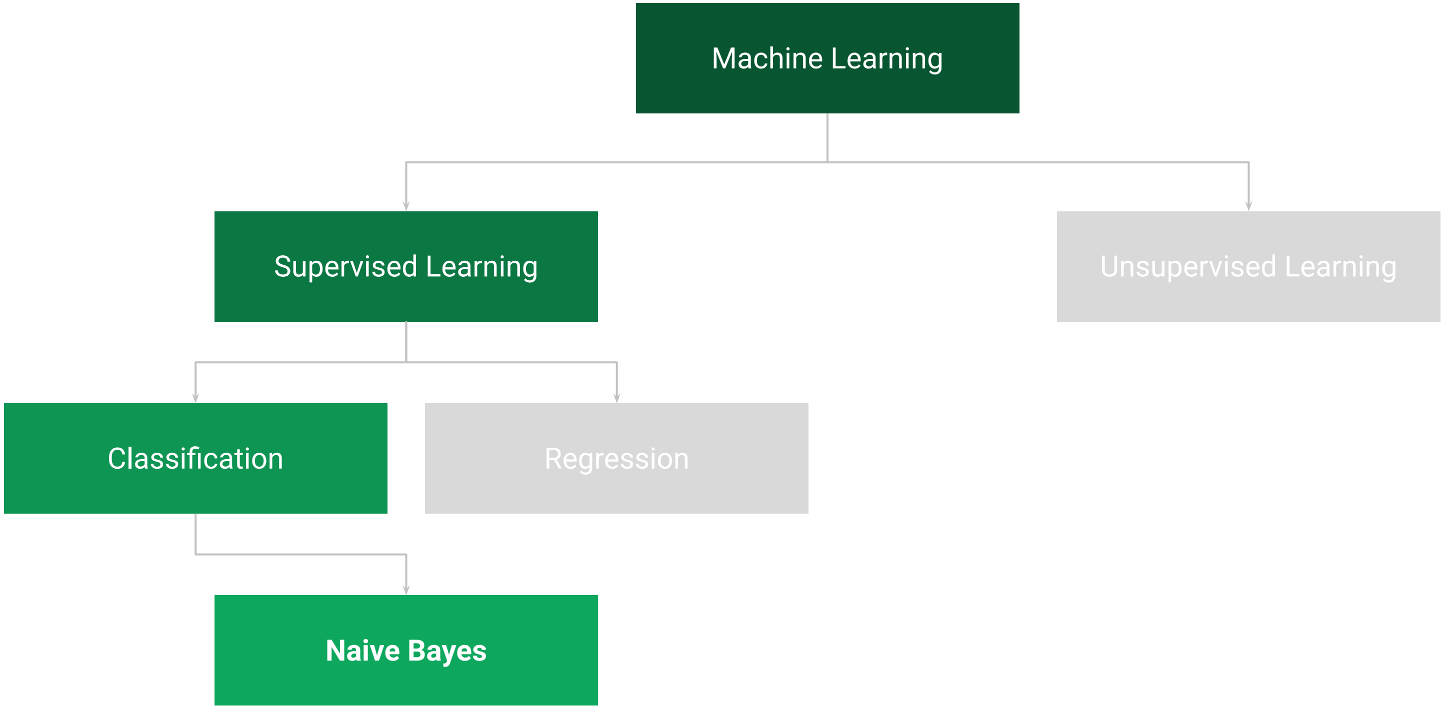
Naive Bayes


Machine Learning
Supervised Learning
Unsupervised Learning
Classification
Regression
Naive Bayes

What is Naive Bayes?

A classification method that works on the principles of conditional probability as given by the Bayes' theorem:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

Thomas Bayes
1702 - 1761

https://scienceprog.com/simple-explanation-of-naive-bayes-classifier/

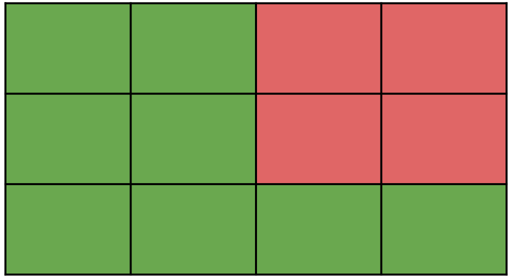
Why “Naive”?

Because it assumes that features are independent of each other.

Probabilities - Marginal Probability

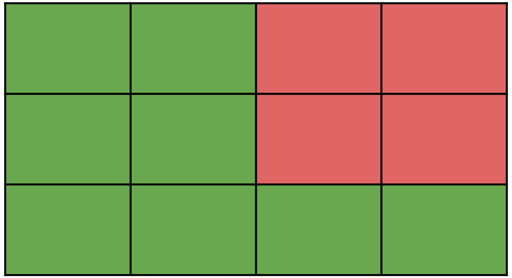
- Tweets dataset (12) labelled either: positive (8 in green) or negative (4 in red)
- **Scenario A**: probability of one event happening, P(A) or P(B)
- P(positive) = N(positive)/N(total) = 8/12 = 0.66
- P(negative) = 1 - P(positive) = 0.34

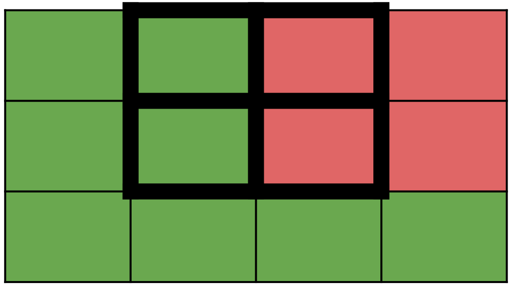
Dataset



Probabilities - Joint Probability

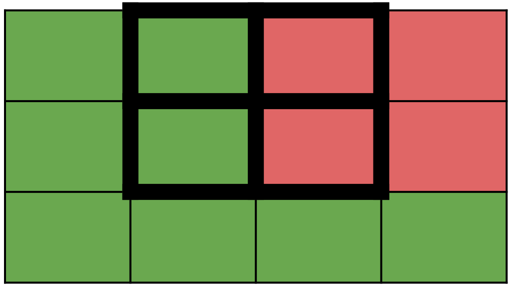
- **Scenario B**: probability of two (or more) events happening at the same time, P(A, B) or P(B, A)
- I.e. Tweets that are positive and contain the word “happy”
- Overall we have 4 tweets containing the word “happy”, 2 of which are positive tweets
- P(“happy”, positive) = P(“happy” ∩ positive) = 2/20 = 0.1

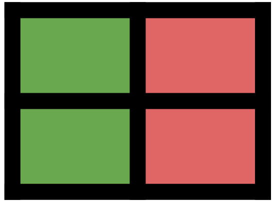




Probabilities - Conditional Probability

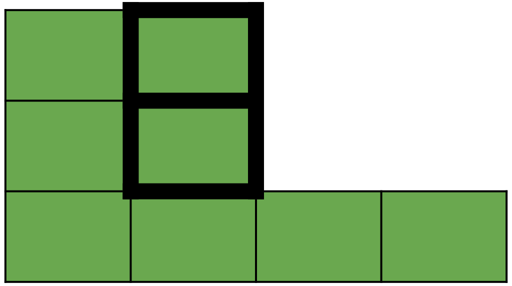
- Wikipedia: “Conditional probability is a measure of the probability of an event occurring, given that another event has already occurred”
- **Scenario C**: probability of one (or more) events happening given an occurrence of another event, P(A given B) or P(A | B)
- I.e. Tweets that are positive given it contains the word “happy”
- Use joint probability
- **P(positive | “happy”) = P(positive ∩ “happy”)/P(“happy”)** = 2/4 = 0.5





Probabilities - Conditional Probability

- **P("happy" | positive) = P("happy" ∩ positive)/P(positive)** = 2/8 = 0.25



- Note:
  P(A | B) != P(B | A), but P(A ∩ B) = P(B ∩ A)

Introducing Bayes’ Rule

- We saw that:
  - **P(“happy” | positive) = P(“happy” ∩ positive)/P(positive)**
  - **P(positive | “happy”) = P(positive ∩ “happy”)/P(“happy”)**

- Now we can derive Bayes’ rule:
  - **P(positive | “happy”) = P(“happy” | positive)*P(positive)/P(“happy”)**

Bayes' Rule

In general:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

- P(X|Y) => **Posterior Probability**; Probability of X given Value of Y.
- P(Y|X) => **Likelihood** of Y given X is True.
- P(X) => **Prior Probability**; Probability of event X.
- P(Y) => **Evidence**; Probability of event Y.

Posterior = Likelihood * Prior / Evidence

Bayes' Rule

What is the probability that there is fire given that there is smoke, P(Fire|Smoke)?

- Where:
  - P(Fire) is the Prior,
  - P(Smoke|Fire) is the Likelihood, and
  - P(Smoke) is the evidence:

$$P(Fire|Smoke) = P(Smoke|Fire) * P(Fire) / P(Smoke)$$

!!!QUESTION ALERT!!!

What is the probability that the sentence “**ML Bootcamp is awesome!**” is **positive**?

Given that:

40% of positive tweets contain the word “**awesome**”,

A total of 30% of the tweet contain the word “**awesome**”, and

60% of the total number of tweets are **positive**

1. P(**positive** | “**awesome**”) = 0.2
2. P(**positive** | “**awesome**”) = 0.8
3. P(**positive** | “**awesome**”) = 0.5
4. P(**positive** | “**awesome**”) = 0.9

Naive Bayes

- Real-world: $P(C \mid A_1, A_2 \ldots A_n)$

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

- Choose value of C that maximizes the top expression
- Basically: $P(C \mid A_1 A_2 \ldots A_n) = P(A_1, A_2, \ldots, A_n \mid C) * P(C)$

Naive Bayes

How do we estimate $P(A_1, A_2, ..., A_n \mid C)$?

- Assume independence for each A given C:

$$P(A_1, A_2, \ldots, A_n \mid C) = P(A_1 \mid C)*P(A_2 \mid C) \ldots P(A_n \mid C)$$

- We can estimate $P(A_i \mid C)$

Naive Bayes

- Under the previous independence assumptions

$$p(C_k \mid x_1, \dots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

- Where Z is the evidence, P(x), and dependent only on x_1, …, x_n which is a constant if feature variables are known

Naive Bayes Classifier

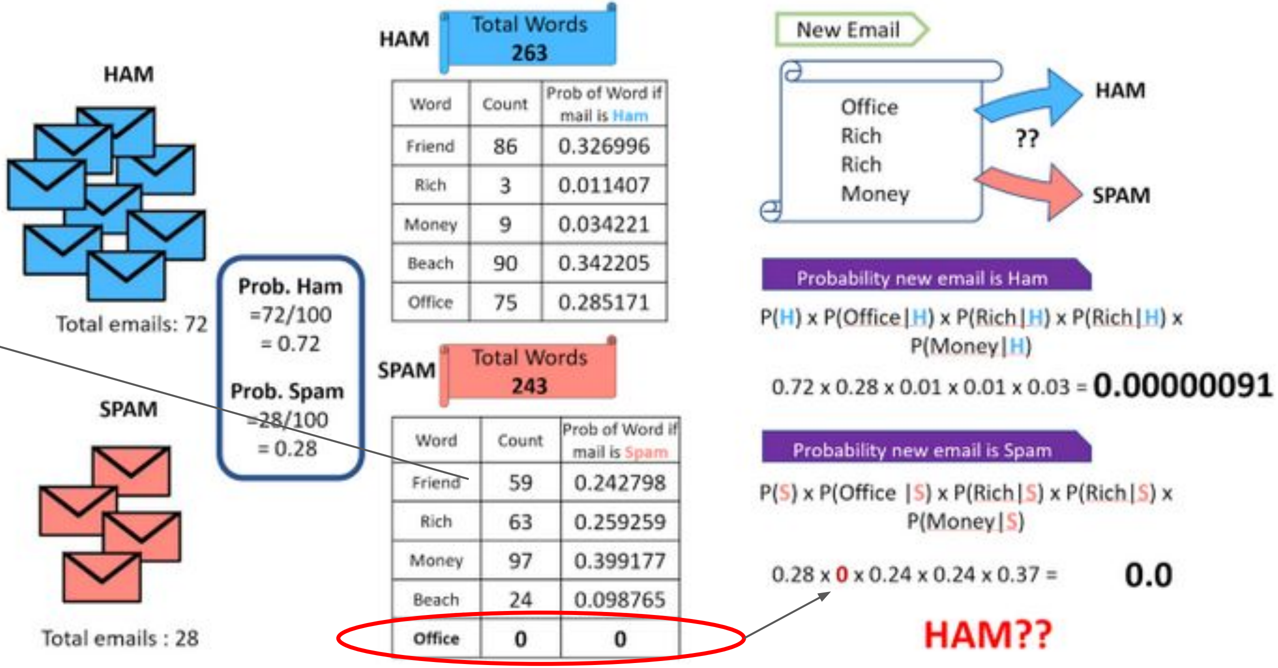
- With the previous model combined with a decision rule (*maximum a posteriori* or *MAP*) we get our classifier:

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \ p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

The Zero-Probability Problem!

- When an instance in test dataset has a category that was not present during training.

59/243 = 0.242798



https://medium.com/atoti/how-to-solve-the-zero-frequency-problem-in-naive-bayes-cd001cabe211

The Zero-Probability Problem!

