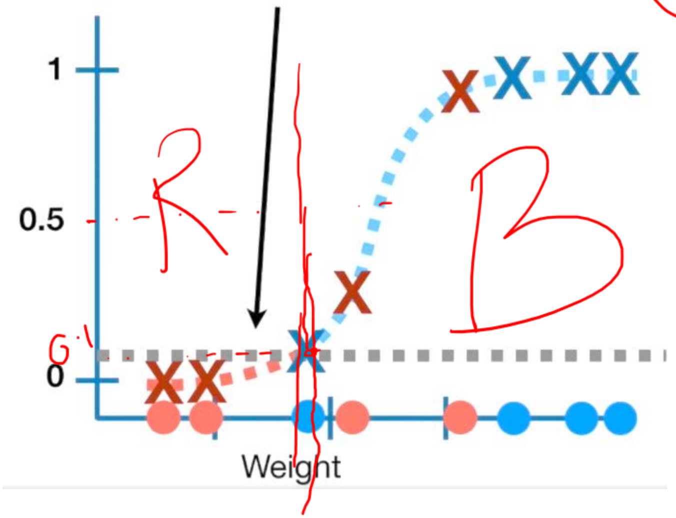
Specificity = 75%

Acc → 75%

https://www.youtube.com/watch?v=4jRBRDbJemM&t=936s

Perf of Log Reg on test data with diff Thresholds

Predictions on test data with tp = eg 0.1



		Actual	
		Is Obese	Is Not Obese
Predicted	Is Obese	4	2
	Is Not Obese	0	2
		$\frac{4}{4}$	$\frac{2}{4}$

Sensitivity = 100%

Specificity = 50%

Acc 75%

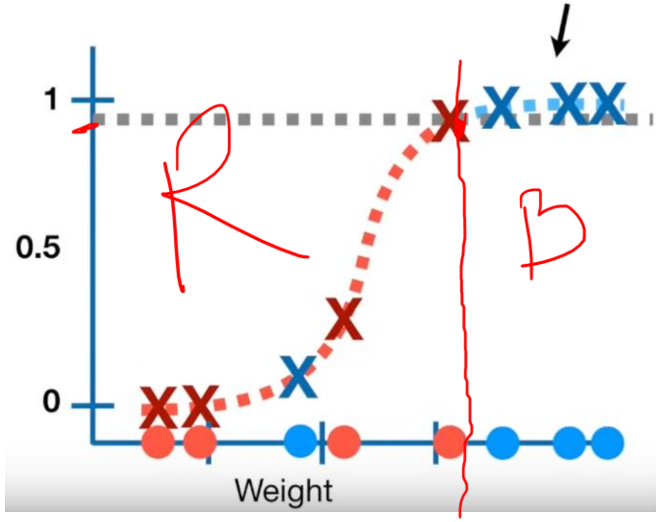
Think about an infectious disease. This is very important to correctly predict all the “yes” infected cases

https://www.youtube.com/watch?v=4jRBRDbJemM&t=936s

Perf of Log Reg on test data with diff Thresholds

Covid: sensitivity

Predictions on test data with tp = eg 0.9



This is better than 0.5 for sure

		Actual	
		Is Obese	Is Not Obese
Predicted	Is Obese	3	0
	Is Not Obese	1	4
		3/4	4/4

Acc

Sensitivity = 75%

Specificity = 100%

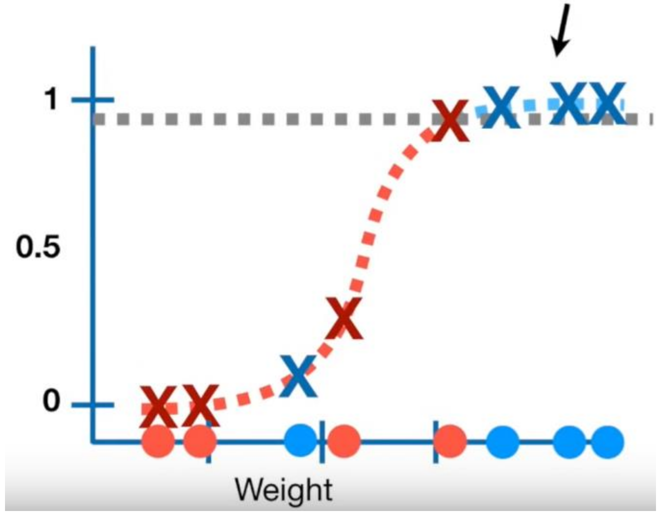
Acc 75%

But which threshold is the best?

