AGENDA

1. **Classification Models**
2. **Other Classification Algorithms**
3. **Main performance metrics**

Data Science Academy

I

Classification Models

Data Science Academy

Classification Models

- A classification model tries to draw some conclusion from the input values given for training. It will predict the class labels/categories for the new data
- Predicting what class an input belongs to is extremely useful in many problems such as determining whether someone can be approved for a loan, or if student should be accepted into a competitive university etc.

Binary Classification

- Predicting whether an input belongs to which class in two classes situation is called Binary Classification.
- The problem is phrased like our Hypothesis testing. Imagine we're trying to predict someone's gender based on their height and weight.
- Our Logistic Regression Classifier will look at the input and treat it as if it were responding to our Hypothesis, example "Does this input data belong to a male" or "This is a male sample". The response output of our binary classifier is typically:
  - 0 for False or No
  - 1 for True or Yes

Classification Models

- Classification models aren’t limited to binary classification either, it can in fact be extended to multiple classes e.g.
  - Predicting handwritten digits (Computer Vision Application)
  - Predicting load default, churn and disease
  - Predicting article categories e.g., Sports, Politics, Health etc. (NLP)
  - Or levels of risk involved in an investment.

Numerical vs Classification

- **Numerical**
  - Continuous variables
  - Solves Regression Problems
- **Classification**
  - Categorical variables
  - Solves Classication problems

Popular Classification Algorithms

- Logistic Regression
- Naive Bayes
- K-Nearest Neighbors
- Support Vector Machines
- Decision Trees
- Random Forests
- Neural Networks/Deep Learning

Logistic Regression

- What if wanted to predict a Yes or No type answer, then we can’t use a Linear Regression type model.

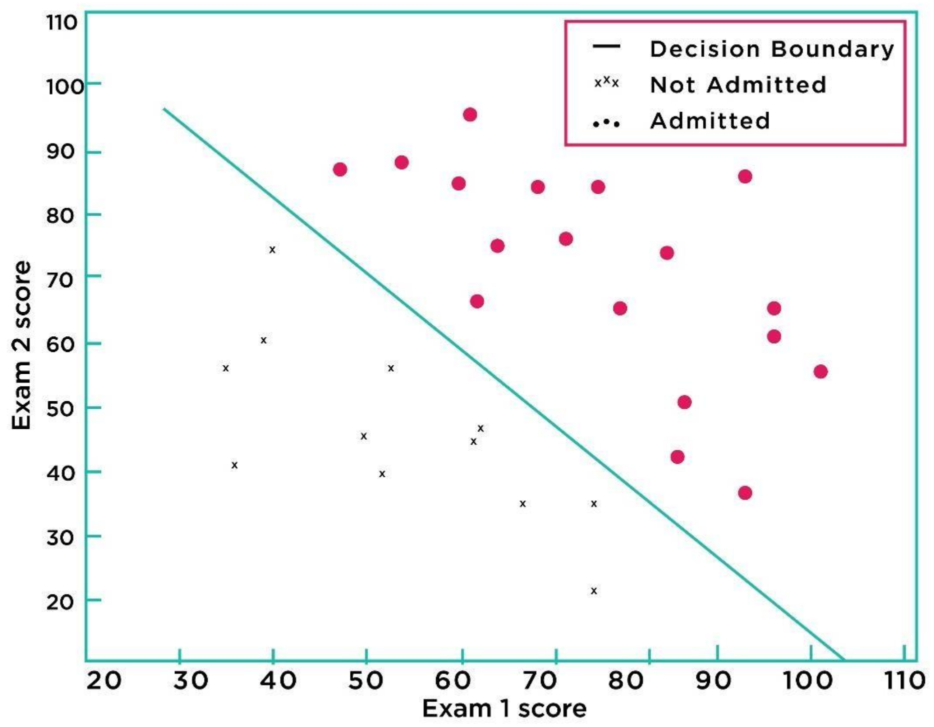
- We need something that takes our inputs and identifies whether something is True or False. Example, predicting whether:

  - a person is Male or Female
  - an email is Spam or not
  - a person is likely to say Yes or No

Logistic Regressions Create Decision Boundaries

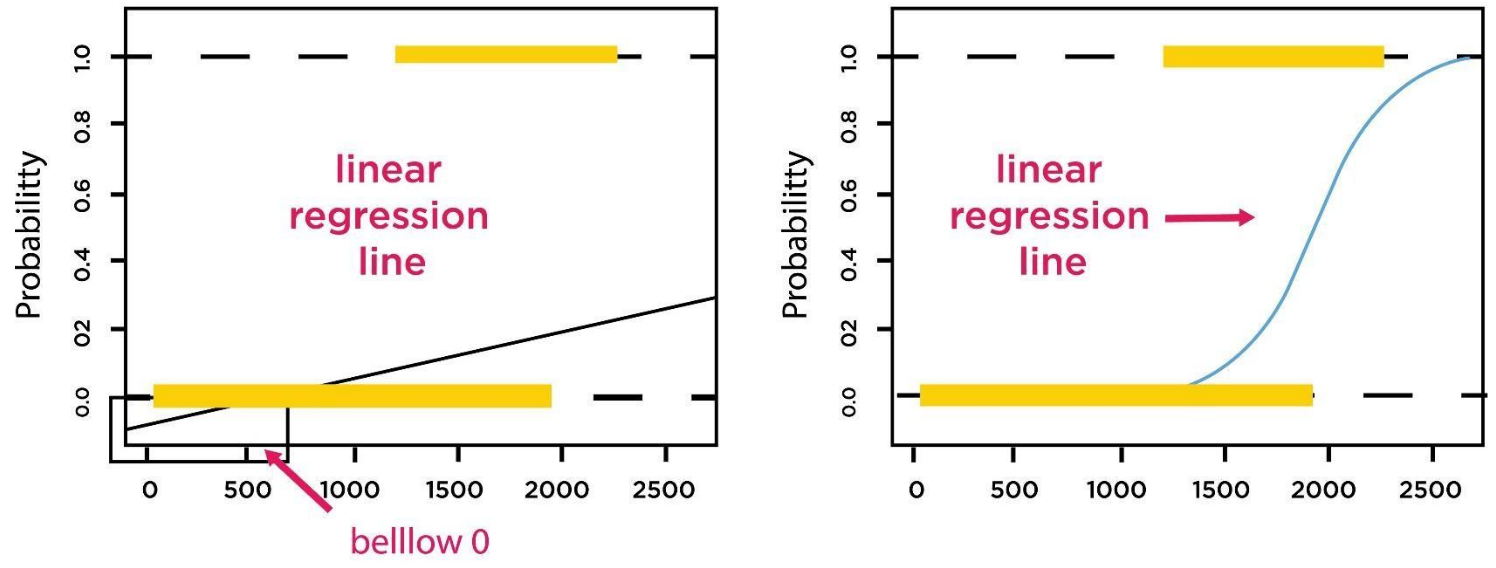
- We can visually interpret the Logistic Regressio threshold by plotting its Decision Boundary.