- Solution: Laplace smoothing
- Add one to the count for every attribute value-class combination
- Without smoothing:

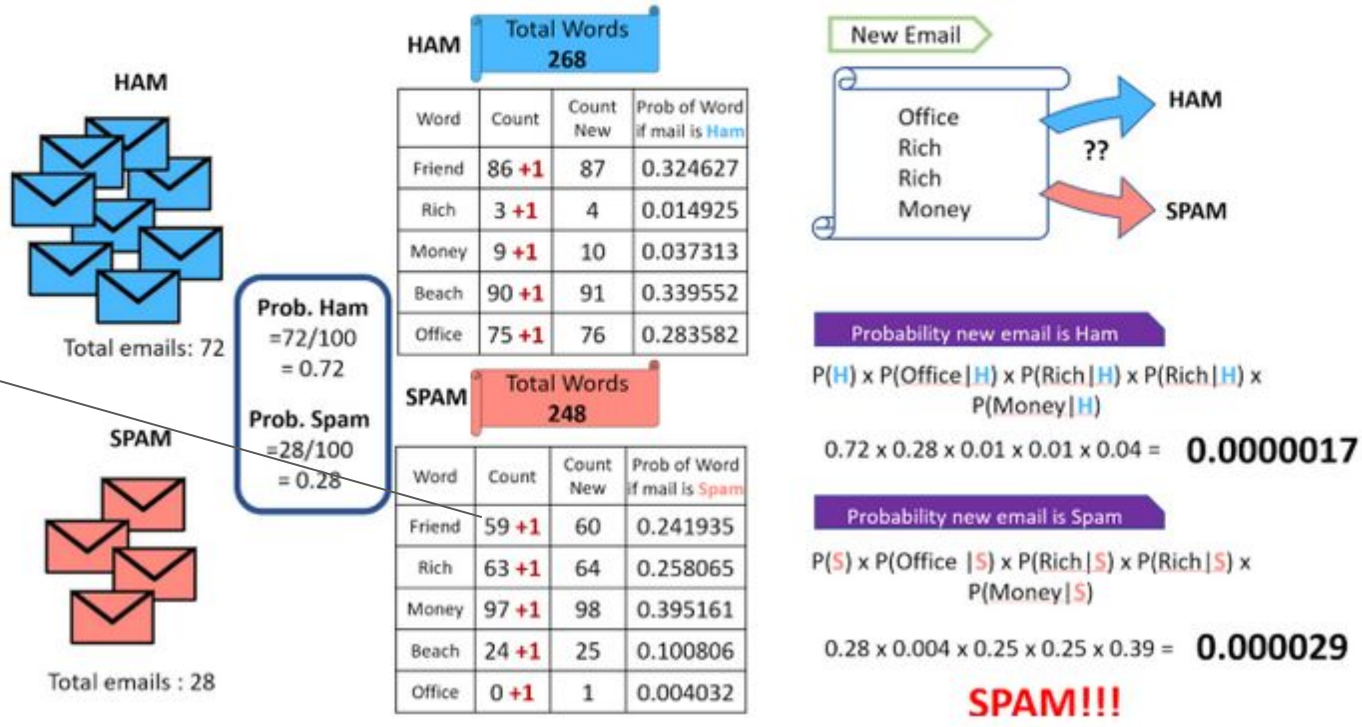
$$P(A_i = a \mid C = c) = \frac{N_{ac}}{N_c}$$

- With smoothing:

$$P(A_i = a \mid C = c) = \frac{N_{ac} + 1}{N_c + N_i}$$

The Zero-Probability Problem!

60/243+5 = 0.241935



https://medium.com/atoti/how-to-solve-the-zero-frequency-problem-in-naive-bayes-cd001cabe211

Vanishing Value Problem!

- Happens when multiplying many very small numbers
- Number get very small and close to zero => numerical underflow
- Can fix by computing the logarithm of the conditional probability

$$\log P(C|A) \sim \log P(A|C) + \log P(A)$$
$$= \sum_{i} \log(A_i|C) + \log P(A)$$

Handling Continuous Attributes

- One way is to discretize each continuous attribute and then replace the continuous attribute value with its corresponding discrete interval
- Another way is by assuming a certain form of probability distribution for the continuous variables and estimate the parameters of the distribution using the training data:

$$P(X|Y=c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{\frac{-(x-\mu_c)^2}{2\sigma_c^2}}$$

Naive Bayes Classifiers

- Multinomial Naive Bayes
  - Assumes discrete features
- Bernoulli Naive Bayes
  - Assumes binary features
- Gaussian Naive Bayes
  - Assumes continuous features

Advantages

- Relatively simple to understand and implement
- Fast to train and classify compared to other models
- Can perform better than other models on a small dataset
- Not sensitive to irrelevant features
- Handles both continuous and discrete features
- Can be used for binary or multi-class problems
- Works well with high dimensions

Disadvantages

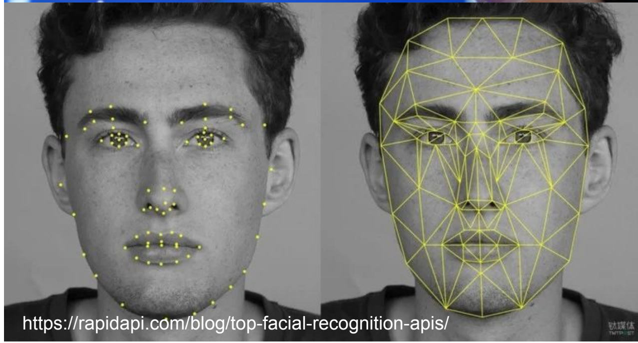
- Naive! assumes all features are independent
- Treats all attributes equally
- Vanishing value
- Zero probability problem
- Smoothing is an over-head
- Continuous value attribute problem

Trending Applications

- Text classification, i.e.:
  - Spam filtering
  - Sentiment analysis
- Medicine, i.e.:
  - Disease detection based on genome wide data
- Image classification, i.e.:
  - Face recognition

https://towardsdatascience.com/spam-filtering-using-naive-bayes-98a341224038


https://www.genengnews.com/insights/cytogenomic-analyses-for-genetic-disease-detection-and-diagnosis/


https://rapidapi.com/blog/top-facial-recognition-apis/

Support Vector Machines

Hayden Fiege E.I.T.

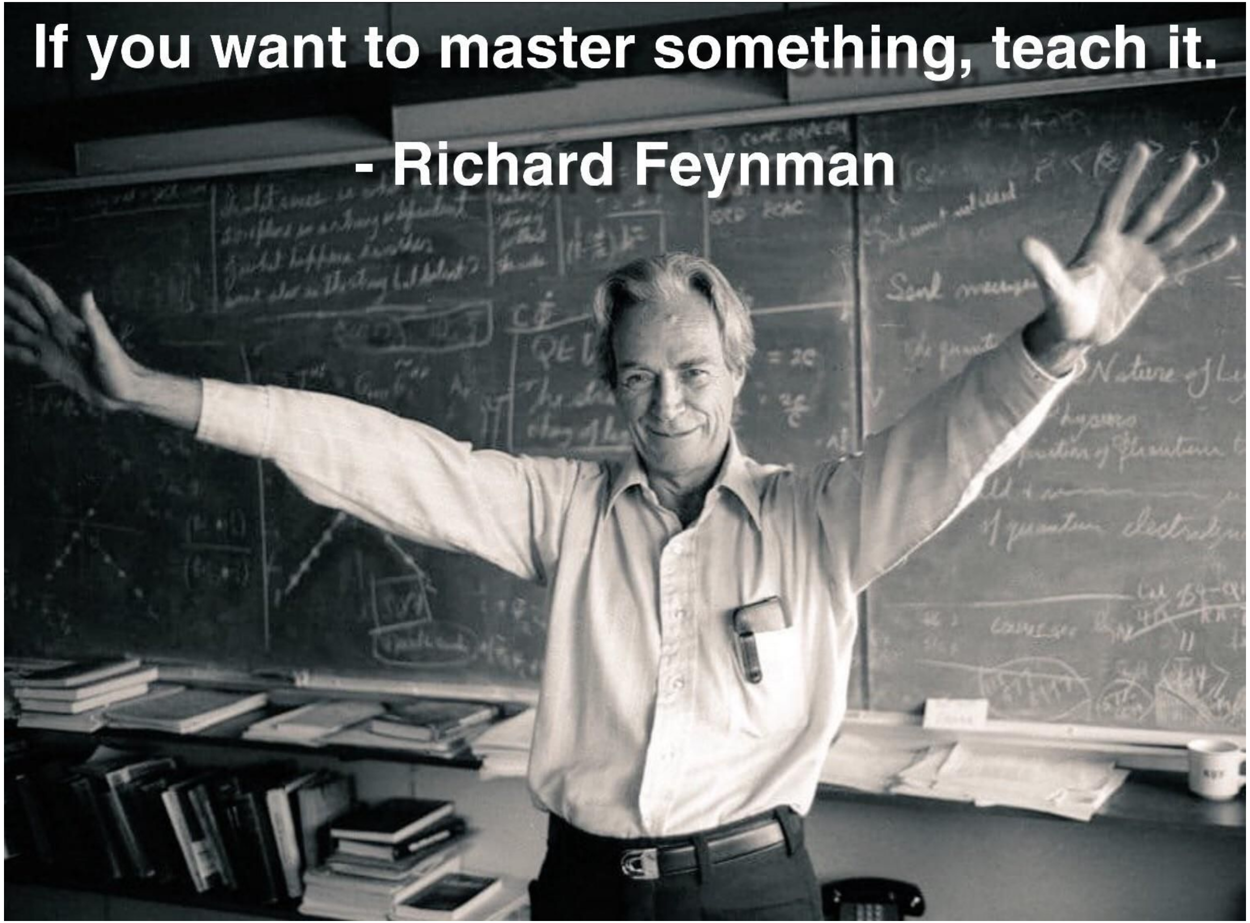


Image Source: https://i.redd.it/27jroi47gqy21.jpg

What is SVM?

- A powerful and versatile ML model developed by Vladimir Vapnik and colleagues in the early 90’s. SVM can be used for linear and non-linear classification and regression, for outlier detection, for clustering and more.

$$\min_{\beta^{\wedge},\beta^{\vee}} \frac{1}{2}\sum_{i=1,j=1}^{m}\left(\beta_i^{\wedge}-\beta_i^{\vee}\right)\left(\beta_j^{\wedge}-\beta_j^{\vee}\right)K_{ij}+\sum_{i=1}^{m}\left(\varepsilon-y^{(i)}\right)\beta_i^{\wedge}+\left(\varepsilon+y^{(i)}\right)\beta_i^{\vee}$$

$$s.t: \sum_{i=1}^{m}\left(\beta_i^{\wedge}-\beta_i^{\vee}\right)=0$$

$$0<\beta_i^{\wedge}<C, i=1,2,\ldots,m$$

$$0<\beta_i^{\vee}<C, i=1,2,\ldots,m$$

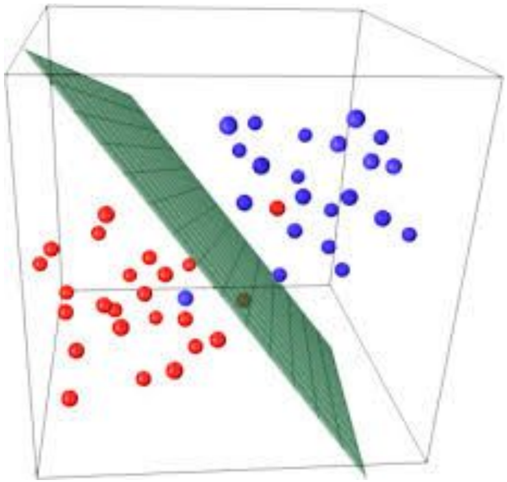
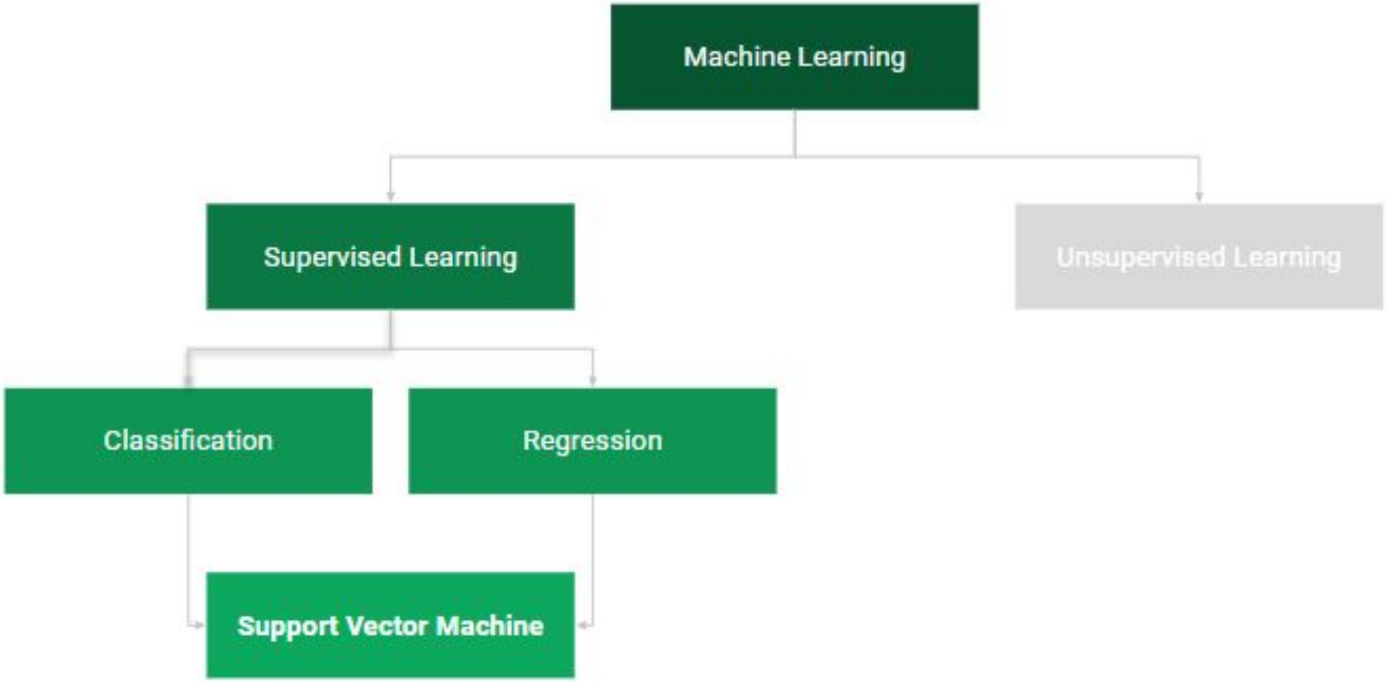