https://www.youtube.com/watch?v=4jRBRDbJemM&t=936s

Perf of Log Reg on test data with diff Thresholds

Predictions on test data with tp = eg 0.9



		Actual	
		Is Obese	Is Not Obese
Predicted	Is Obese	3	0
	Is Not Obese	1	4

Sensitivity = 75%

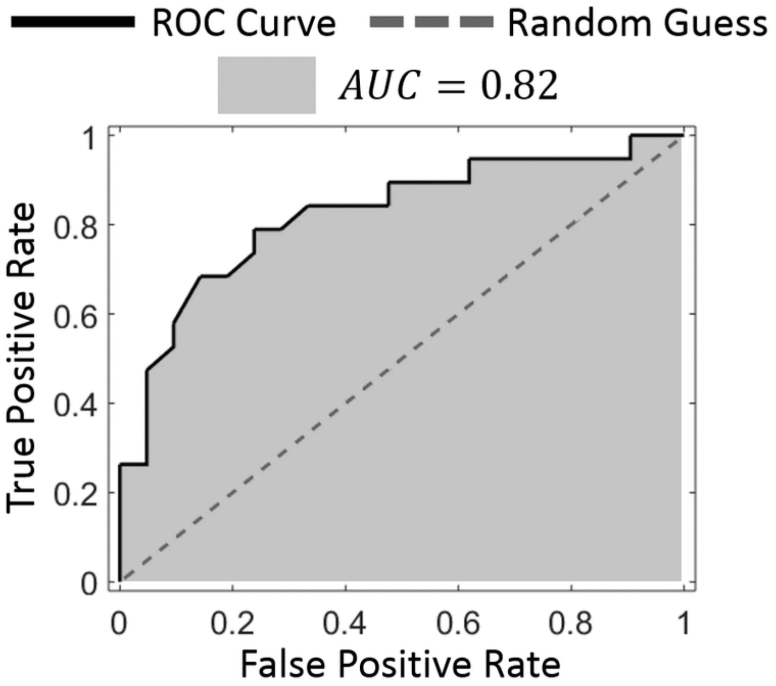
Specificity = 100%

This is better than 0.5 for sure

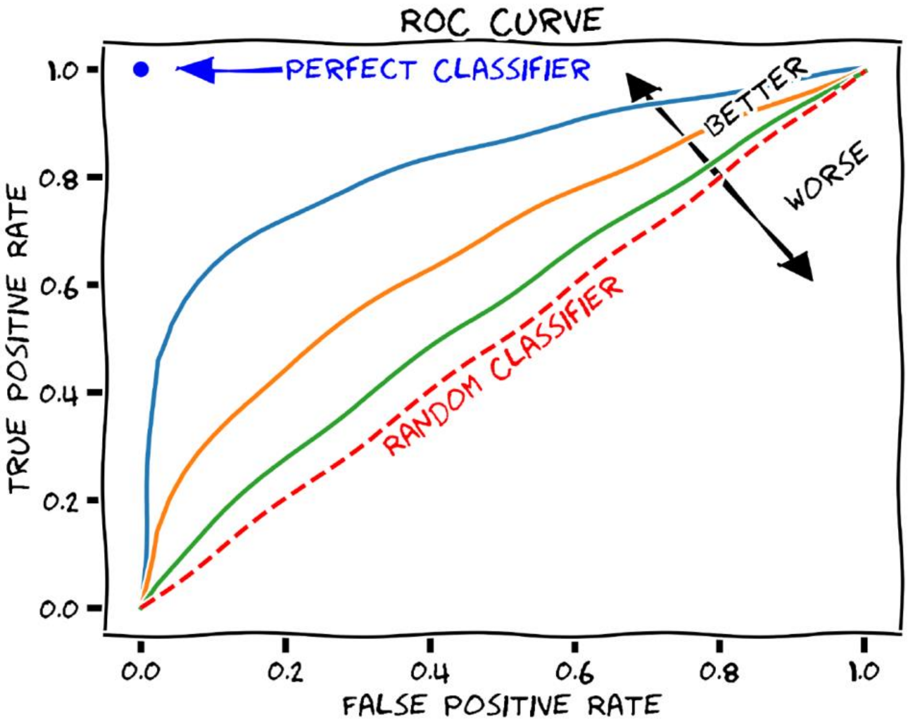
But which threshold is the best?

https://www.youtube.com/watch?v=4jRBRDbJemM&t=936s

ROC (Receiver Operator Curve) Curve and AUC (Area Under Curve)



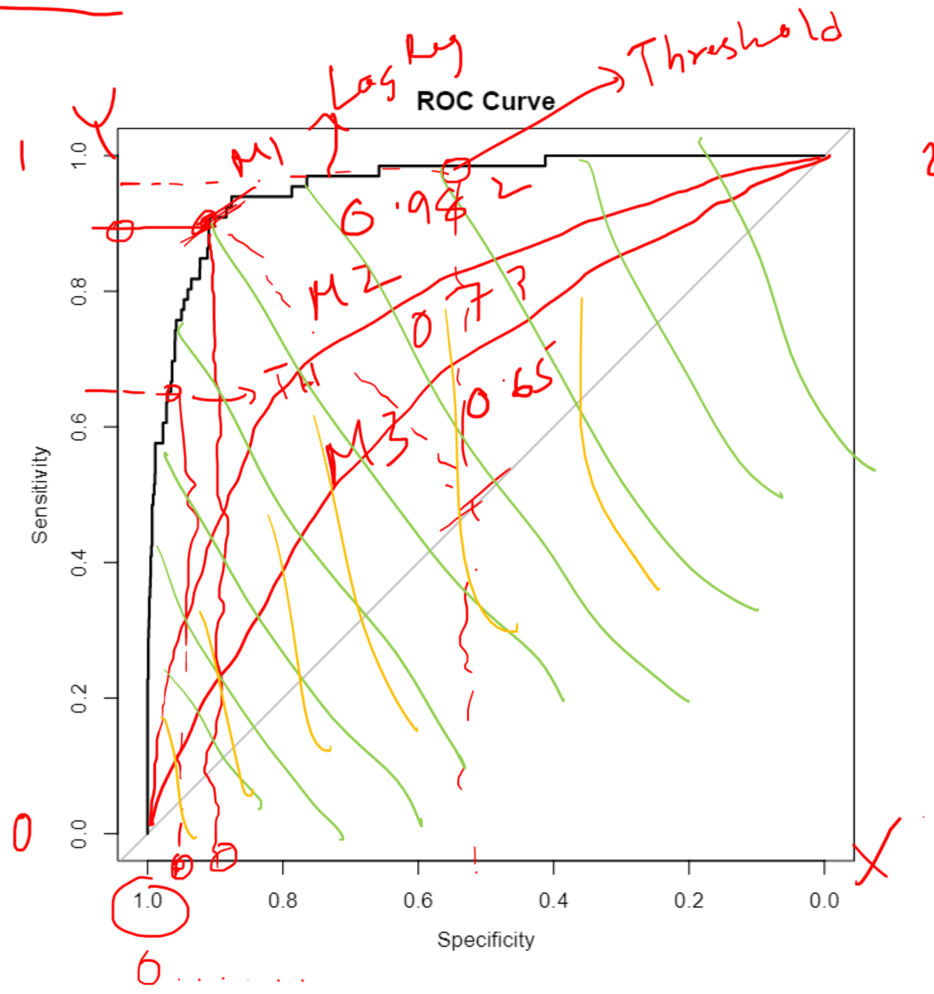
https://www.researchgate.net/profile/Jeffrey_Cochran2/publication/327954265/figure/fig5/AS:893652507312129@1590074769668/ROC-Area-Under-the-Curve-AUC-ROC-curves-plot-the-true-positive-rate-vs-the-false.ppm



https://mlwhiz.com/images/roc-auc-curves-explained/5_hu84485d9ac406291b53c5109d7ec0e2a3_312173_1500x0_resize_box_2.png

ROC (Receiver Operator Curve) Curve and AUC (Area Under Curve)