- This decision boundary line is given by solving the linear term for values that evaluate to 0.5



Logistic Regression

- In a large number of classification problems, the targets are designed to be binary.
- Instead, we can transform our linear regression to a logistic regression curve.



Modelling Logistic Regression

In [ ]:

```
# Importing Logistic regression library:
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
Regressor = LogisticRegression()
# Fittining splitted variables:
regressor.fit(X_train, y_train)
# Predicting X_test:
y_pred = regressor.predict(X_test)
# Calculating accuracy score:
accuracy_score(y_test, y_pred)*100
```

Naive Bayes

- Naive Bayes is the most simple classification algorithm based on Bayes' Theorem for applying to data.
- Algorithm makes an assumption as all the variables in the dataset is "Naive" i.e not correlated to each other.

$$P(A/B) = \frac{P(B/A) \; * P(A)}{P(B)}$$

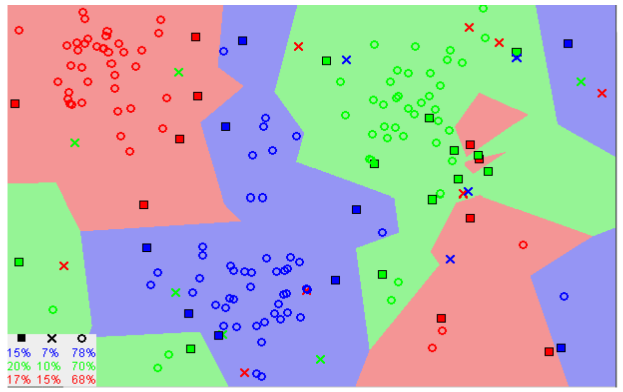
What Naive Bayes is used for?

- Text classification
- Spam Filtering
- Sentiment Analysis
- Recommendation System
- and etc...

K-Nearest Neighbors (KNNs)

- KNNs are one of the most popular machine learning algorithms as it gives somewhat decent performance and is also quite easy to understand and implement.

- It’s called a lazy learner (as opposed to eager learners) because it doesn’t have a training phase, what it does instead is compute its classification using the training data (can be slow). This also makes it a non-parametric algorithm.

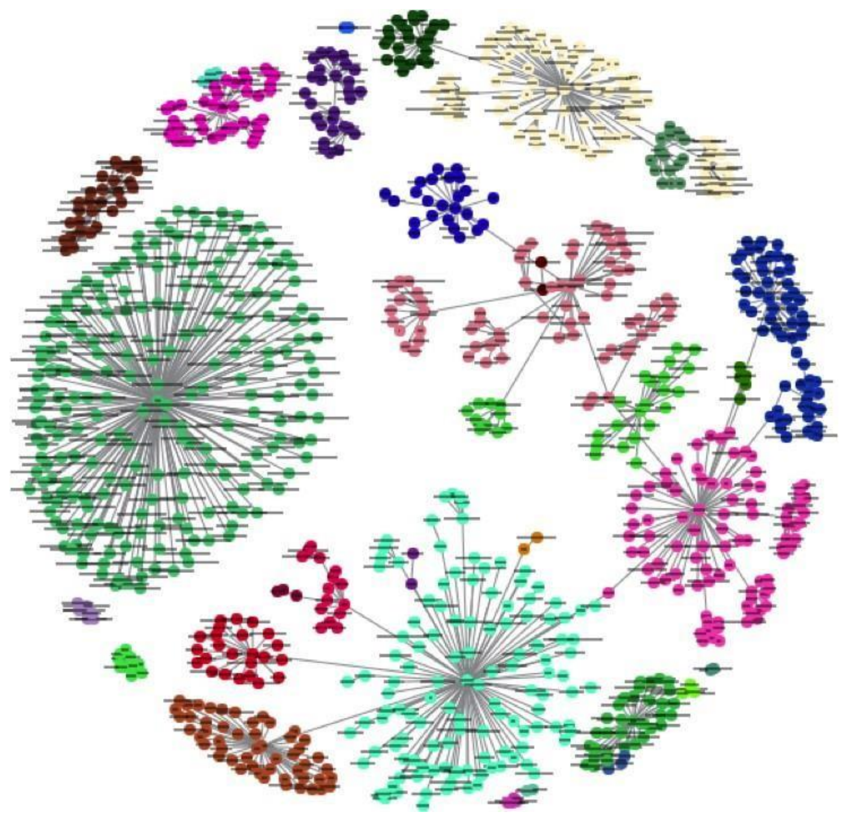


Where to use KNN

- KNN is often used in simple recommendation systems, image recognition technology, and decision-making models.

- Netflix or Amazon use KNN in order to recommend different movies to watch or books to buy.

  - Note: Netflix even launched the Netflix Prize competition, awarding $1 million to the team that created the most accurate recommendation algorithm.



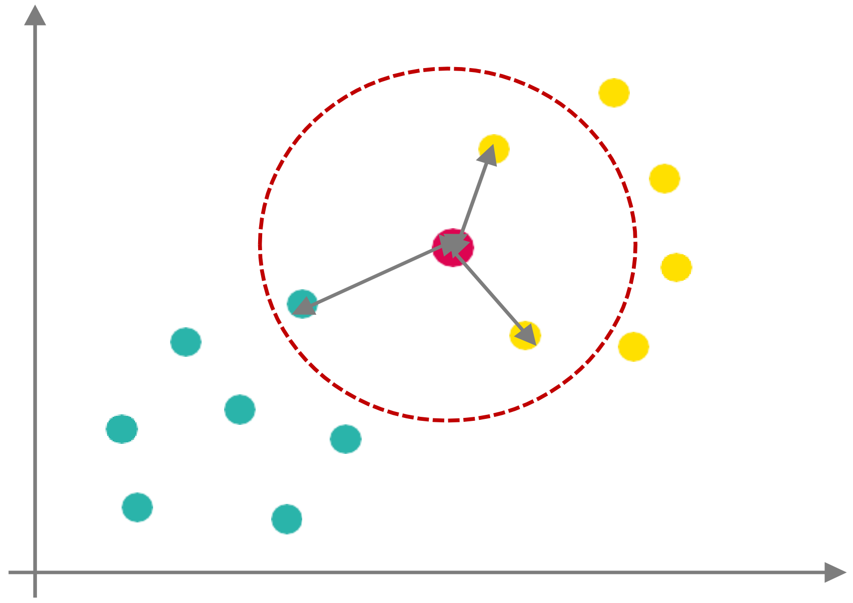
The KNN Algorithm – Step by Step

How KNNs classify a new data input:

- Load all training data
- Use a pre-set value of k (integer only)
- Calculate the distance between the test data input and each value of the training data
- Sort on our distances (ascending i.e. smallest distance to largest distance)
- Choose the top K rows of our sorted data
- Assign class to our input data to the most frequent class in our K rows

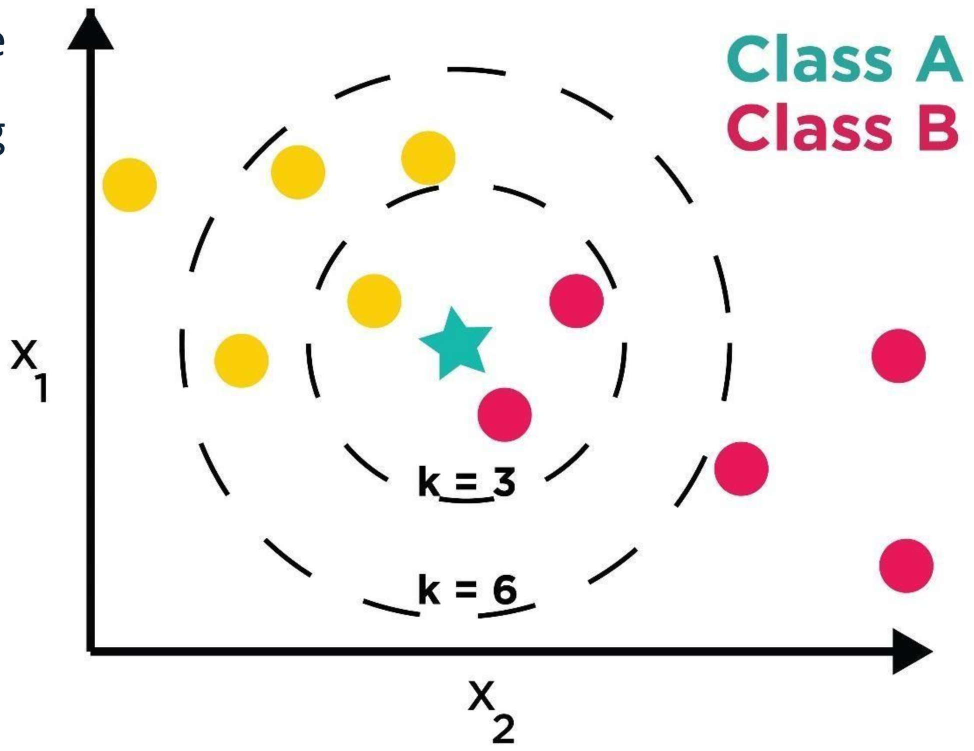
The KNN Algorithm – Using k = 3

- As K = 3, let's draw a circle enclosing the 3 nearest points to our input (red circle)
- Two out our 3 points were yellow when sorted by distance, therefore the class KNN will assign the red point to will be the yellow class



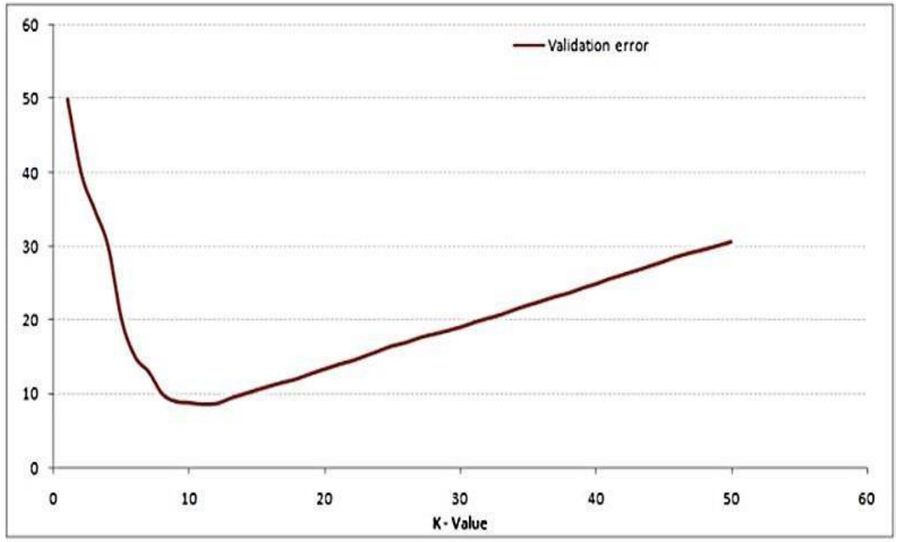
Choosing K

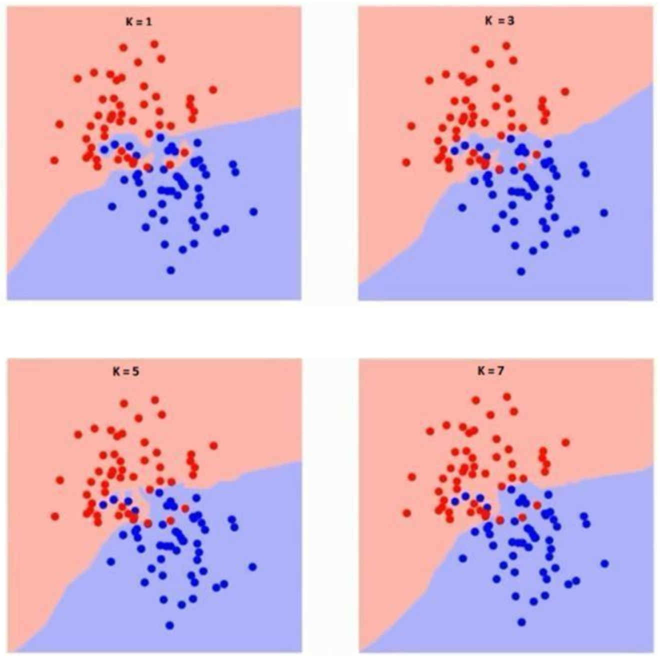
- As we can see from our last diagram, the larger k, the more points it considers. But how do we go about knowing which k to choose?
- If k = 1, we will only assign classes considering the closest point, this can lead to a model that overfits
- However, as k goes up, we begin to simplify the model too much thus leading to a model that is under-fitting (high bias and low variance)



Validation error

- Notice how our decision boundary becomes smoother as k increases (less overfitting) but only up to a point!