Image Source: www.programmersought.com/article/92694818372/


Machine Learning
Supervised Learning
Unsupervised Learning
Classification
Regression
Support Vector Machine

Linear SVM – Multiple Lines

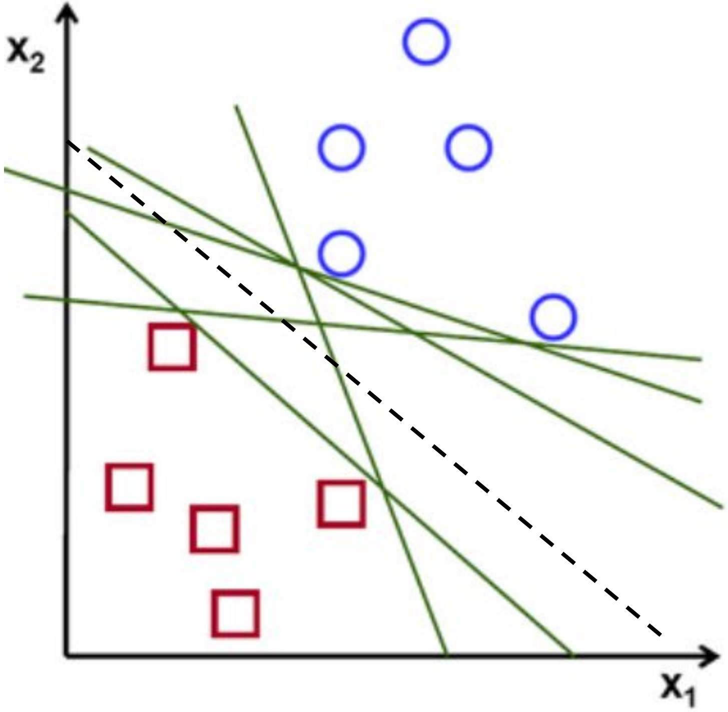


Image Source: towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f

Linear SVM – Optimal Hyperplane

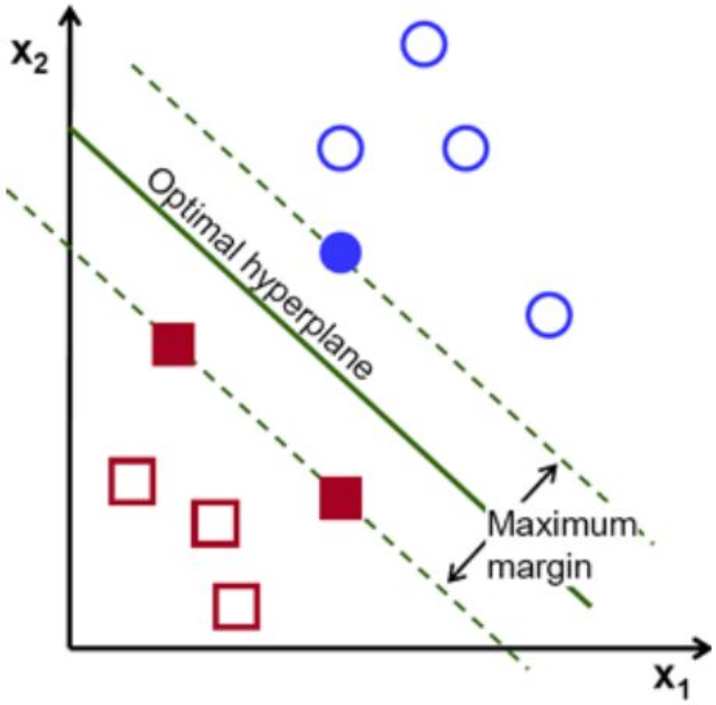
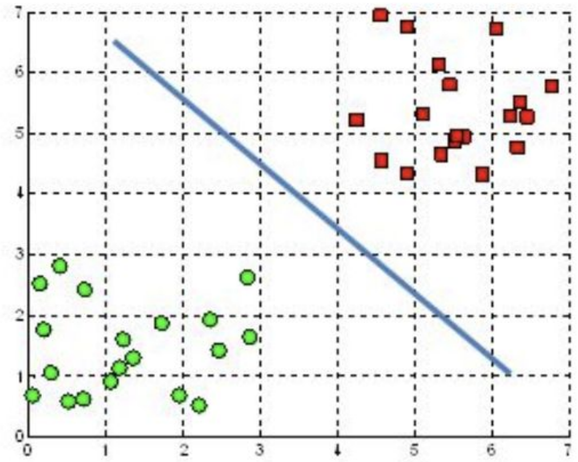


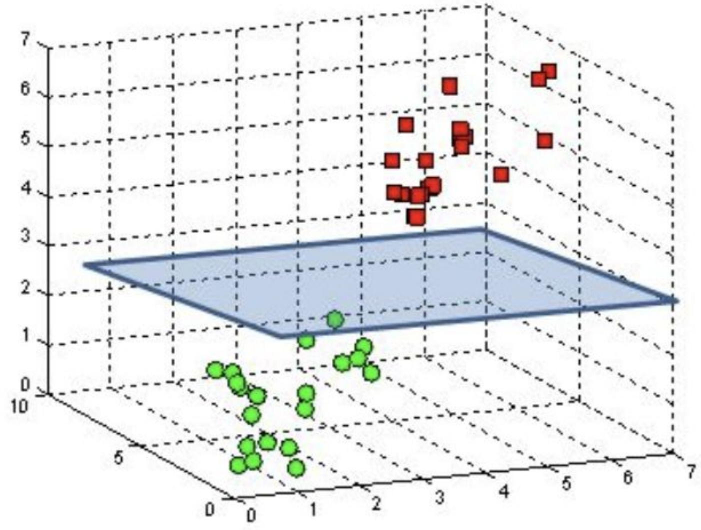
Image Source: towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f

What is a hyperplane?

A hyperplane in $\mathbb{R}^2$ is a line



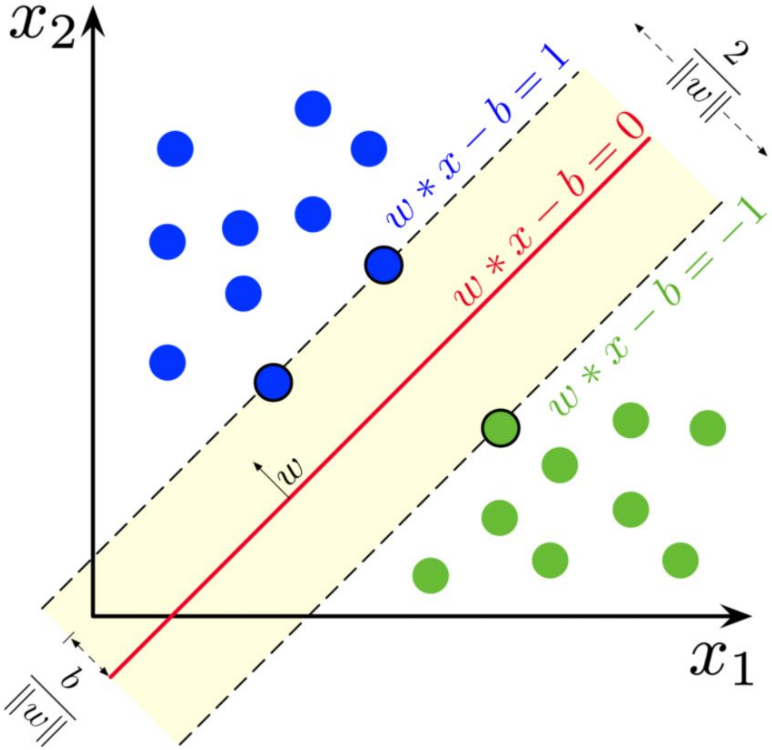
A hyperplane in $\mathbb{R}^3$ is a plane



$$Hyper\ plane\ equation\ \rightarrow w \cdot x + b = 0$$

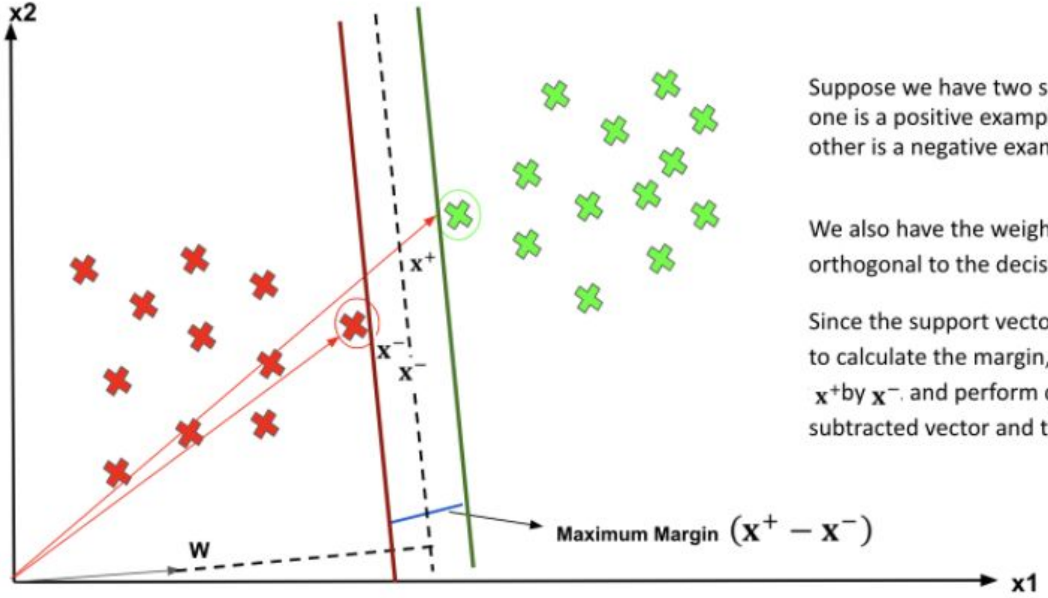
Image Source: www.kdnuggets.com/2020/03/machine-learning-algorithm-svm-explained.html

Linear SVM



source: wikipedia (Larhmam)

Max Margin



Suppose we have two support vectors where one is a positive example (e.g., $\mathbf{x}^+$) and the other is a negative example (e.g., $\mathbf{x}^-$)

We also have the weight vector $\mathbf{w}$ which is orthogonal to the decision boundary

Since the support vectors define the margin, to calculate the margin, we can first subtract $\mathbf{x}^+$by $\mathbf{x}^-$. and perform dot product on the subtracted vector and the unit vector of $\mathbf{w}$

$$(\mathbf{x}^+ - \mathbf{x}^-) \cdot \hat{\mathbf{w}} = (\mathbf{x}^+ - \mathbf{x}^-) \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} = \mathbf{x}^+ \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} - \mathbf{x}^- \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

Image Source: www.kdnuggets.com/2020/03/machine-learning-algorithm-svm-explained.html

Max Margin

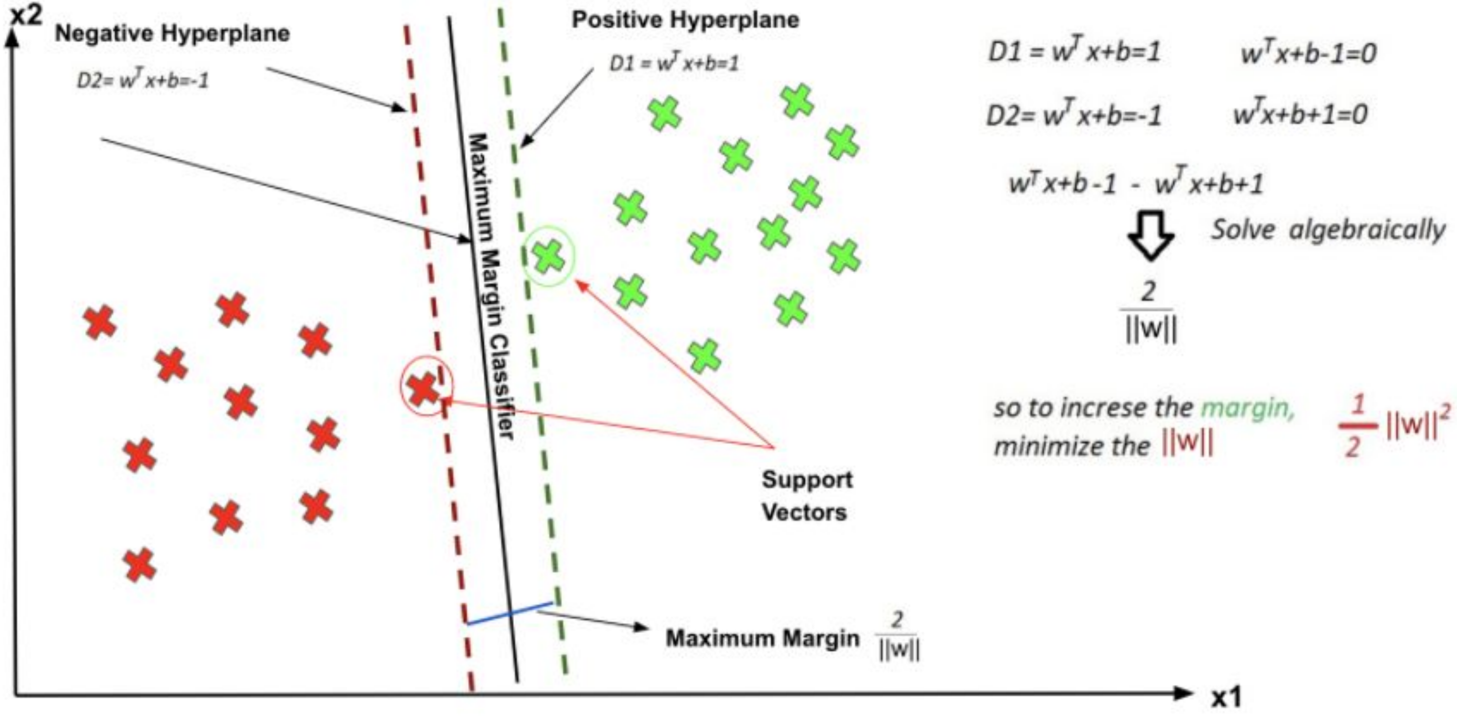


Image Source: www.kdnuggets.com/2020/03/machine-learning-algorithm-svm-explained.html