R/python



Threshold p

200

0.0
0.005
0.010
0.015
⋮
0.0350
0.355
⋮
0.875
0.880
⋮

200

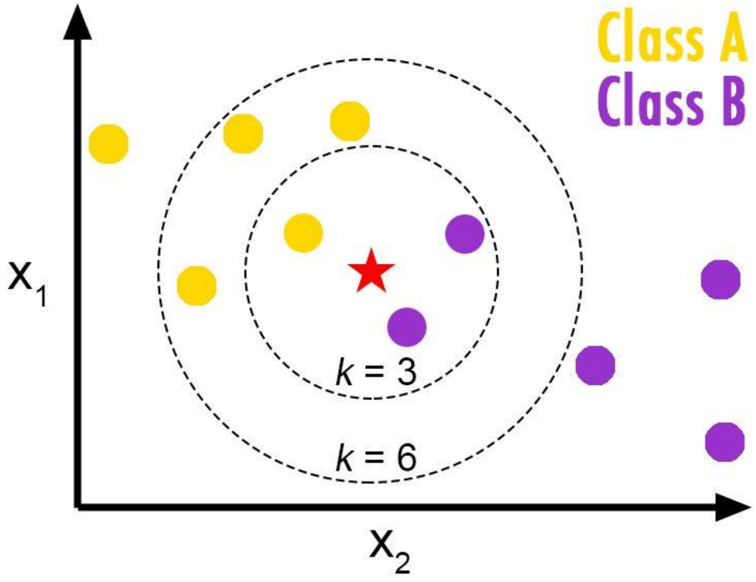
Sp Sen

200 pairs

LogReg | KNN

Area 1

KNN (K Nearest Neighbour)

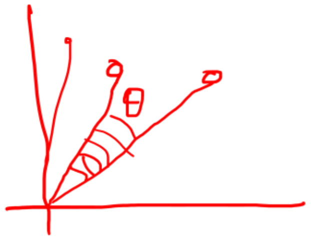


https://matlab1.com/wp-content/uploads/2017/11/knn-concept.jpg

KNN (K Nearest Neighbour)

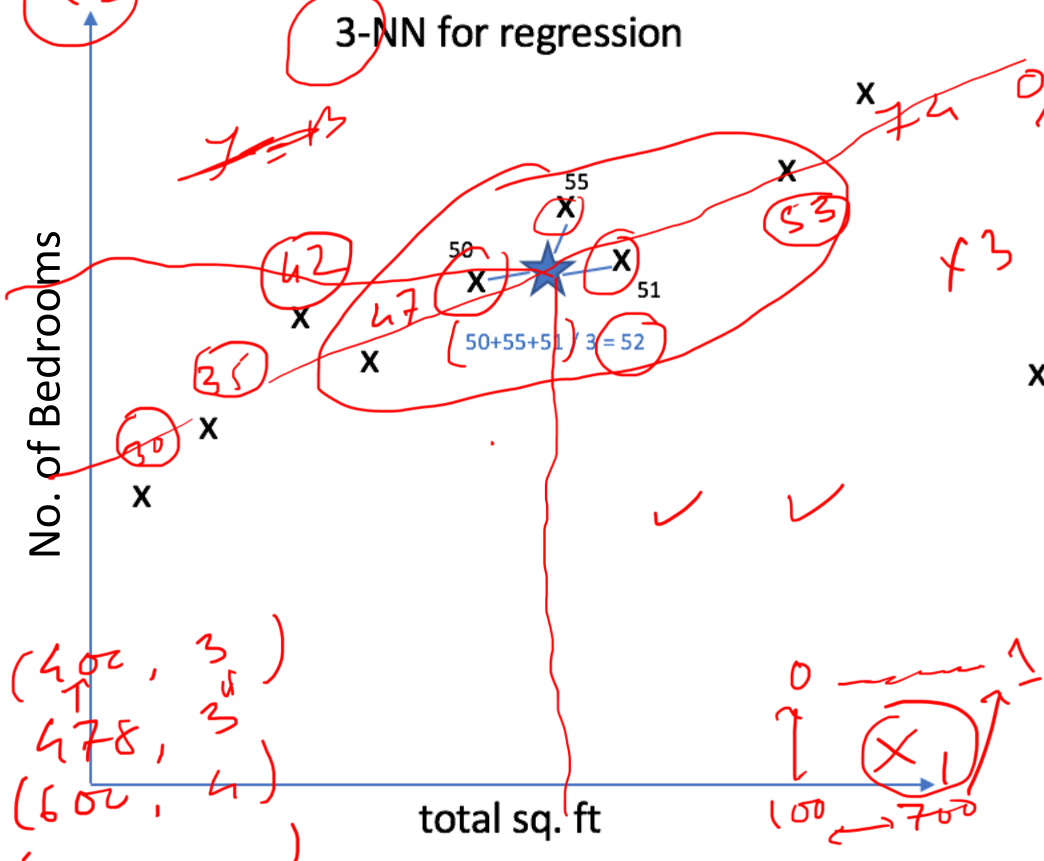
- Cartesian/Manhattan/Cosine etc
- 2D or 3D or ND
- Mean or voting
- K = ? (HP tuning)
- Which/How many IVs should we consider?
- Any feature engineering? (stan/norm/unit transform etc)