Pros & Cons of K-Nearest Neighbors

- Very simple
- Training is trivial
- Works with any number of classes
- Easy to add more data
- Few parameters
  - K
  - Distance Metric

- High Prediction Cost (worse for large data sets)
- Not good with high dimensional data
- Categorical Features don’t work well

Maths behind KNN

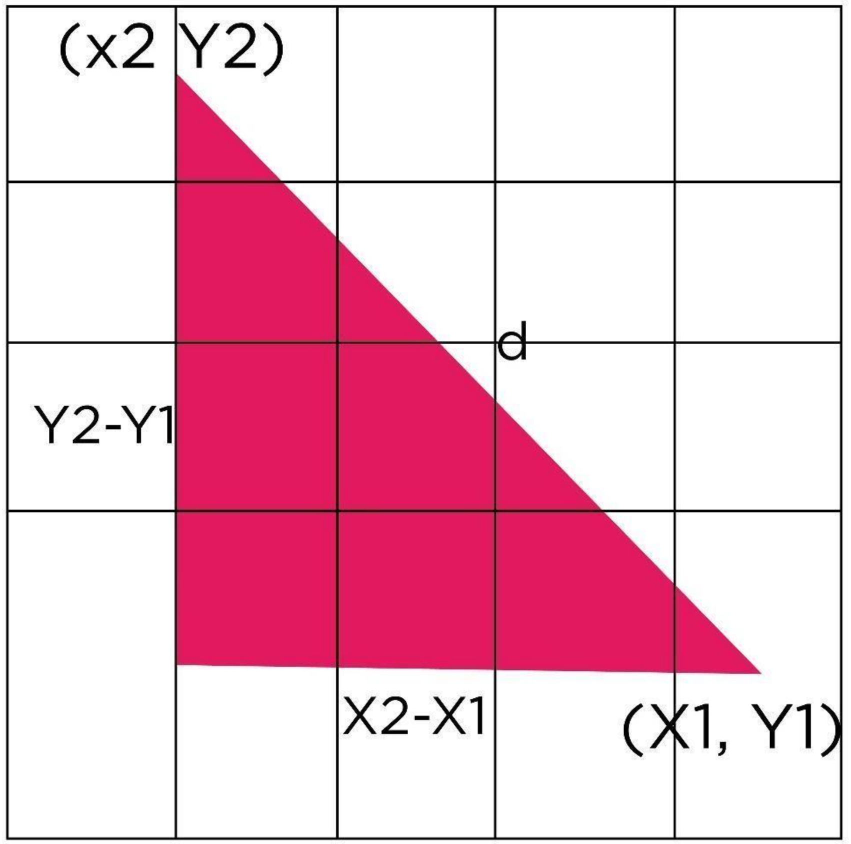
- KNN works because of the deeply rooted mathematical theories it uses.
- First step is to transform data points into feature vectors, or their mathematical value.
- Second step is algorithm works by finding the distance between the mathematical values of these points.
- The most common way to find this distance is the Euclidean distance, as shown below.

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2} =$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Maths behind KNN

- KNN runs this formula to compute the distance between each data point and the test data.
- It then finds the probability of these points being similar to the test data and classifies it based on which points share the highest probabilities.
- To visualize this formula, it would look something like this



II

Other Classification Algorithms

Data Science Academy

What is SVM?

- SVM offers very high accuracy compared to other classifiers such as logistic regression, and decision trees.
- It is known for its kernel trick to handle nonlinear input spaces.
- It is use in a variety of applications such as face detection, intrusion detection, classification of emails, news articles and web pages, classification of genes, and handwriting recognition.

What is SVM?

- Support Vector Machine (SVM) is a supervised machine learning algorithm capable of performing classification, regression and even outlier detection. However, it is mostly used in classification problems.
- The linear SVM classifier works by drawing a straight line between two classes. All the data points that fall on one side of the line will be labeled as one class and all the points that fall on the other side will be labeled at the second.

How it works?

- In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.
- Then, we perform classification by finding the hyper-plane (or decision boundary) that differentiates the two classes very well