SVM objective function

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)$$

Regularization term:

- Maximize the margin
- Imposes a preference over the hypothesis space and pushes for better generalization
- Can be replaced with other regularization terms which impose other preferences

Empirical Loss:

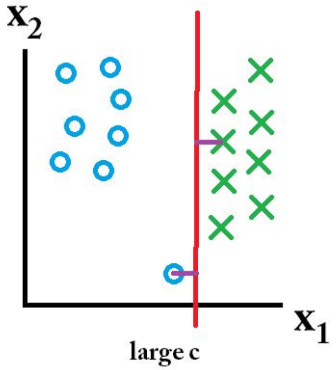
- Hinge loss
- Penalizes weight vectors that make mistakes
- Can be replaced with other loss functions which impose other preferences

A **hyper-parameter** that controls the tradeoff between a large margin and a small hinge-loss

Image Source: slideplayer.com/slide/13367887/

C hyperparameter

Large C: Lower bias, high variance
i.e. More prone to over fitting



Small C: Higher bias, low variance
i.e. More prone to under fitting

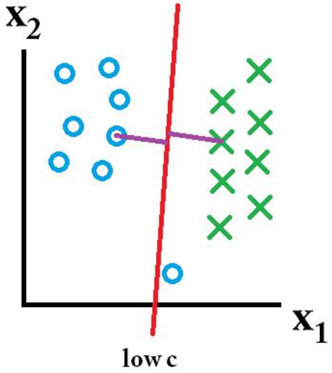
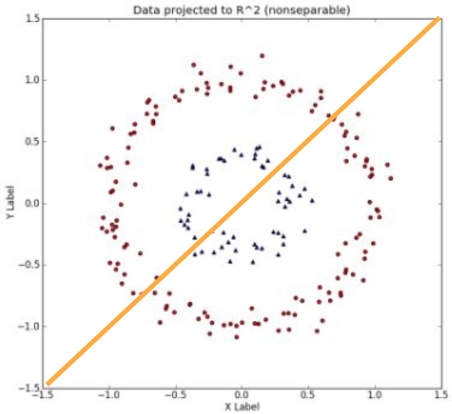
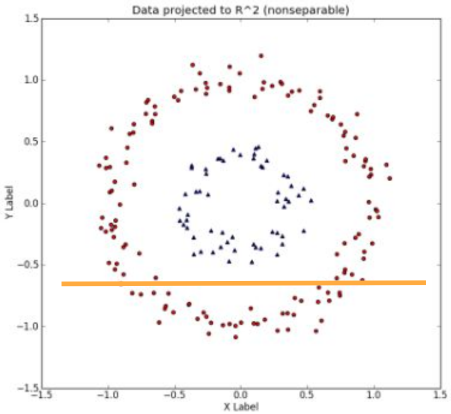


Image Source: stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel

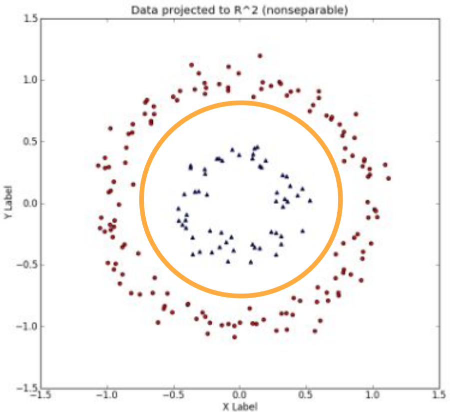
How do we separate the data with SVM?



A



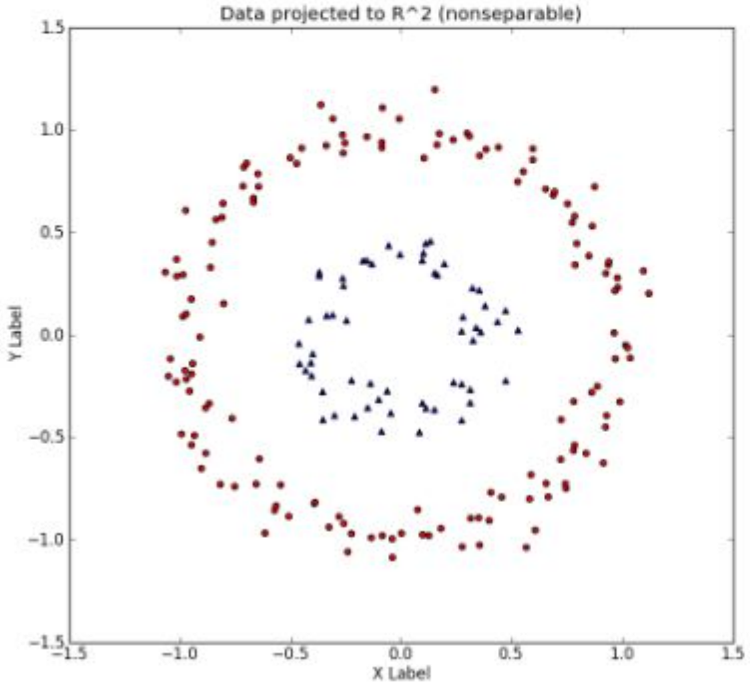
B



C

Image Source: sites.google.com/site/machinelearningnotebook2/classification/binary-classification/kernels-and-non-linear-svm

Non-linear SVM



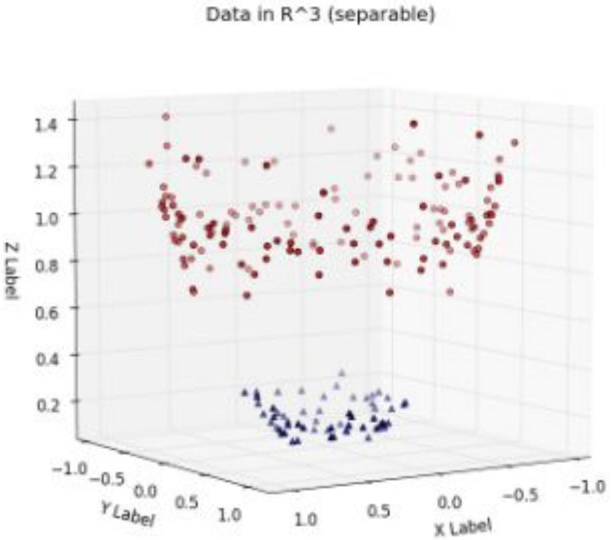


Image Source: sites.google.com/site/machinelearningnotebook2/classification/binary-classification/kernels-and-non-linear-svm

Non-linear SVM

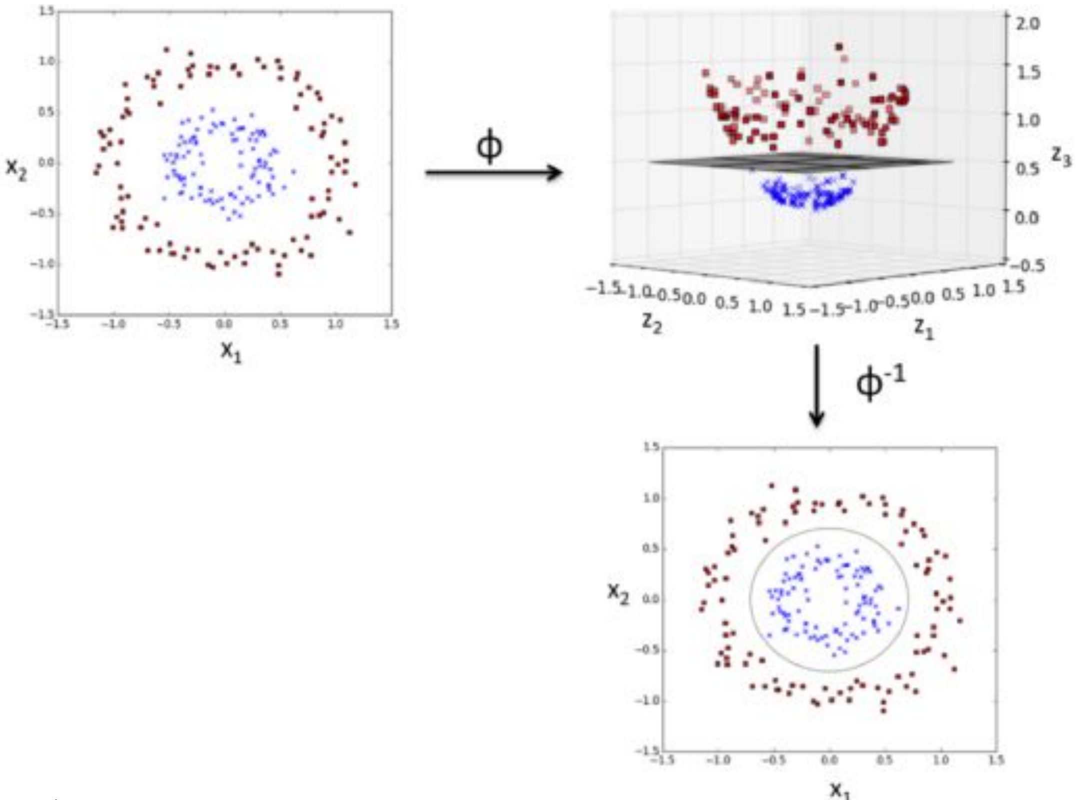


Image Source: www.bogotobogo.com/python/scikit-learn/scikit_machine_learning_Support_Vector_Machines_SVM_2.php

Kernel Trick

- It is a scientific term with the name trick in it (Exciting!)
- Provides mapping for linear learning algorithms to work on non-linear data
- The function must meet the definition provided in Mercers theorem to be a Kernel Function
- The functions employ mathematical tricks that allow the functions to operate in a higher-dimension without ever computing the coordinates of the data in that space

Some of the Kernel Types

- polynomial: $(\gamma\langle x, x'\rangle + r)^d$, where d is specified by parameter `degree`, r by `coef0`.
- rbf: $\exp(-\gamma\|x - x'\|^2)$, where γ is specified by parameter `gamma`, must be greater than 0.
- sigmoid $\tanh(\gamma\langle x, x'\rangle + r)$, where r is specified by `coef0`.

Other kernels: String kernel, chi-square kernel, histogram intersection kernel, etc.

Image Source: scikit-learn.org/stable/modules/svm.html

Main Types of Kernels

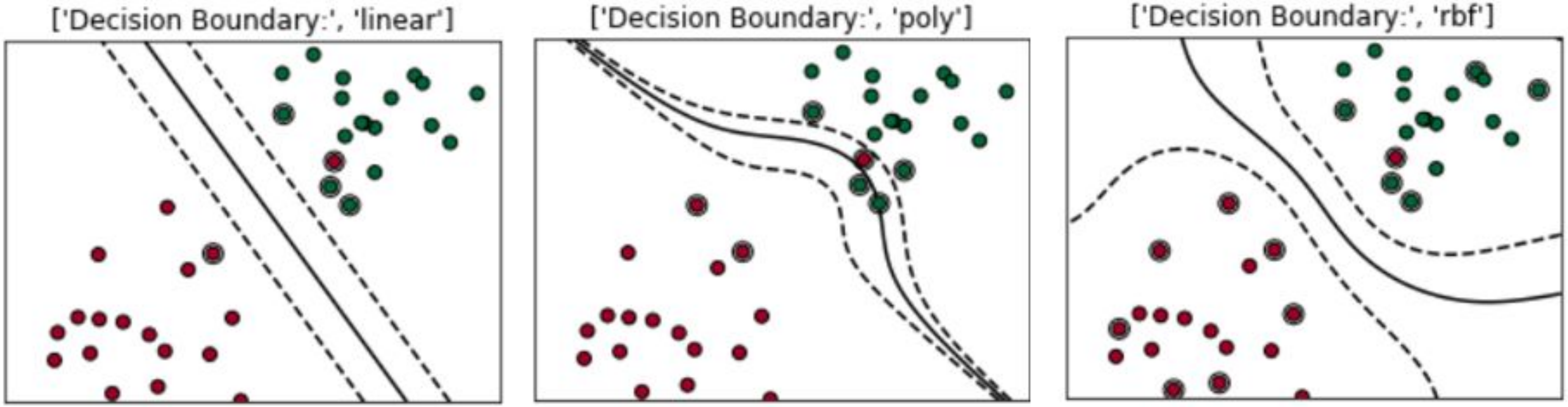
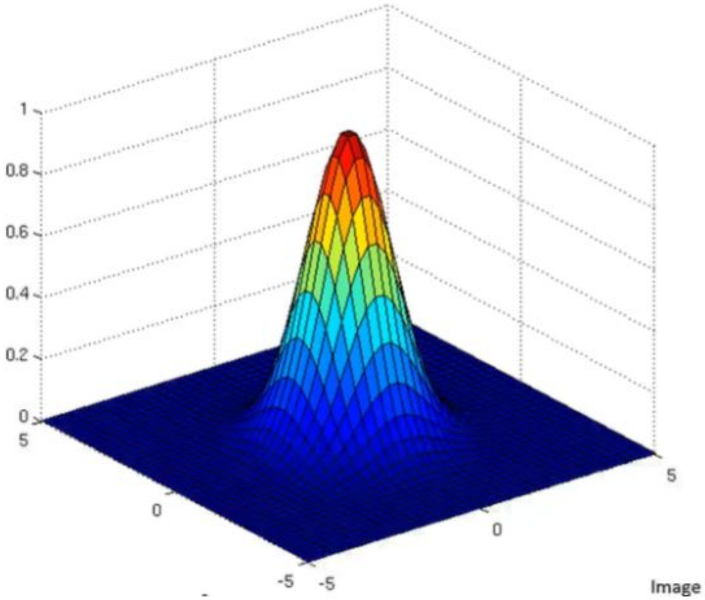