Cos θ



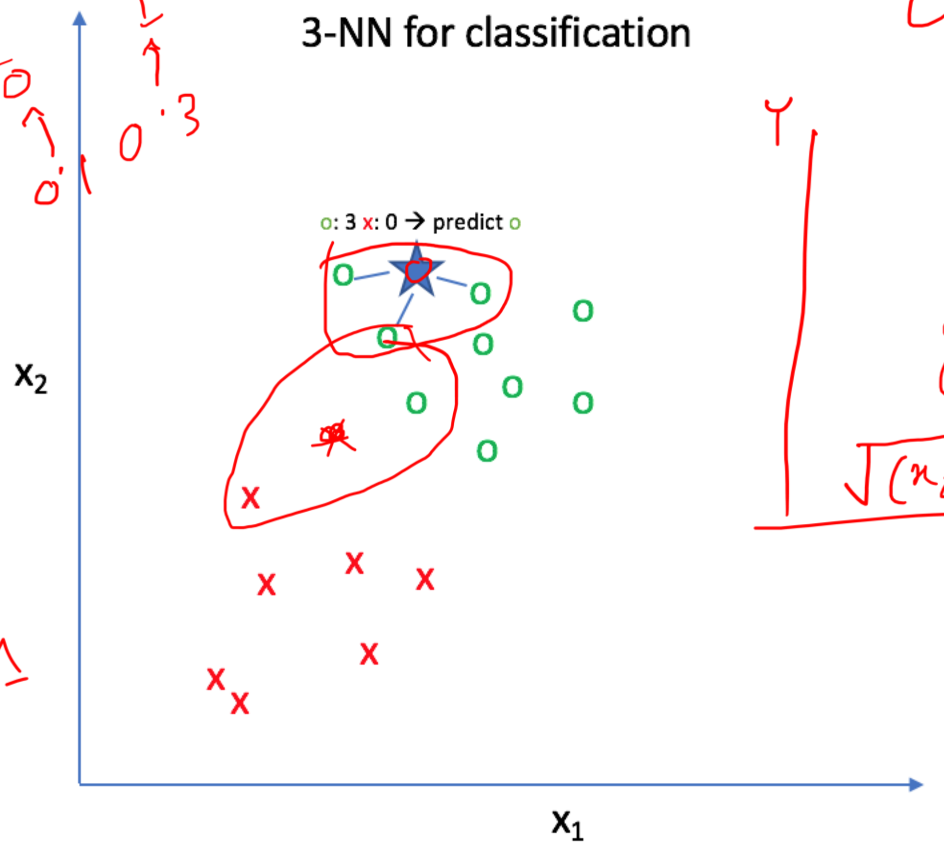
Reg Prob

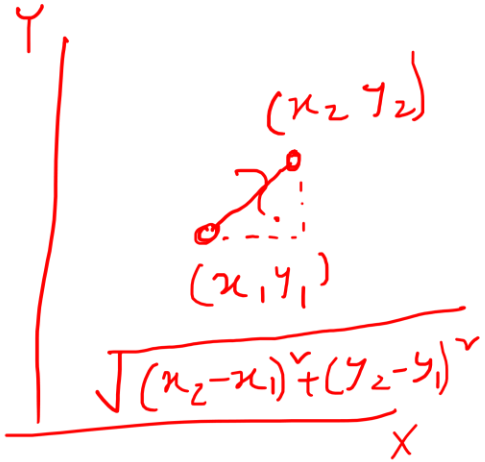
K = 5



House price prediction

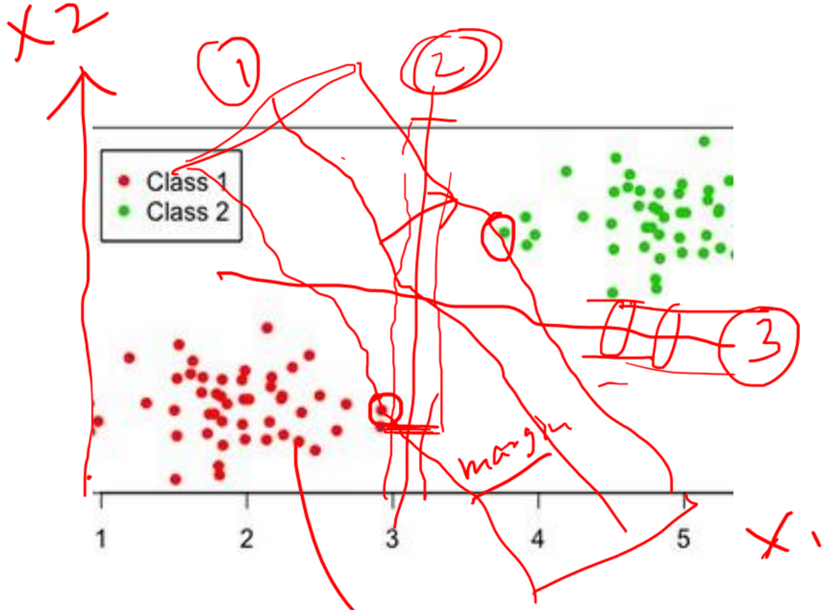
Class Prob





https://www.jeremyjordan.me/content/images/2017/06/Screen-Shot-2017-06-17-at-9.30.39-AM-1.png

SVM (Support Vector Machine)



Vector

(✓, ✓)