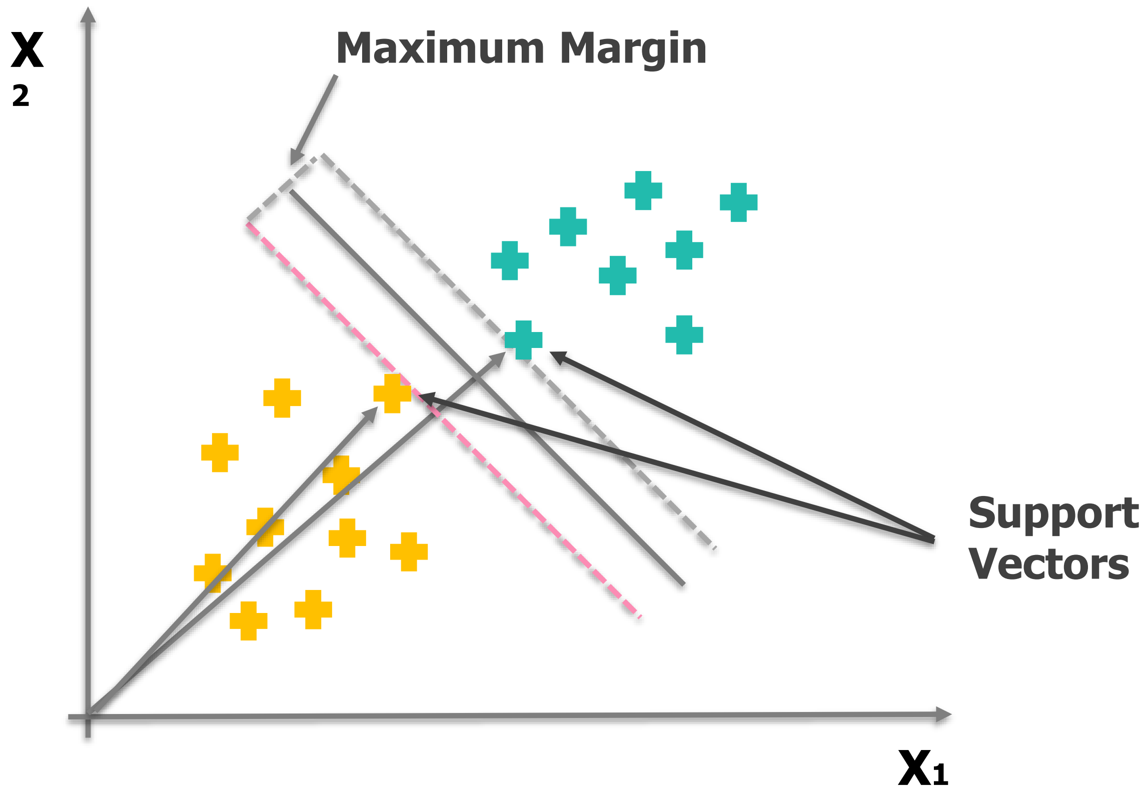
SVM – Support Vectors

- To find the which line (in 2-dimensional) will do the best job of classifying the data we SVM algorithm.
- The SVM algorithm will select a line that not only separates the two classes but stays as far away from the closest samples as possible.
- The “support vector” in “support vector machine” refers to two position vectors drawn from the origin to the points which dictate the decision boundary.

Support Vectors

Support vectors are the data points, which are closest to the hyperplane.

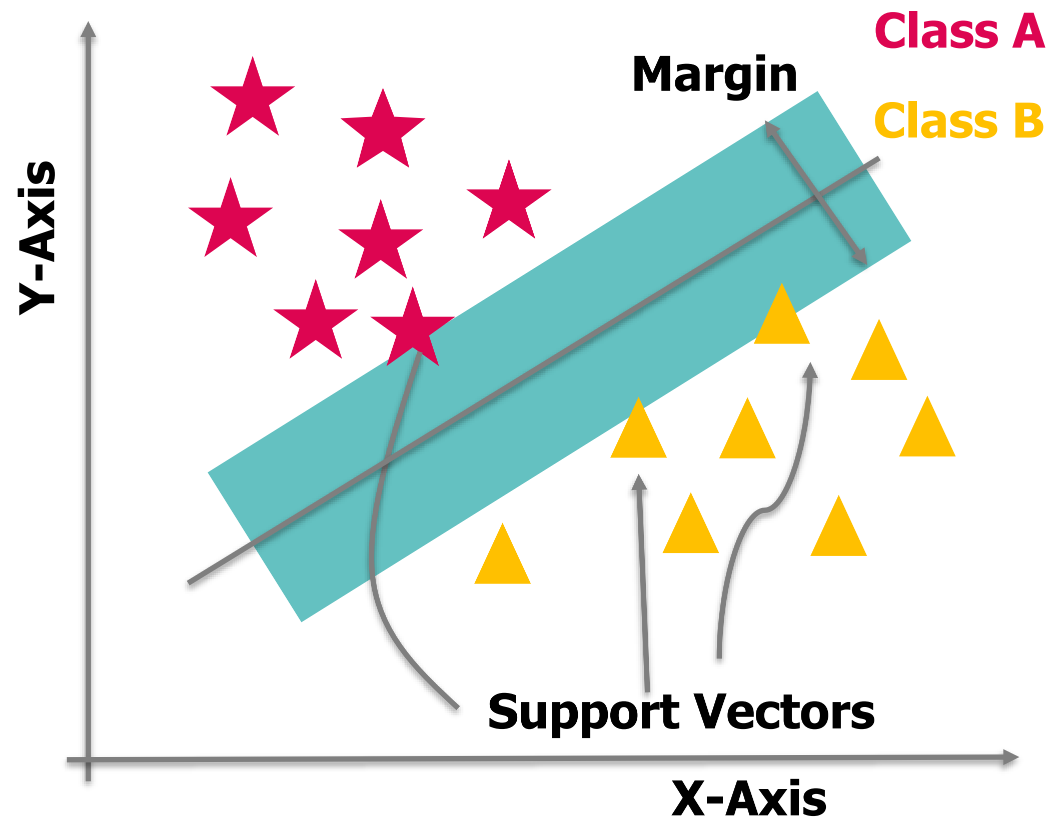
- These points will define the separating line better by calculating margins.
- These points are more relevant to the construction of the classifier



Margin

- A margin is a gap between the two lines on the closest class points.
- This is calculated as the perpendicular distance from the line to support vectors or closest points.
- If the margin is larger in between the classes, then it is considered a good margin, a smaller margin is a bad margin.

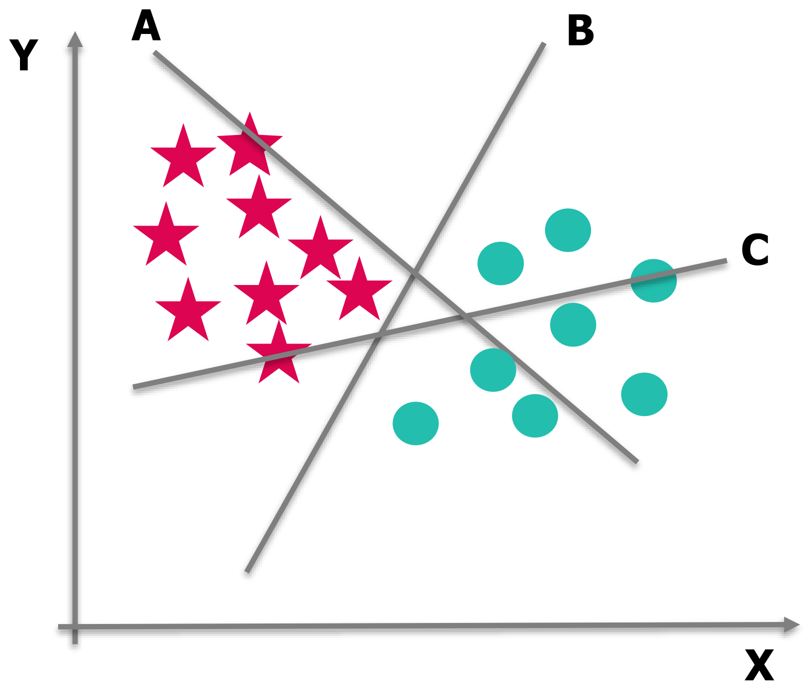
Maximum Marginal Hyperplane



- The core idea of SVM is to find a maximum marginal hyperplane (MMH) that best divides the dataset into classes.
- SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error.