Image Source: towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496

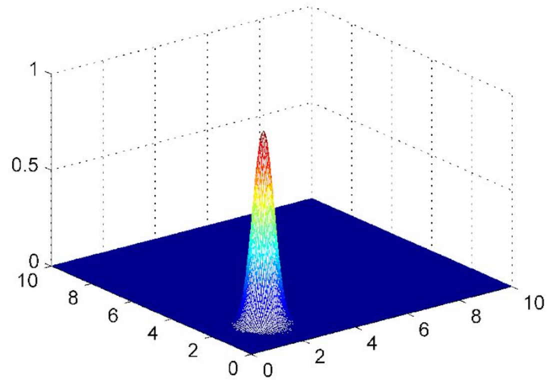
Radial Basis Function (RBF) Kernel



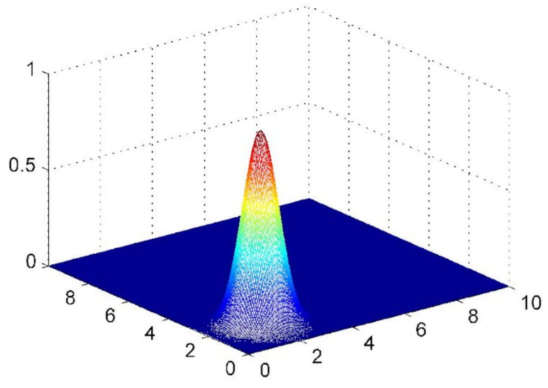
$$RBF\colon\ e^{-\gamma\|x-y\|^2}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

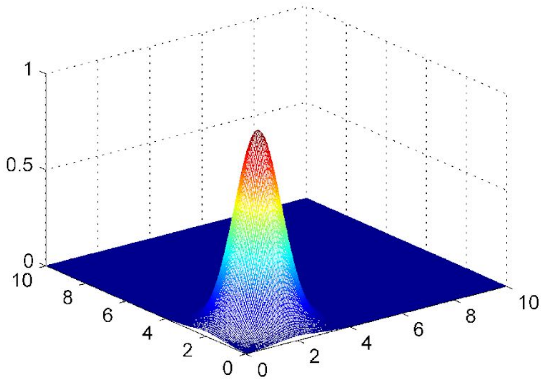
Radial Basis Function (RBF) Kernel



$\gamma = 12.5$



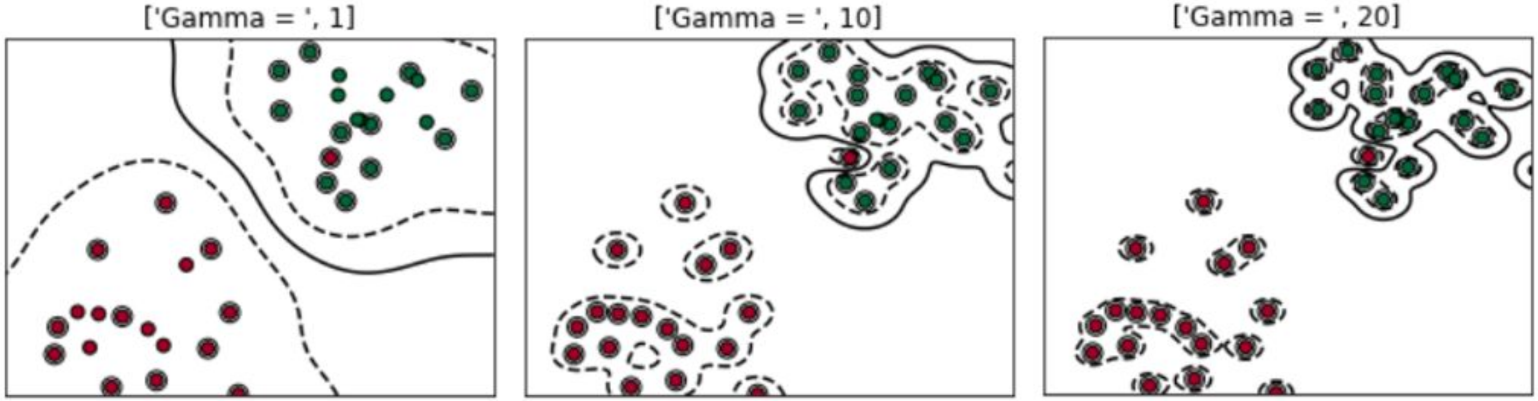
$\gamma = 1.4$



$\gamma = 0.5$

Image Source: https://www.mdpi.com/2072-4292/12/13/2154/html

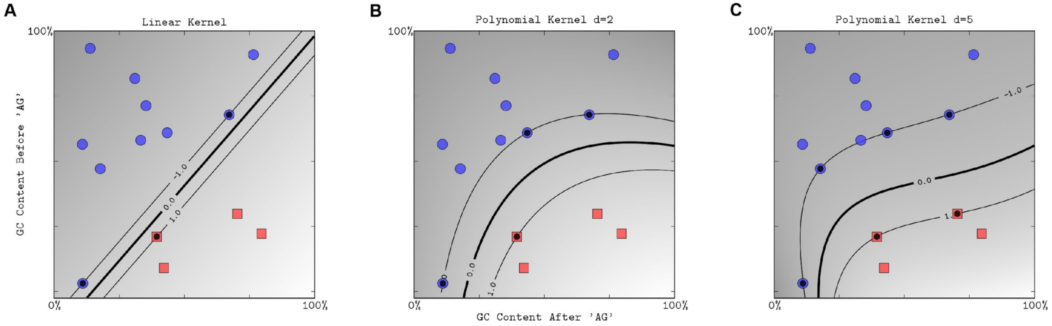
Radial Basis Function (RBF) Kernel



$\gamma = 1$ $\gamma = 10$ $\gamma = 20$

Image Source: towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496

Polynomial Kernel



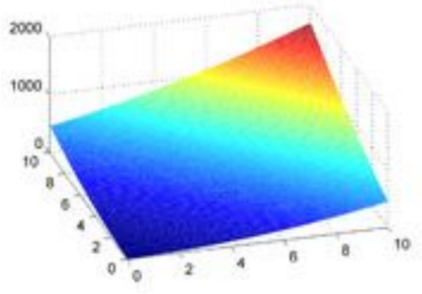
d= 2

d= 5

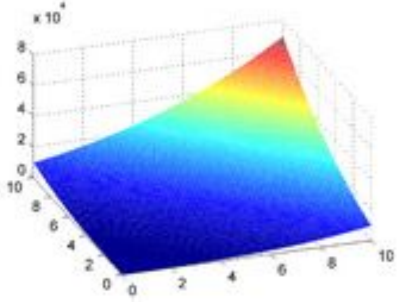
$$\text{Poly.} = (x \cdot y + r)^d$$

Image Source: journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000173

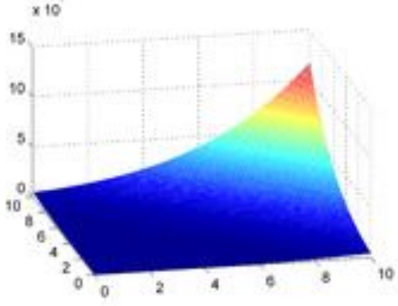
Polynomial Kernel



$d = 2$

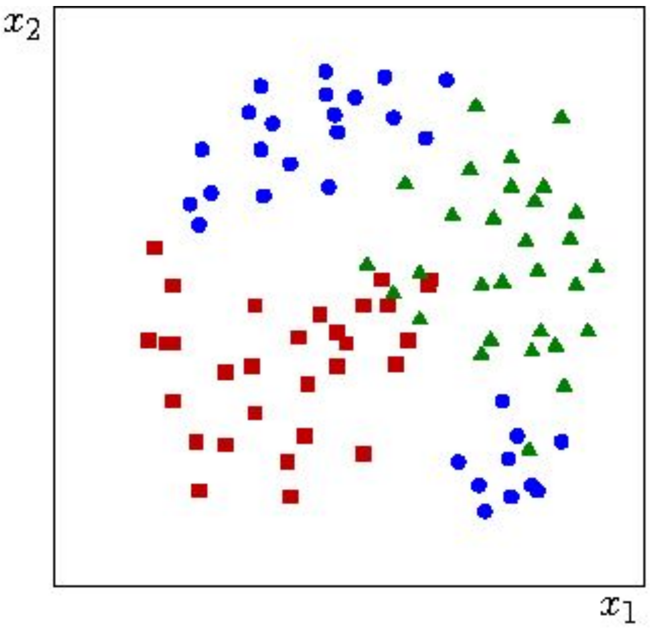


$d = 3$



$d = 5$

Multi-class SVM



Multi-class SVM

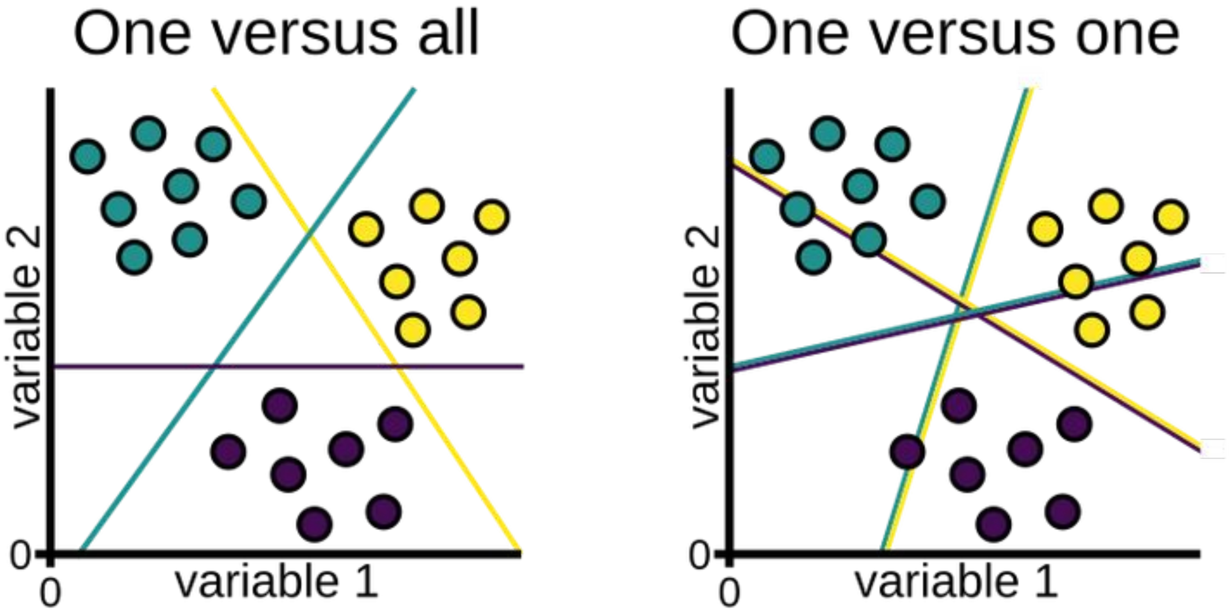


Image Source: www.r-bloggers.com/2019/10/support-vector-machines-with-the-mlr-package/