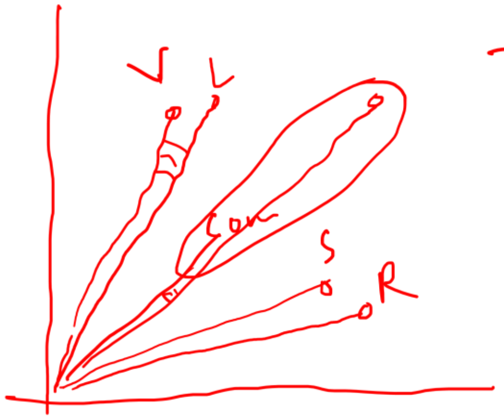
n5

R→ll

Lin

x1 . . . xn

embedding



Prompt

VAC

- GPT4
- gemini
- T5
- Bard
→ Bert

FM

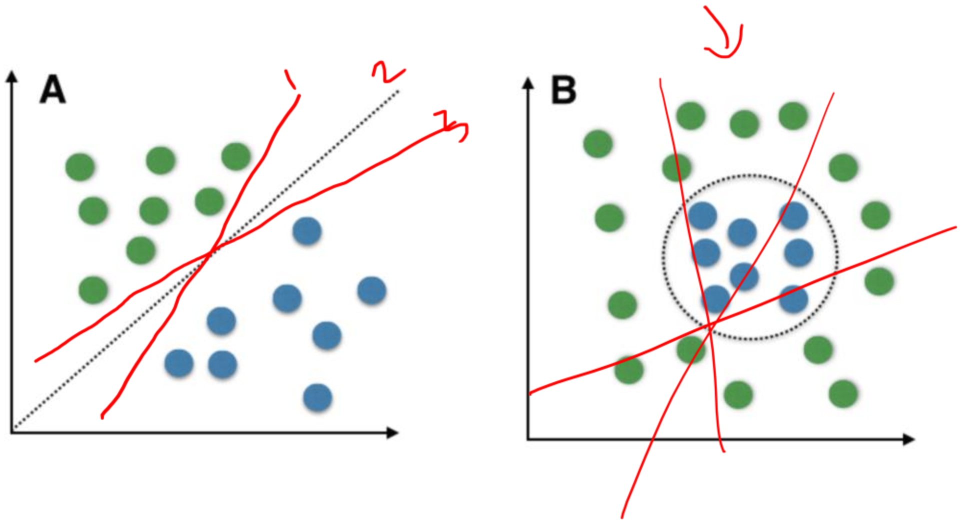
RAG

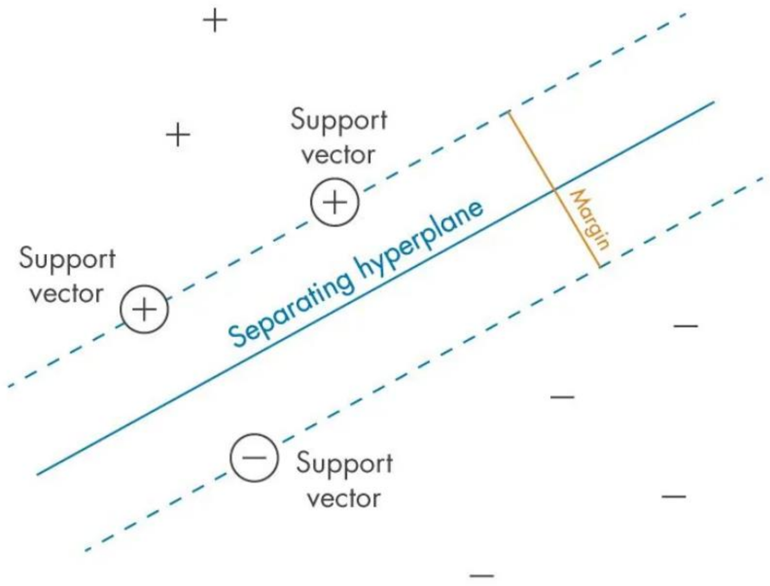
HR

docs

AI Chatbot

SVM (Support Vector Machine)





https://www.geeksforgeeks.org/introduction-to-support-vector-machines-svm/

https://medium.com/@vk.viswa/support-vector-regression-unleashing-the-power-of-non-linear-predictive-modeling-d4495836884