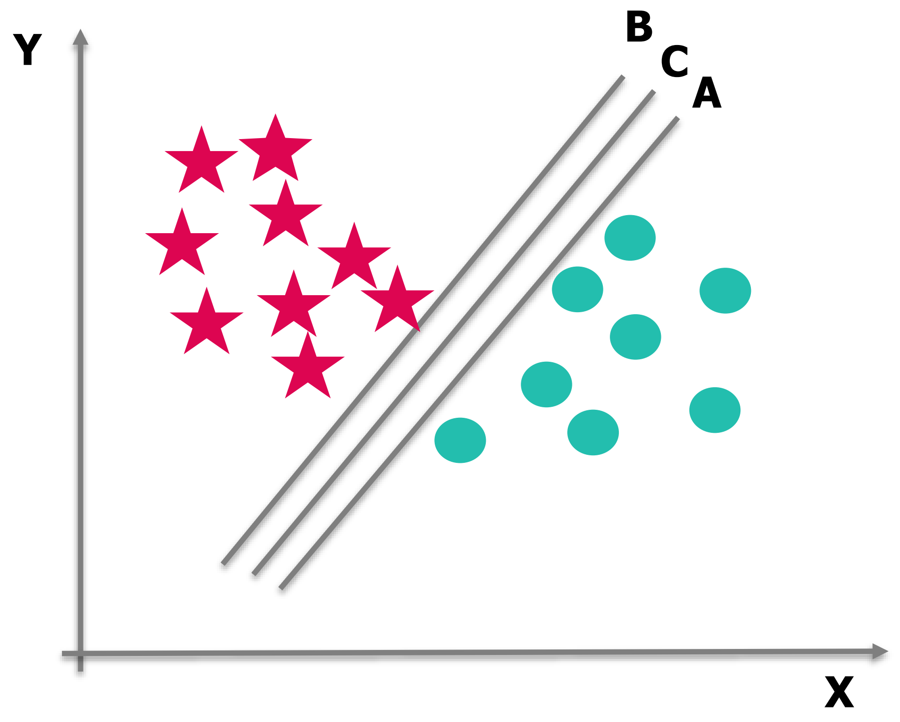
Identify the right hyperplane

- Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.
- Select the hyper-plane which segregates the two classes better. In this scenario "B" has excellently performed this job.



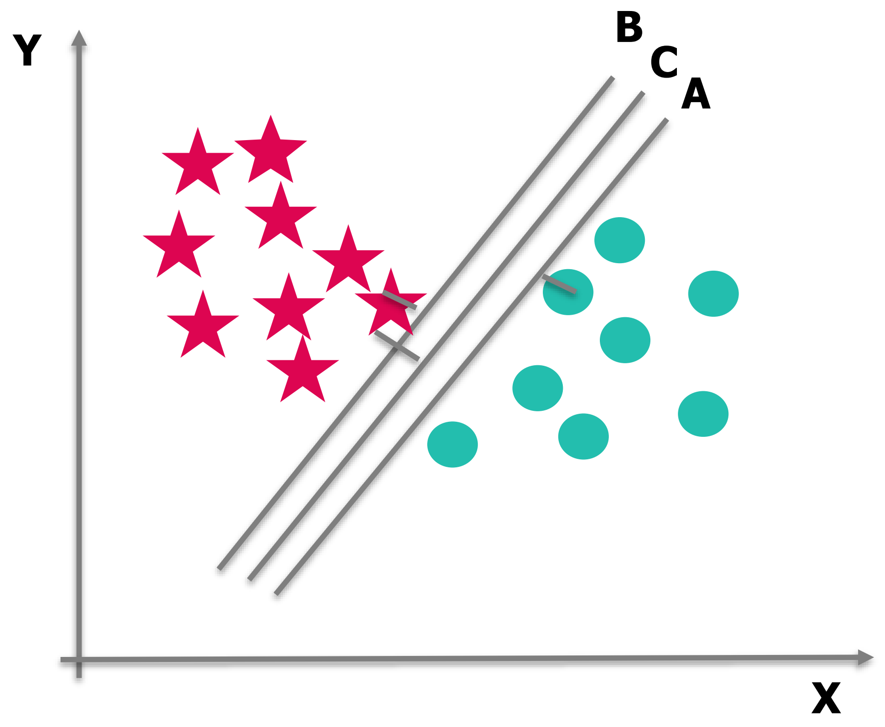
Identify the right hyperplane (Scenario 1)

- We have three hyper-planes (A, B and C) and all are segregating the classes well.
- Maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin.



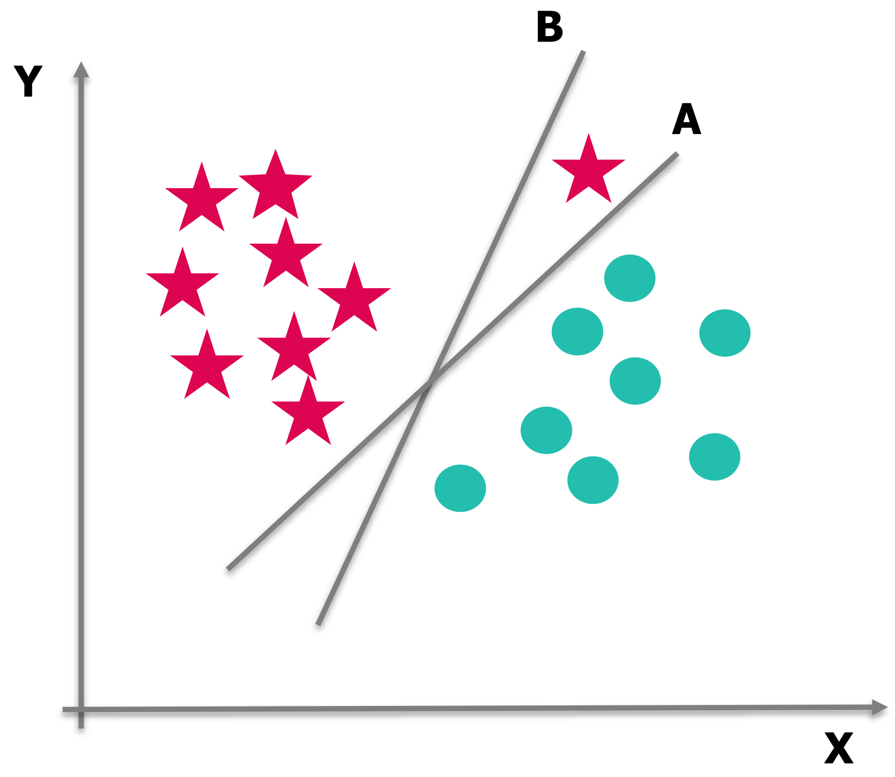
Identify the right hyperplane (Scenario 1)

- You can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C.
- Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.