Application of SVM

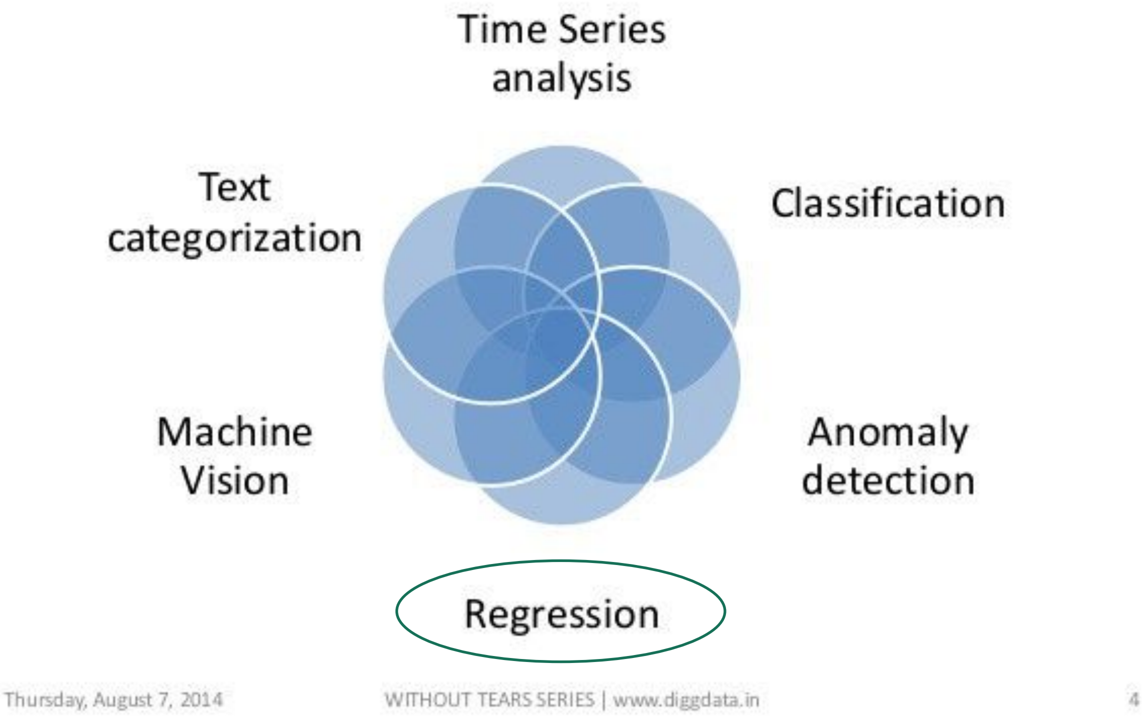


Image Source: towardsdatascience.com/svm-feature-selection-and-kernels-840781cc1a6c

Support Vector Regression (SVR)

- Works by changing the objective function
- We want to fit as many points on the street as possible
- Scikit-learn’s sklearn.svm.SVR class is used for linear and non-linear SVM Regression
- Some of our inputs are kernel type, C, epsilon, and gamma for RBF, Polynomial, and Sigmoid kernels
- SVR does not scale well to large data sets

SVR

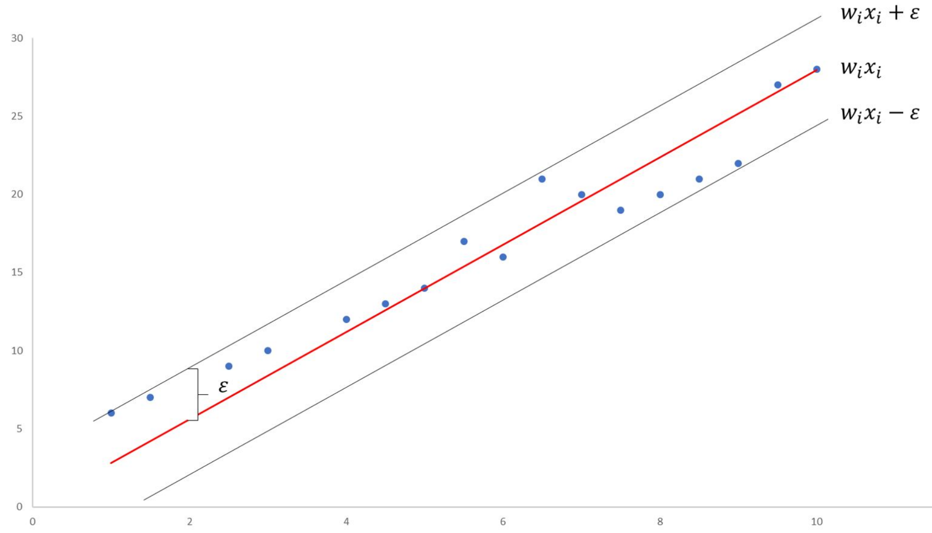


Image Source: towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2

SVR

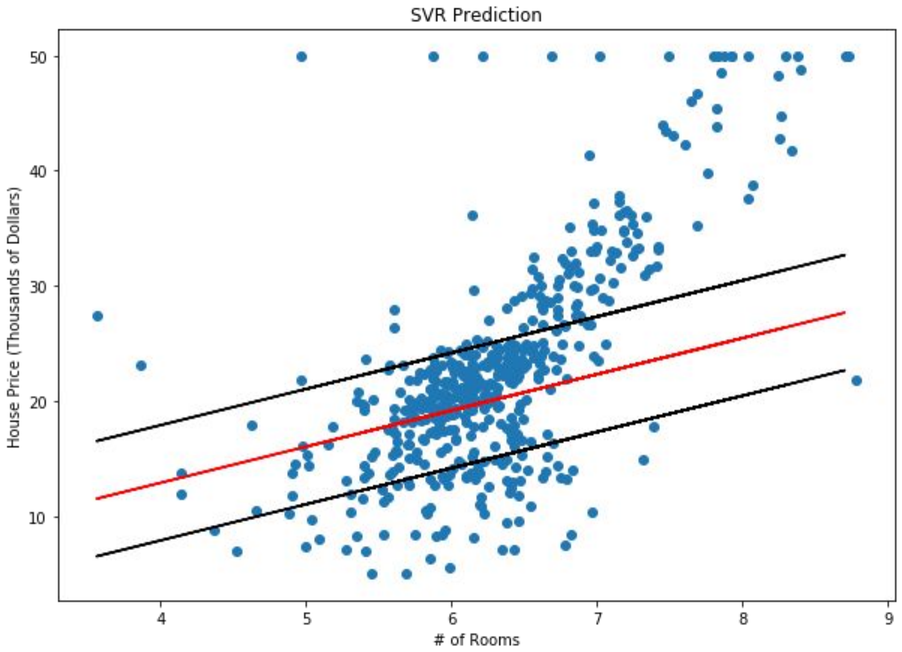


Image Source: towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2

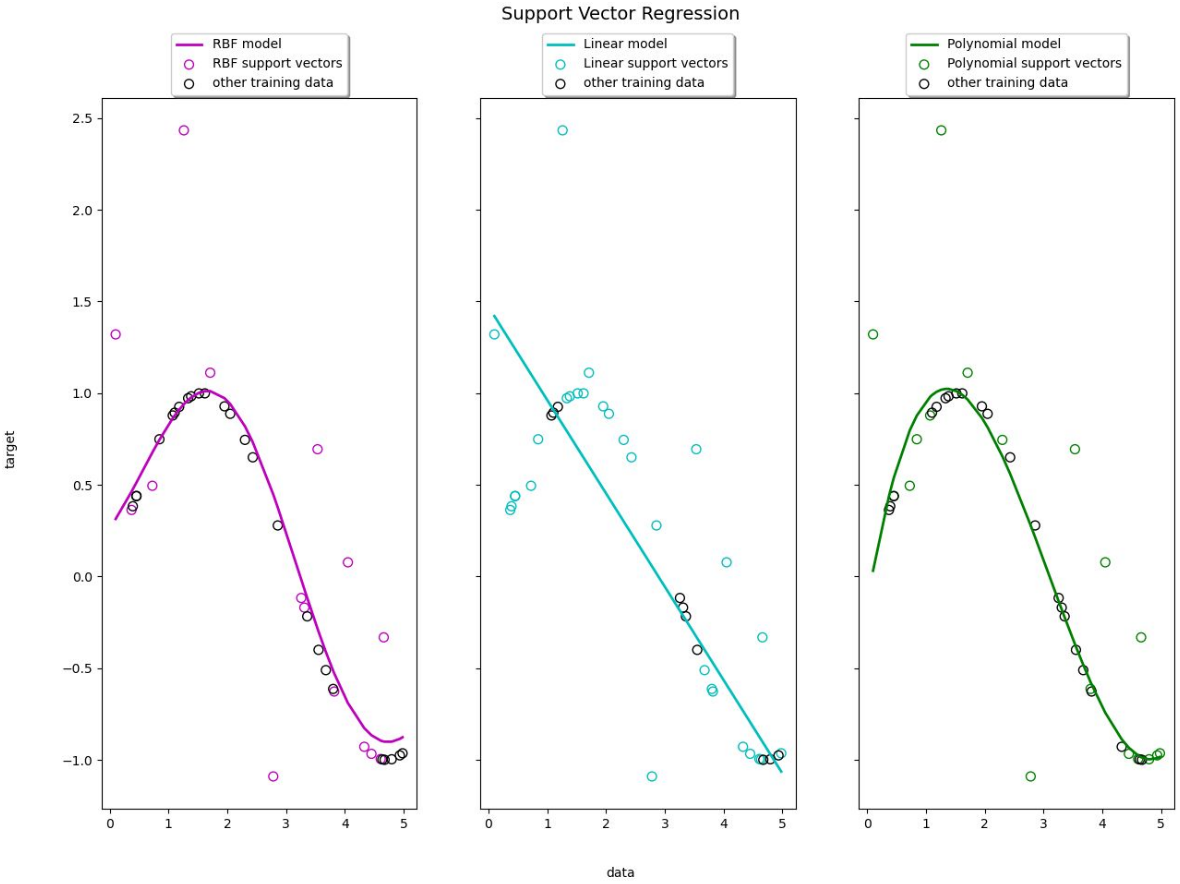


Image Source: scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html

Advantages

- Easy to implement and understand and provides powerful base-line predictions
- Few hyperparameters and therefore hyperparameter tuning
- Memory efficient
- Kernel Tricks
- Versatile in:
    - Types of data
    - Number of dimensions
    - Number of classes

Disadvantages

- Mapping the data to a higher dimension could lead to overfitting the model
- Results are dependent on our choice for the C parameter
- Choosing the correct kernel function for high dimensional data and data regularization are very crucial steps
- Performs poorly on very noisy or very large data sets
- Required balanced data

Trending Applications

- General
  - Facial detection
  - Text and Hypertext Categorization
  - Handwriting classification
- Medical
  - Cancer Classification
  - Protein remote homology detection

Questions?

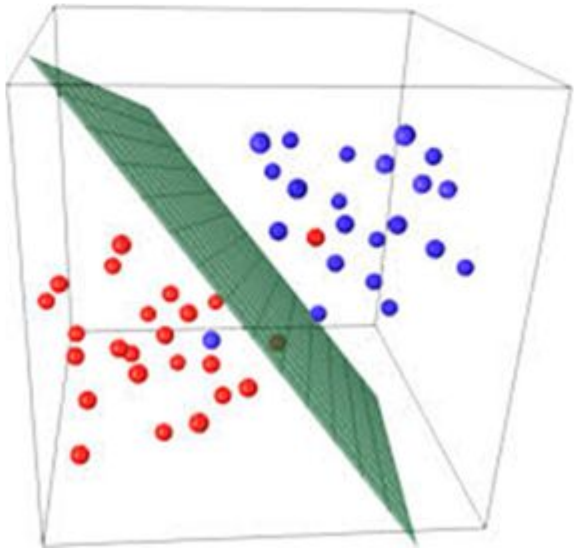
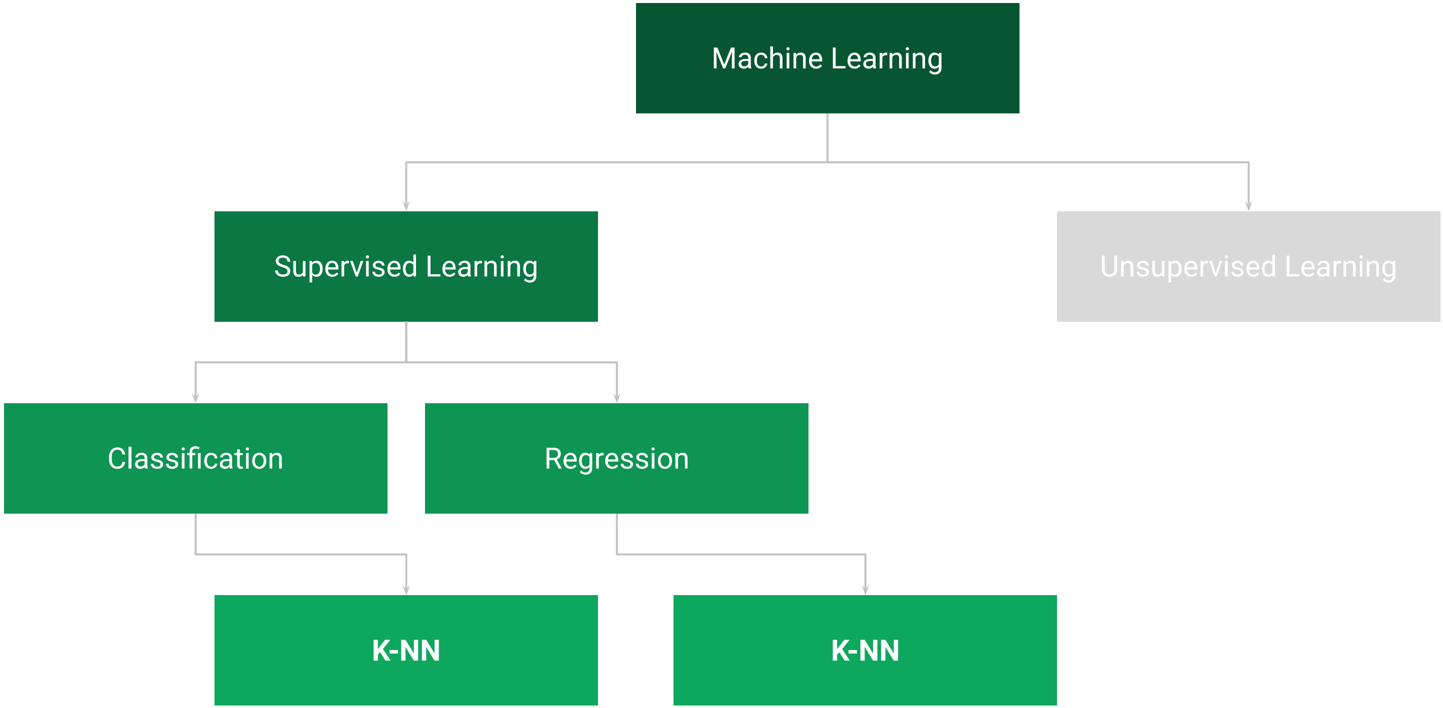