SVM (Support Vector Machine) – Kernel trick

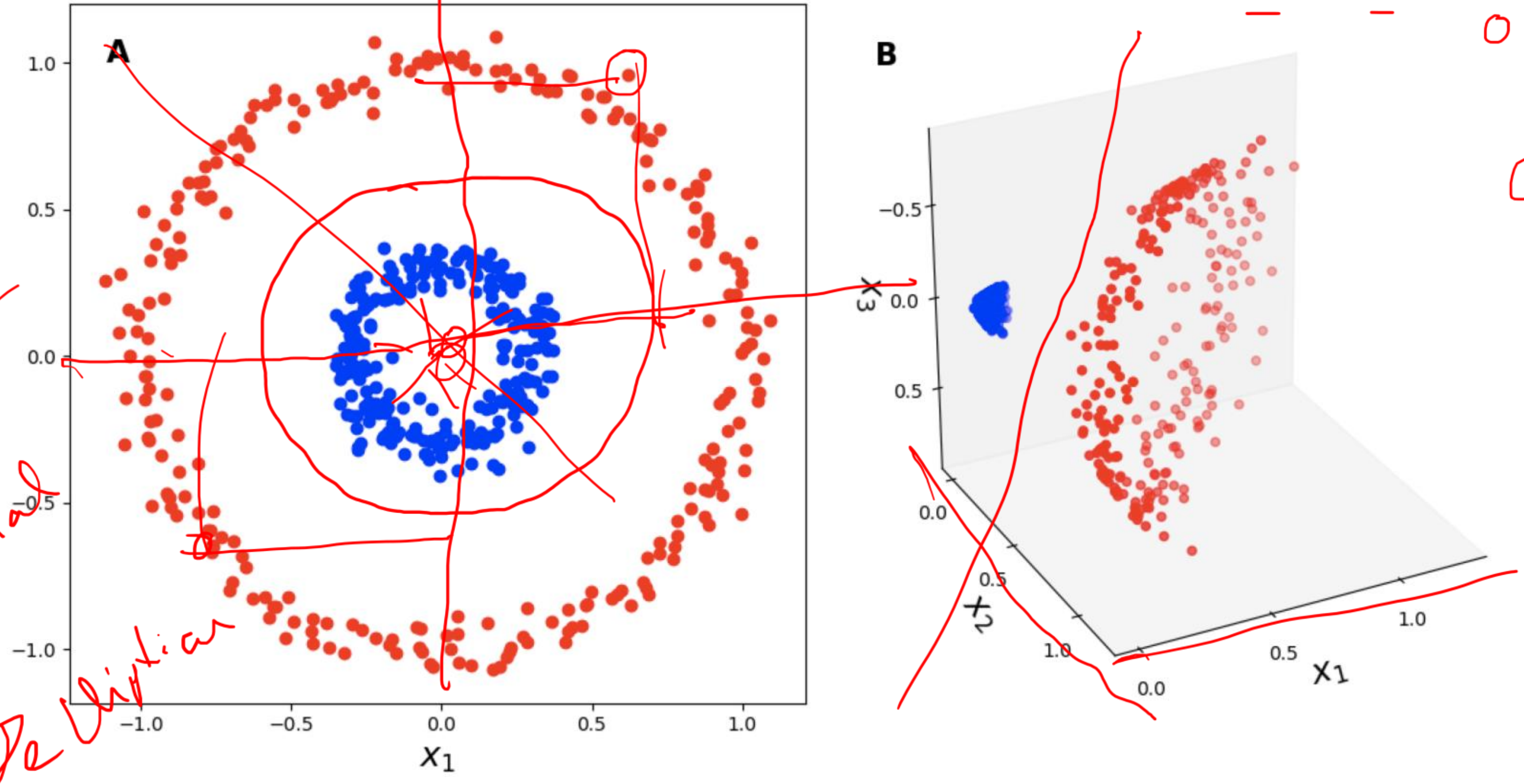
$X_3 = X_1 * X_2$

Linear

NL

Radial

elliptical



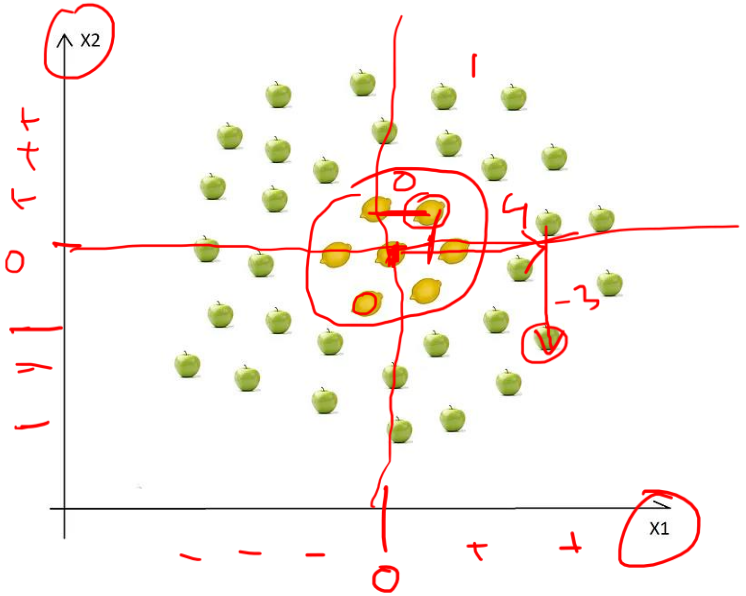
BBBR RRRR X_3

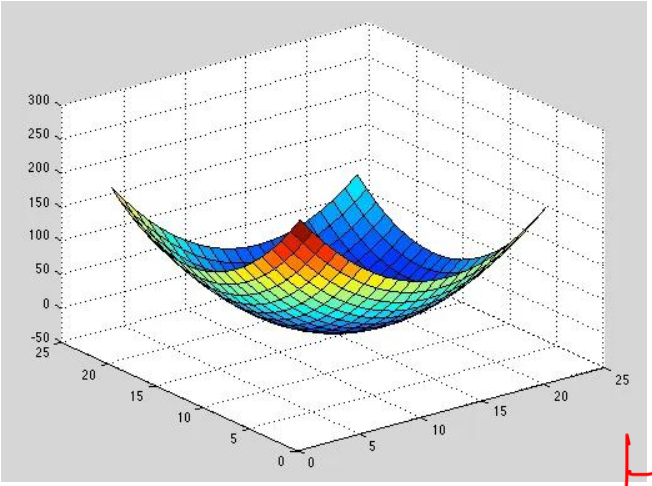
− − 0 + + +

https://gregorygundersen.com/blog/2019/12/10/kernel-trick/

SVM (Support Vector Machine) – Kernel trick

$x_3 = (x_1 * x_2)$

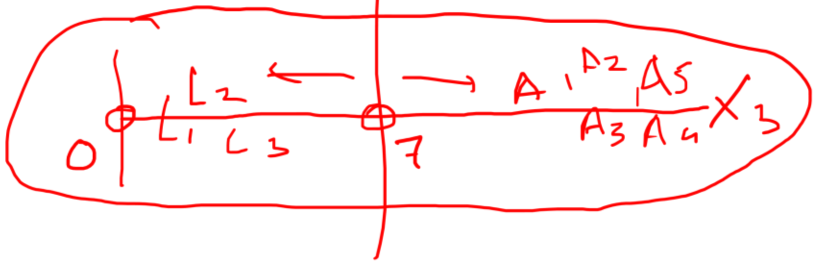




$\mathbf{x^2 + y^2}$

$x_3 = x_1^2 + x_2^2$

$x_3 = 7$



$x_3 = x_1^2 + x_2^2$

Apple $x_3 = 4^2 + (-3)^2$

$= 16 + 9$

$= 25$

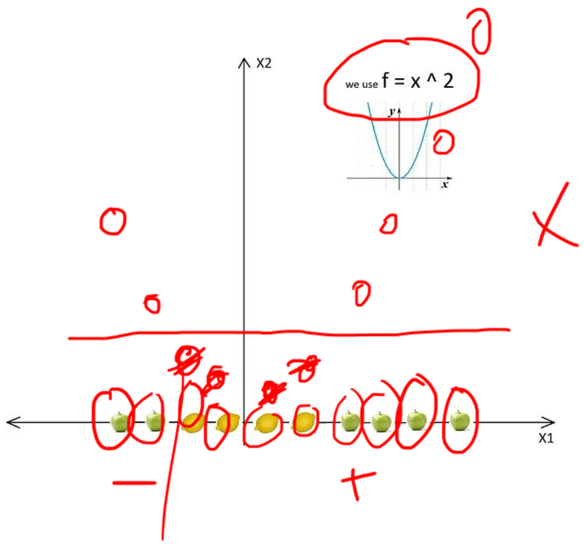
Lemon $= 2^{2.25} + 2^{2.25}$

$= 4.5$

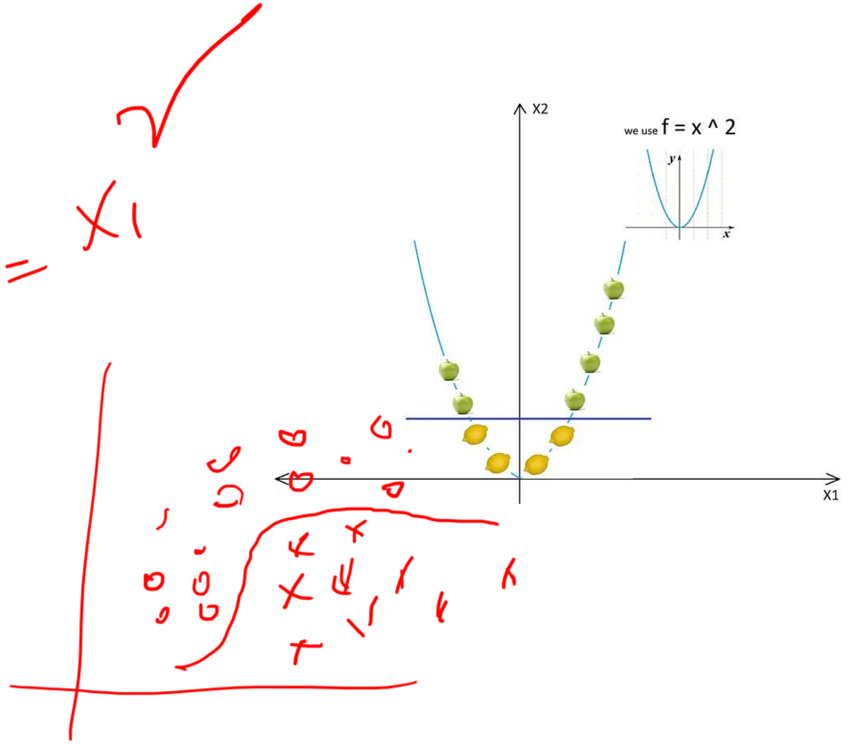
https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200