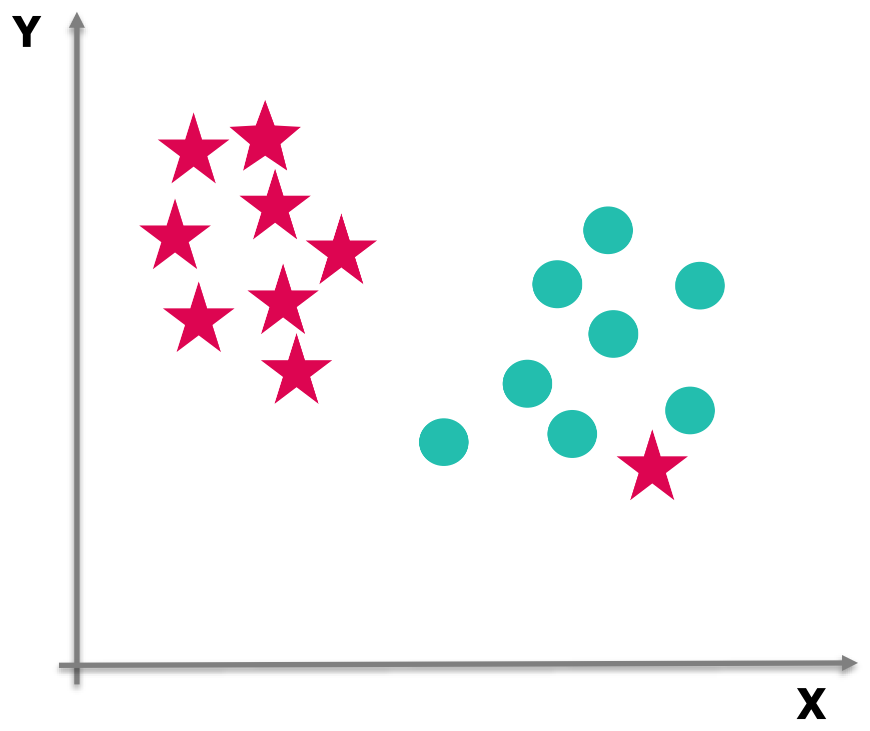
Identify the right hyperplane (Scenario 2)

- Some of you may have selected the hyper-plane B as it has higher margin compared to A.
- But SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin.
- Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is A.



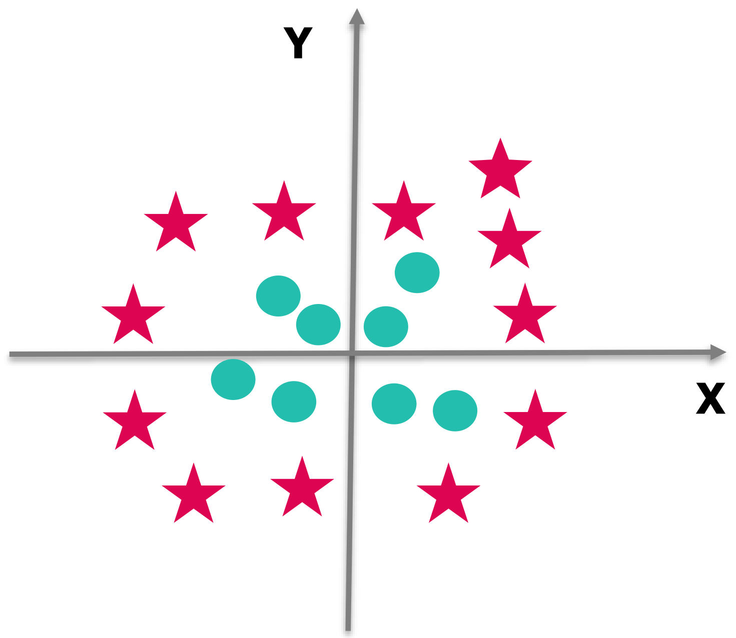
Identify the right hyperplane Outlier

- Over start at other end is like an outlier for star class.
- The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin.
- Hence, we can say, SVM classification is robust to outliers.

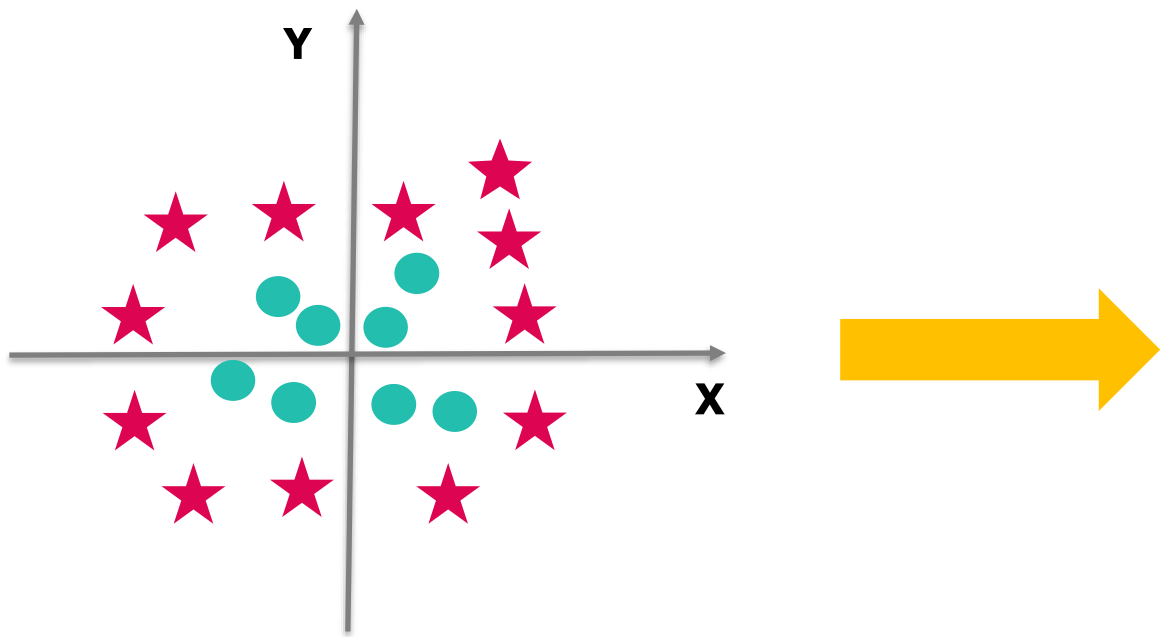


Identify the right hyperplane (Scenario 3)