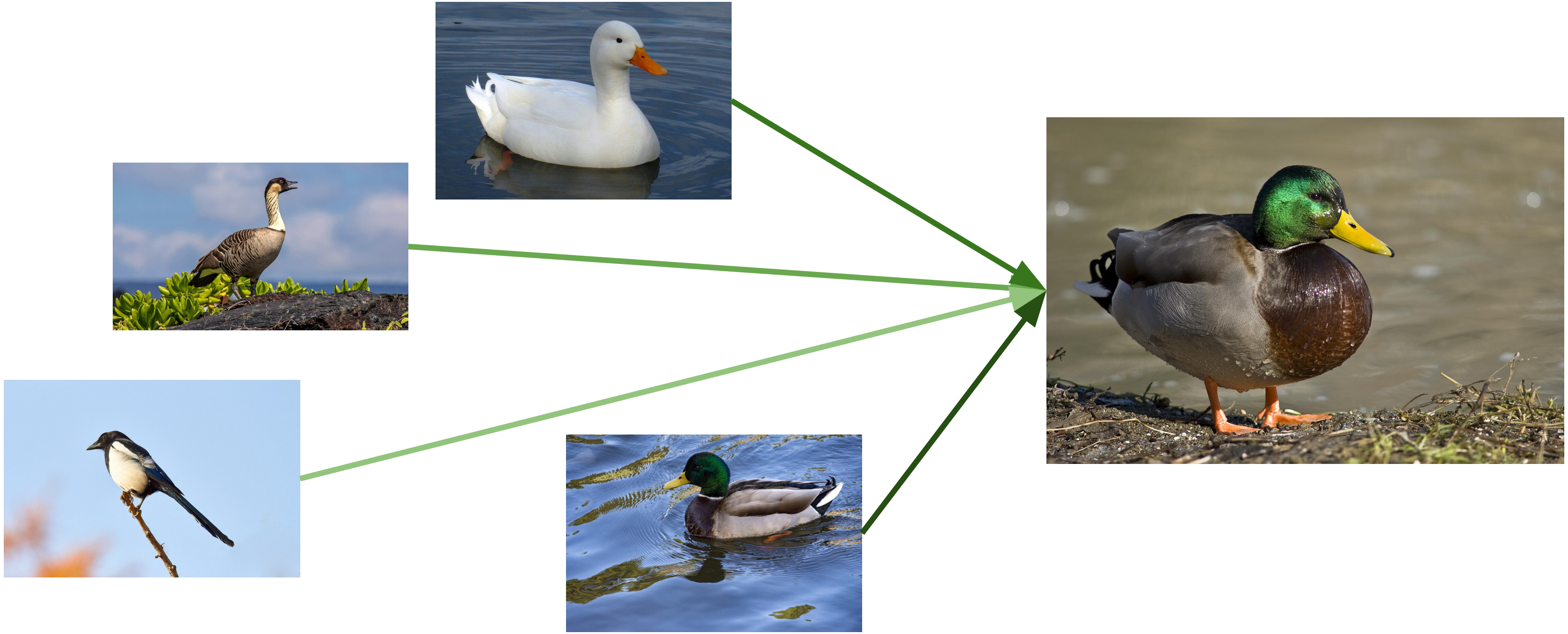
Image Source: appliedmachinelearning.blog/2017/03/09/understanding-support-vector-machines-a-primer/

K-Nearest Neighbors

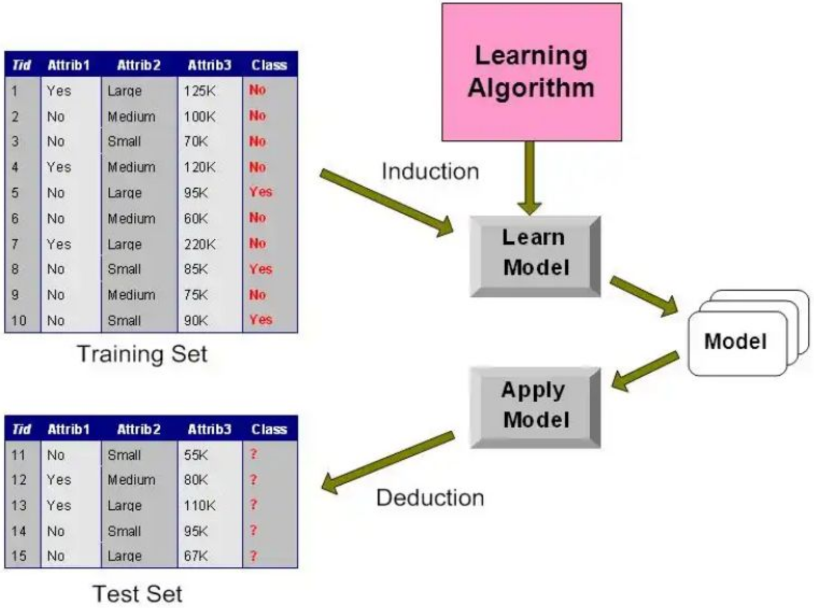

Machine Learning
Supervised Learning
Unsupervised Learning
Classification
Regression
K-NN
K-NN

What is K-Nearest Neighbors?

Basic idea: “If it walks like a duck, quacks like a duck, then it's probably a duck!”



Laziness of K-NN!

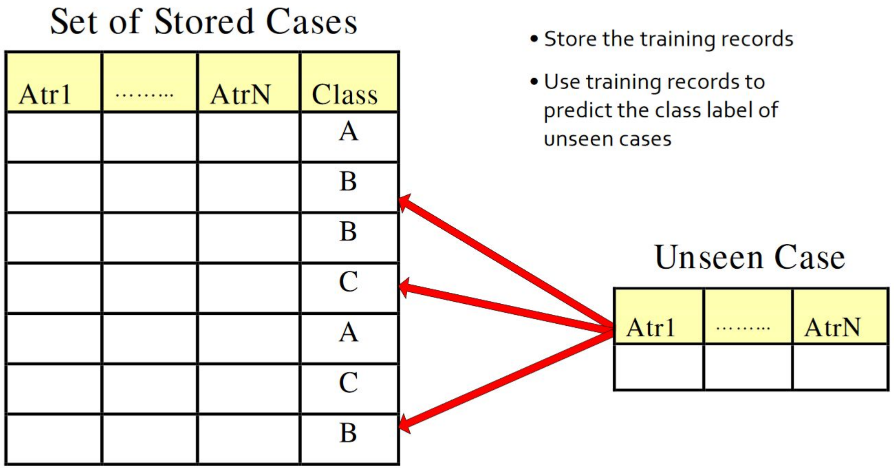


Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

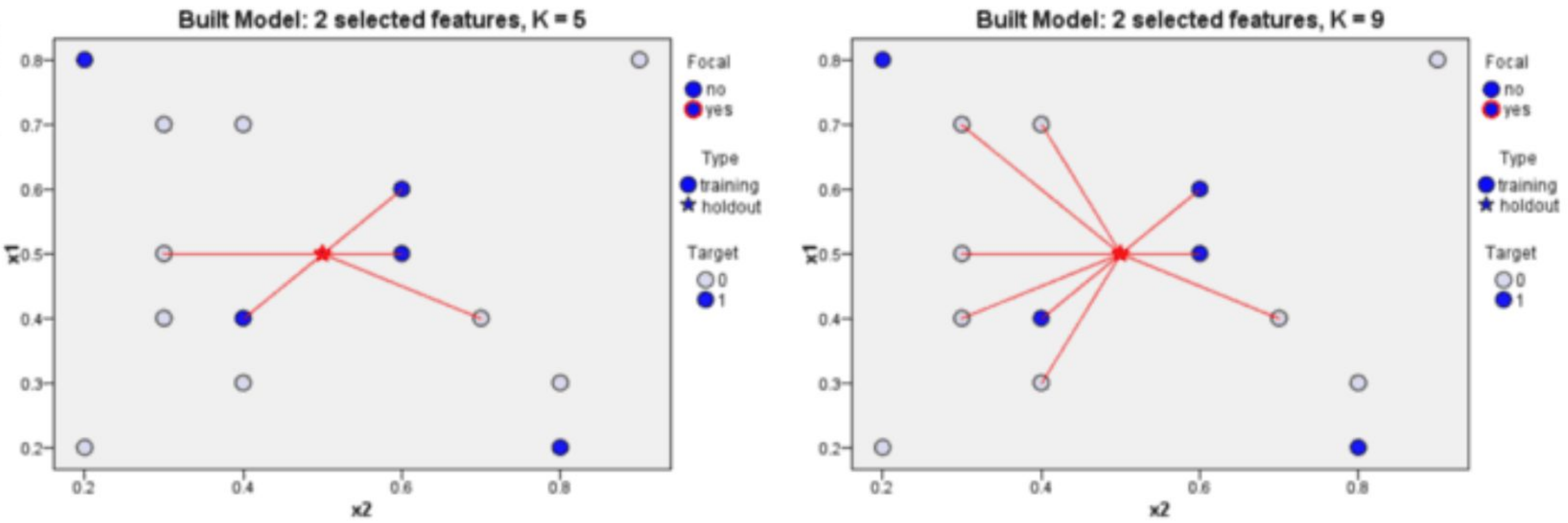


- Store the training records
- Use training records to predict the class label of unseen cases

https://slideplayer-com.cdn.ampproject.org/c/s/slideplayer.com/amp/9533188/

K-Nearest Neighbors

- Classification: k-nearest neighbors common class
- Regression: k-nearest neighbors average value

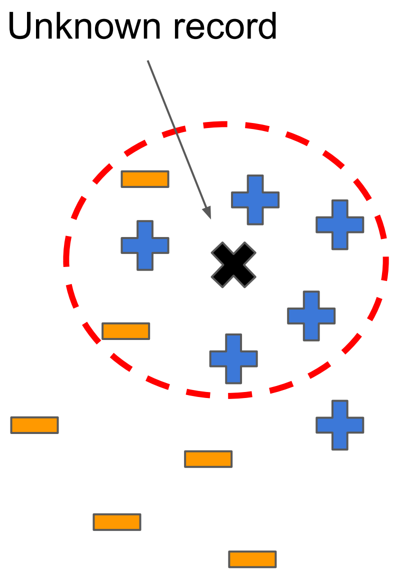


https://miro.medium.com/max/640/0*uqQXfB9PozBiylY1.gif

K-Nearest Neighbors

- K-NN requires:
  - Sample records
  - Distance metric
  - Value of k
  - Weights of neighbors

- To classify a new record:
  - Compute distances
  - Identify k nearest records
  - Use common class or average value of k nearest records



Distance Functions

Various functions could be used, most commonly used in K-NN are:

- Minkowsky (for continuous variables)
- Euclidean (for continuous variables)
- Manhattan (for continuous variables)
- Hamming (for discrete variables)

Chi square distance function: https://springerplus.springeropen.com/articles/10.1186/s40064-016-2941-7

Minkowski, Manhattan and Euclidean Functions

- Used for continuous variables
- Computes distance between two points, A and B in a feature space with k dimensions: $A = (x_1, x_2, \ldots, x_k)$, $B = (y_1, y_2, \ldots, y_k)$

Minkowski $$\left(\sum_{i=1}^{k} \left(|x_i - y_i|\right)^q \right)^{1/q}$$

q=1

Manhattan $$\sum_{i=1}^{k} |x_i - y_i|$$

q=2

Euclidean $$\sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

Simple Example with Euclidean

Customer	Age	Loan	Default
John	25	40000	N
Smith	35	60000	N
Alex	45	80000	N
Jade	20	20000	N
Kate	35	120000	N
Mark	52	18000	N
Anil	23	95000	Y
Pat	40	62000	Y
George	60	100000	Y
Jim	48	220000	Y
Jack	33	150000	Y
Andrew	48	142000	?

We need to predict Andrew default status by using Euclidean distance

https://towardsdatascience.com/a-simple-introduction-to-k-nearest-neighbors-algorithm-b3519ed98e

Simple Example with Euclidean

Customer	Age	Loan	Default	Euclidean distance	Minimum Euclidean Distance
John	25	40000	N	1,02,000.00	
Smith	35	60000	N	82,000.00	
Alex	**45**	**80000**	**N**	**62,000.00**	**5**
Jade	20	20000	N	1,22,000.00	
Kate	**35**	**120000**	**N**	**22,000.00**	**2**
Mark	52	18000	N	1,24,000.00	
Anil	**23**	**95000**	**Y**	**47,000.01**	**4**
Pat	40	62000	Y	80,000.00	
George	**60**	**100000**	**Y**	**42,000.00**	**3**
Jim	48	220000	Y	78,000.00	
Jack	**33**	**150000**	**Y**	**8,000.01**	**1**
Andrew	48	142000	?		

Let assume K = 5

Find minimum euclidean distance and rank in order (ascending)

In this case, 5 minimum euclidean distance. With k=5, there are two Default = N and three Default = Y out of five closest neighbors.