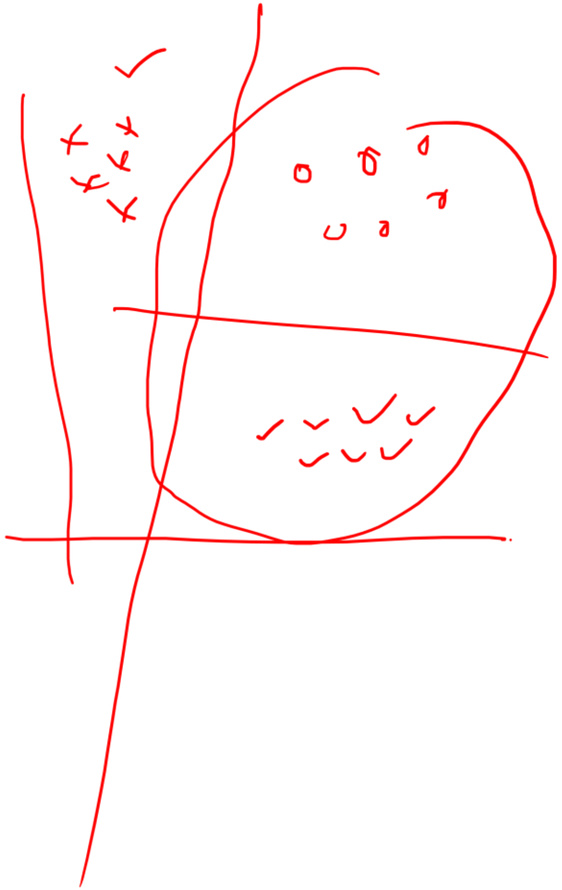
SVM (Support Vector Machine) – Kernel trick

lin / nonlinear / Radial



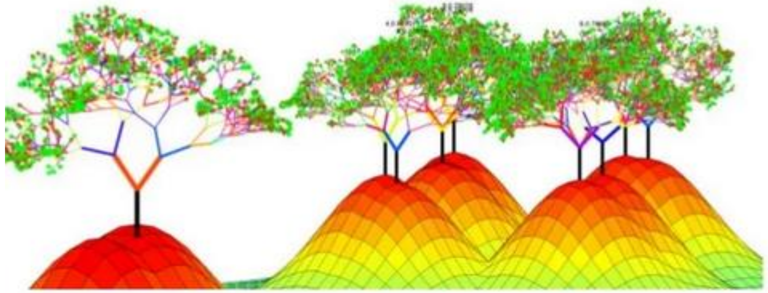
$x_2 = x_1$

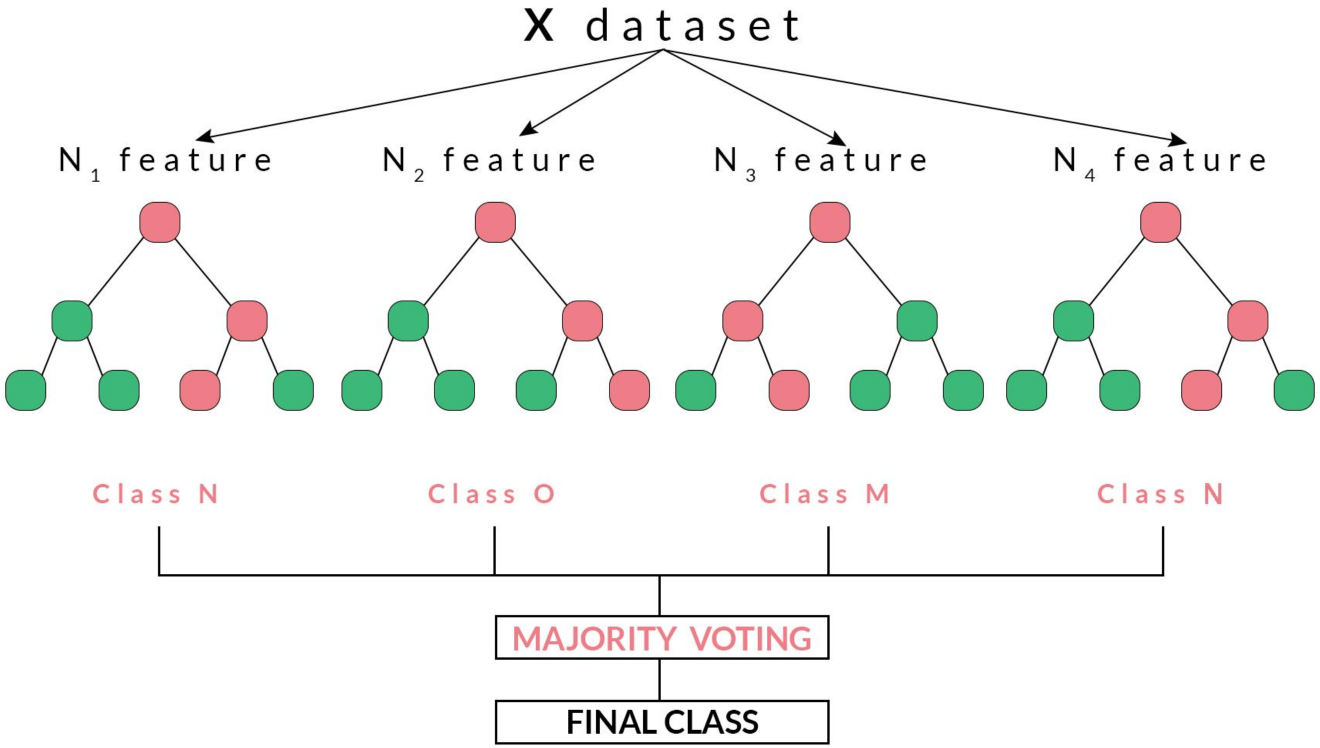




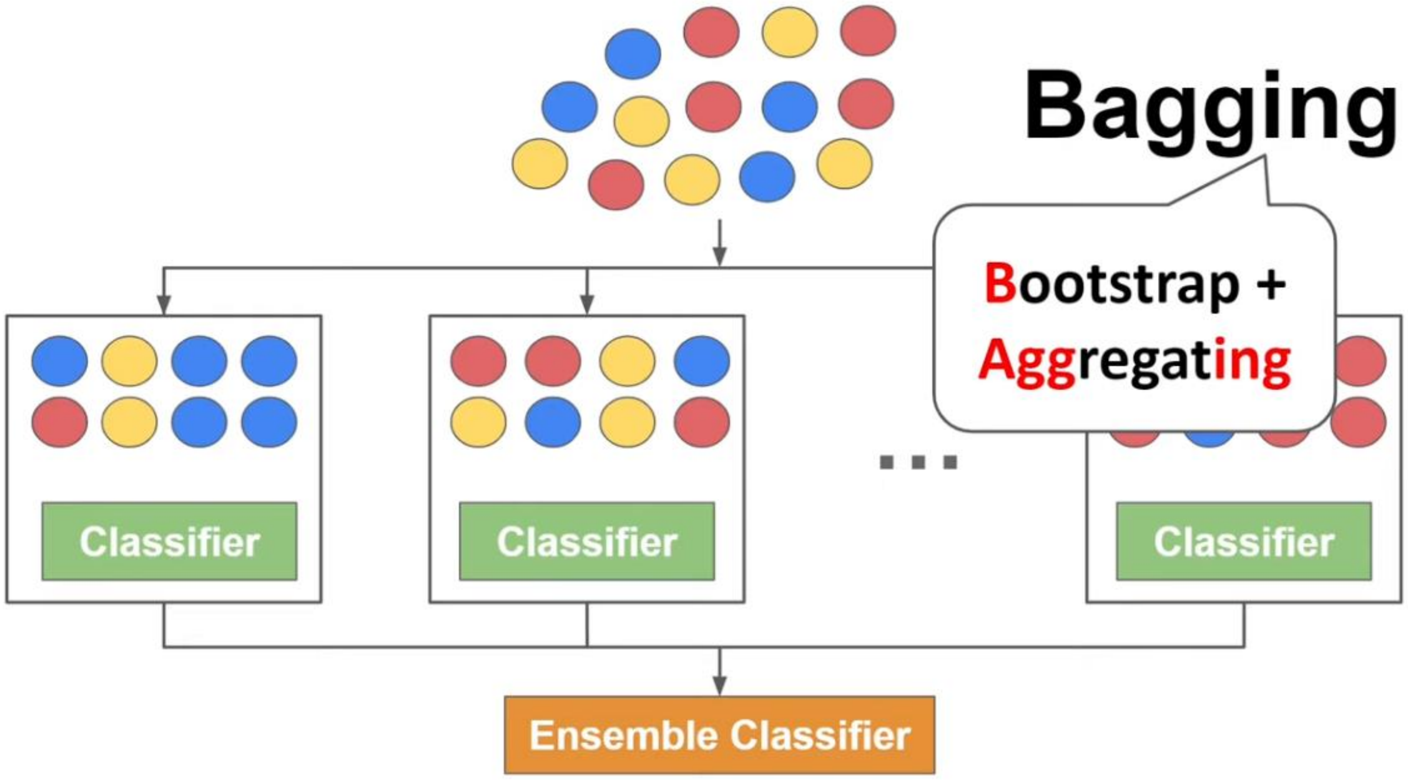
https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200

Random Forest

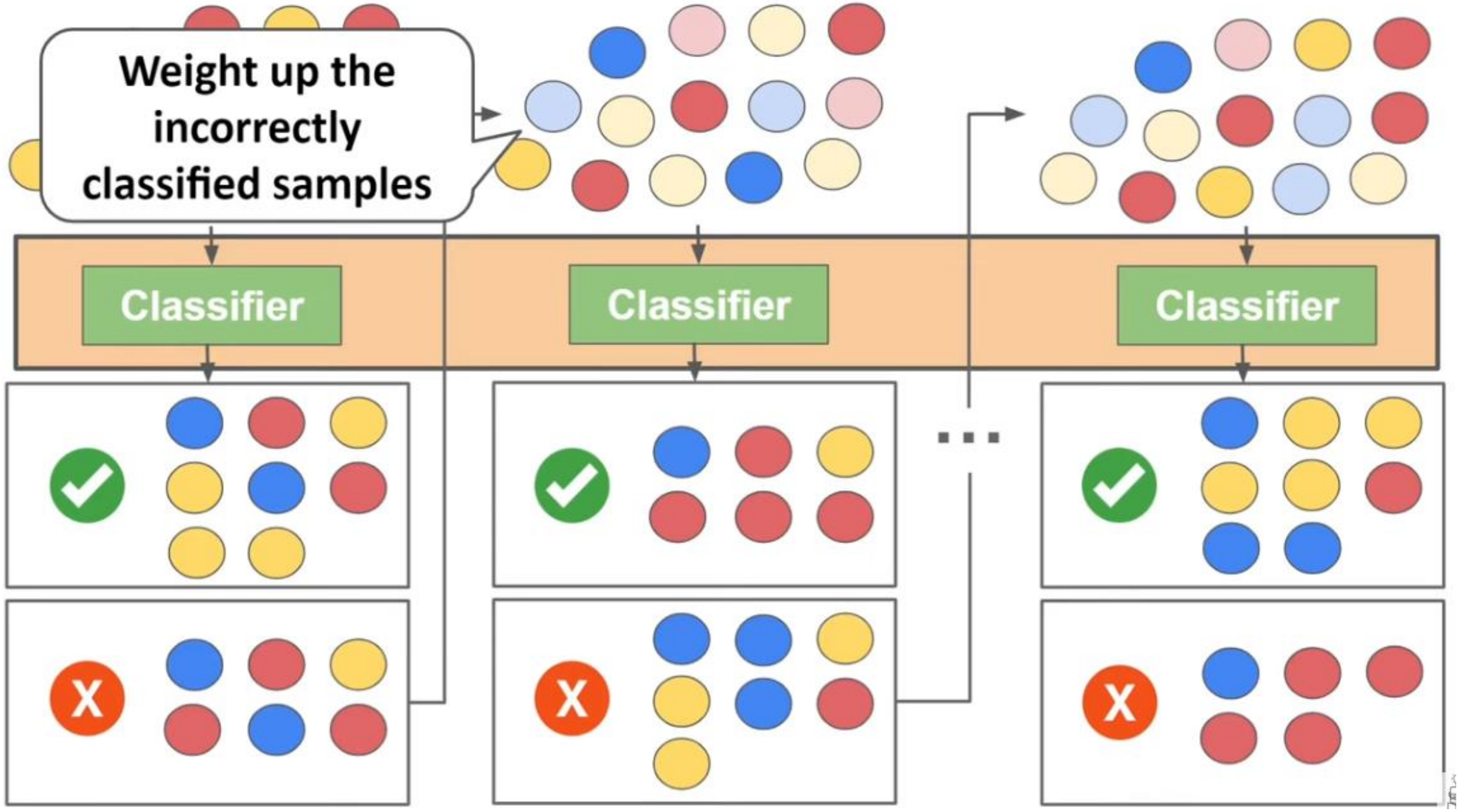




Bagging


Bagging
Bootstrap +
Aggregating
Classifier
Classifier
...
Classifier
Ensemble Classifier

Boosting



Weight up the incorrectly classified samples
Classifier
Classifier
Classifier
X
X
X

Regularization: Lasso vs Ridge vs Elastic

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s \tag{6.8}$$

and

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq s, \tag{6.9}$$