We can say Andrew default stauts is 'Y' (Yes)

https://towardsdatascience.com/a-simple-introduction-to-k-nearest-neighbors-algorithm-b3519ed98e

Hamming Distance Function

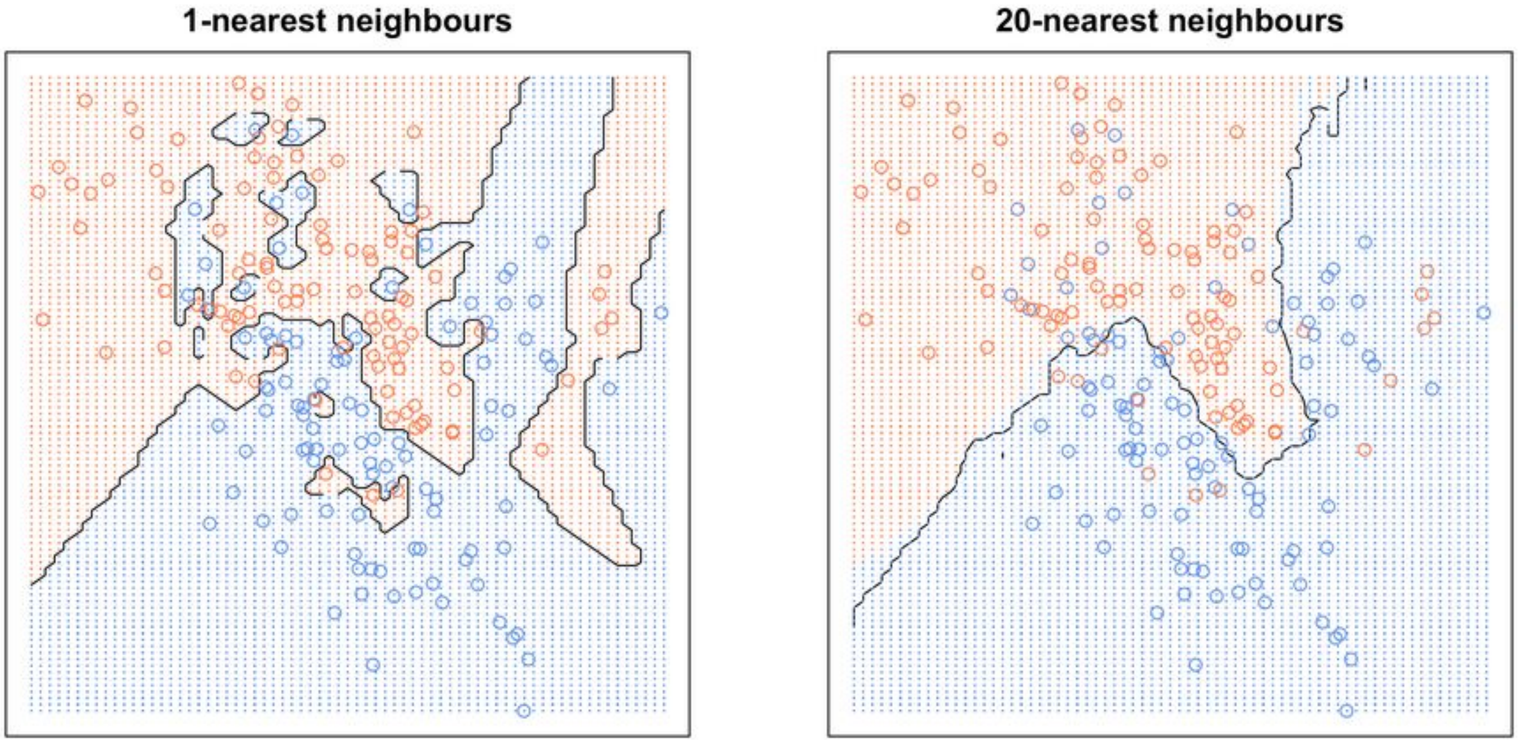
- Used for categorical variables
- Computes the number of instances in which corresponding symbols are different in two strings of equal length
- Number of attributes where x, x' differ

$$D(x,x') = \sum_d 1_{x_d \neq x'_d}$$

Choosing Best K

- No single value of K that will work with every dataset!
- Usually in binary classification, an odd value for K tends to work better than an even value
- Generally, increasing K makes it more precise; tends to smooth out decision boundaries, which will avoid overfitting at the cost of resolution
- We can inspect the data to choose an optimal K value, but usually we perform cross-validation to determine good K value

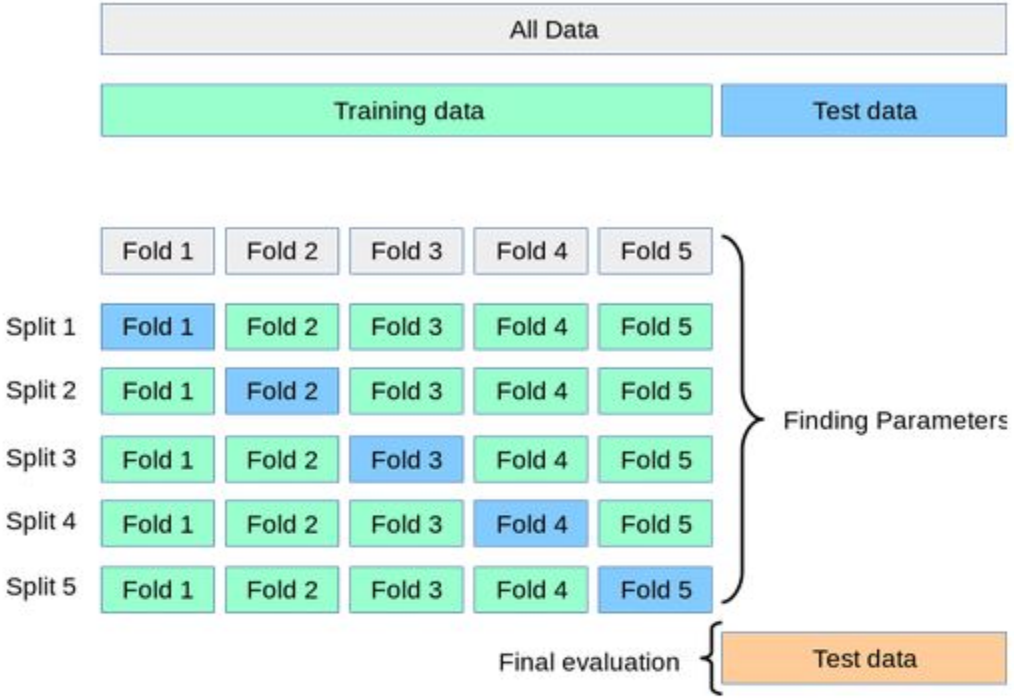
Choosing Best K



https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

Cross-Validation

- Most commonly used is k-fold cross-validation

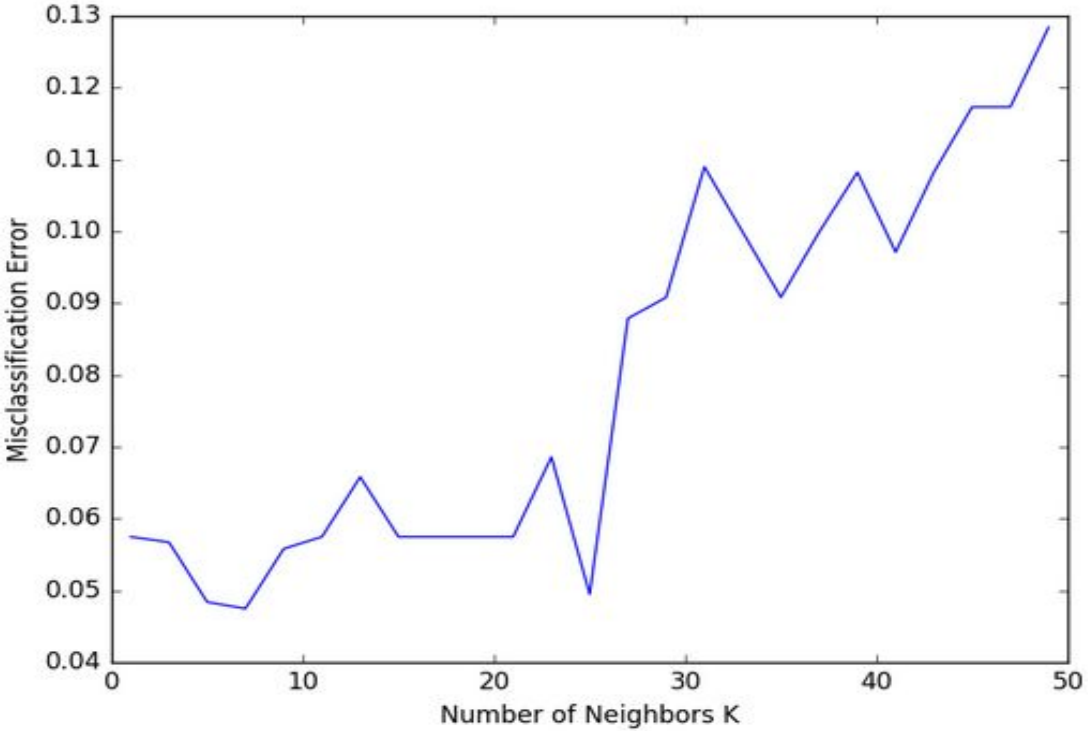


https://scikit-learn.org/stable/modules/cross_validation.html

!!!QUESTION ALERT!!!

In the graph below which is the best value of K to pick for our K-NN algorithm?

1- K = 7
2- K = 25
3- K = 48
4- K = 32

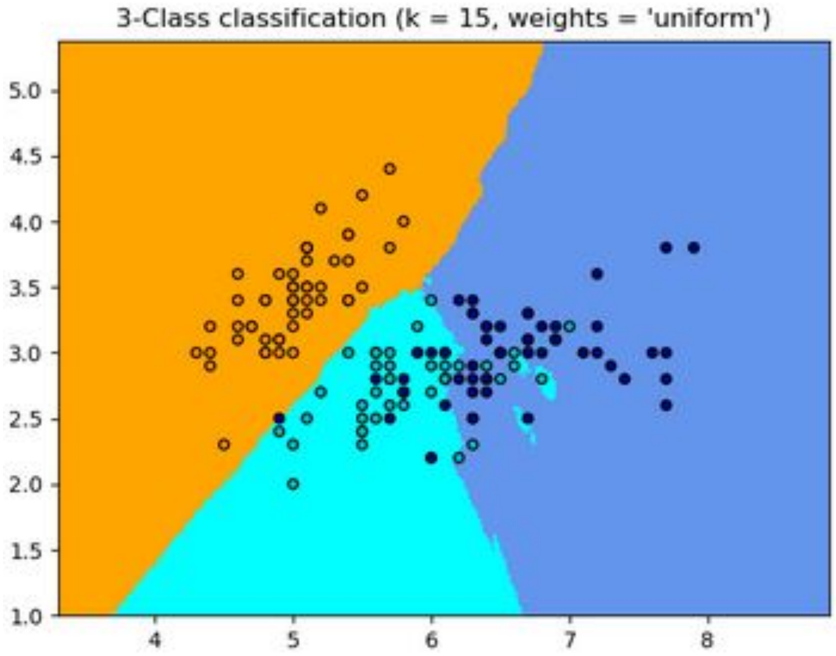


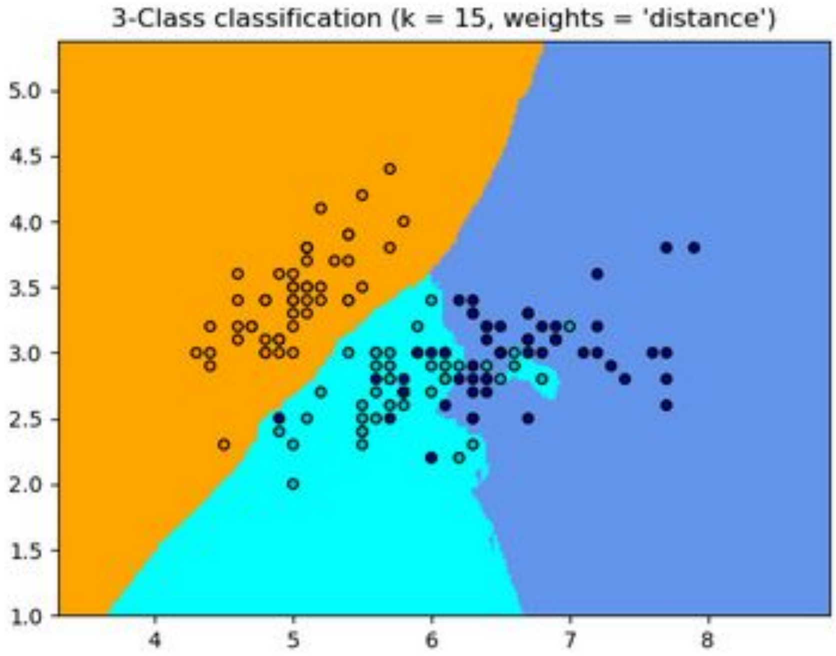
Weights of Neighbors

This can be either:

- **Uniform**: where each point in the neighborhood is weighted equally towards the decision of the classification
- **Distance**: where points are weighted by the inverse of their distance. Closer points will have greater influence than points that are further away

Weighted vs Uniform





https://scikit-learn.org/stable/modules/neighbors.html#classification

Advantages

- No training overhead therefore time efficient
- Very easy to implement as calculations only include the distance between different points
- As there is no training period thus new data can be added at any time since it won't affect the model
- Normalizing the data can dramatically improve the model accuracy
- Handles multi-class cases

Disadvantages

- Very expensive to calculate distances when dataset is large
- Minimal training but expensive testing
- Need to determine the value of K (number of nearest neighbors)
- Doesn’t work well in high dimensional datasets
- Sensitive to noisy and missing data
- Could produce different distance measures from the same training data computed before and after normalization/standardization
- Standardizing features is a drawback

Trending Applications

- Usually in search applications when looking for similar items; i.e. document concept search
- Very popular in recommendation systems
- Used in medical sector, i.e. finding diabetics ratio