- In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes?
  - Till now, we have only looked at the linear hyper-plane.
  - SVM can solve this problem. Easily!
  - It solves this problem by introducing additional feature. Here, we will add a new feature $z=x^2+y^2$.
  - Now, let's plot the data points on axis x and z



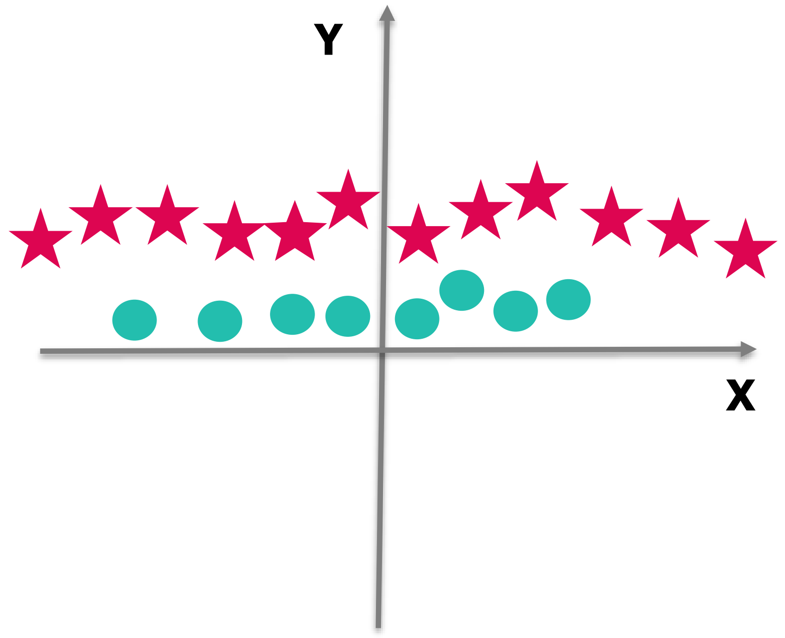
From 2-D to 3-D



From 2-D to 3-D

- All values for z would be positive always because z is the squared sum of both x and y.

- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher of z.



SVM Kernel and Kernel Trick

- The SVM algorithm is implemented in practice using a kernel.
- A kernel transforms an input data space into the required form.
- Should we need to add this feature manually to have a hyper-plane? No, the SVM algorithm has a technique called the kernel trick.
- The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space.
- It converts “not separable” problem to “separable problem” by adding more dimension to it.

Types of Kernel (Linear Kernel)

- A linear kernel can be used as normal dot product of two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.

$$K(x, xi) = sum(x * xi)$$

- Where, x and xi are any two vectors

Types of Kernel (Polynomial Kernel)

- A polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel can distinguish curved or nonlinear input space.

$$K(x,xi) = 1 + sum(x * xi)^d$$

- Where d is the degree of the polynomial.
- d=1 is similar to the linear transformation.
- The degree needs to be manually specified in the learning algorithm.

Types of Kernel (Radial Basis Function Kernel)

- The Radial basis function kernel is a popular kernel function commonly used in support vector machine classification. RBF can map an input space in infinite dimensional space.

K(x,xi) = exp(-gamma * sum((x – xi^2))

- Here gamma is a parameter, which ranges from 0 to 1.
- A higher value of gamma will perfectly fit the training dataset, which causes over-fitting.
- Gamma=0.1 is considered to be a good default value.
- The value of gamma needs to be manually specified in the learning algorithm.

SVM Implementation in Python

- We will use SVM class from Sklearn library:

In [ ]:

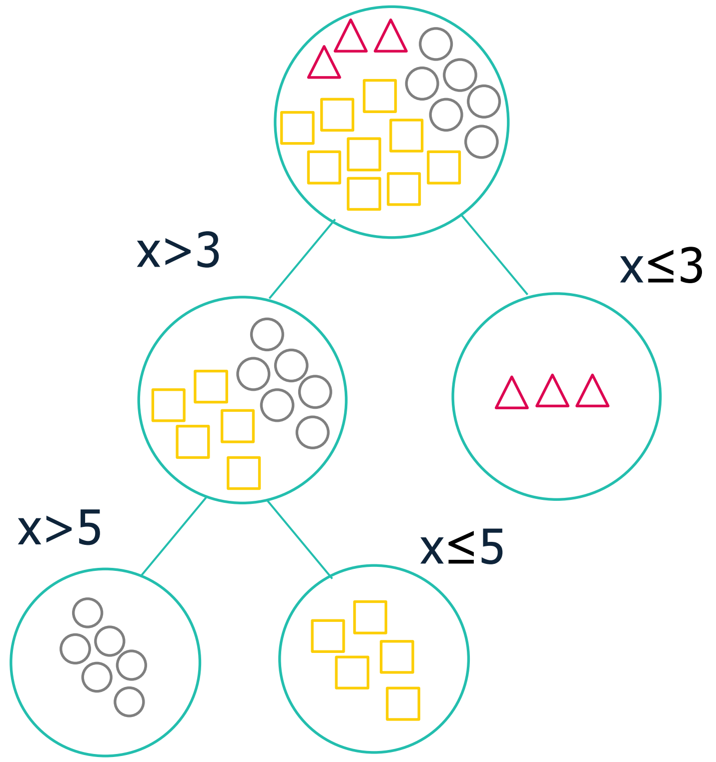
```
from sklearn import svm
#Create a svm Classifier
clf = svm.SVC(kernel= 'linear') #Linear Kernel
clf1 = svm.SVC(kernel= 'poly', degree=2) #“Poly” kernel
clf2 = svm.SVC(kernel= 'rbf', C=1, gamma=a).fit(X,y) #“RBF” kernel
#Train the model using the training sets
clf.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

Decision Tree

- Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving regression and classification problems too.
- The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from prior data(training data).
- The understanding level of Decision Trees algorithm is so easy compared with other classification algorithms. The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label.

Gini Index for Multiple Classes

- Calculating Gini Index is how we construct our Decision Tree.
- Decision trees are consisting of several key building blocks:
  - Root Nodes
  - Decision Nodes (Intermediate nodes)
  - Terminal Nodes (Leaves)
- We only stop when our Gini Gain (i.e. G impurity minus the impurity of our branches is zero). This is because in our last leaves there won't be any more classes to separate.



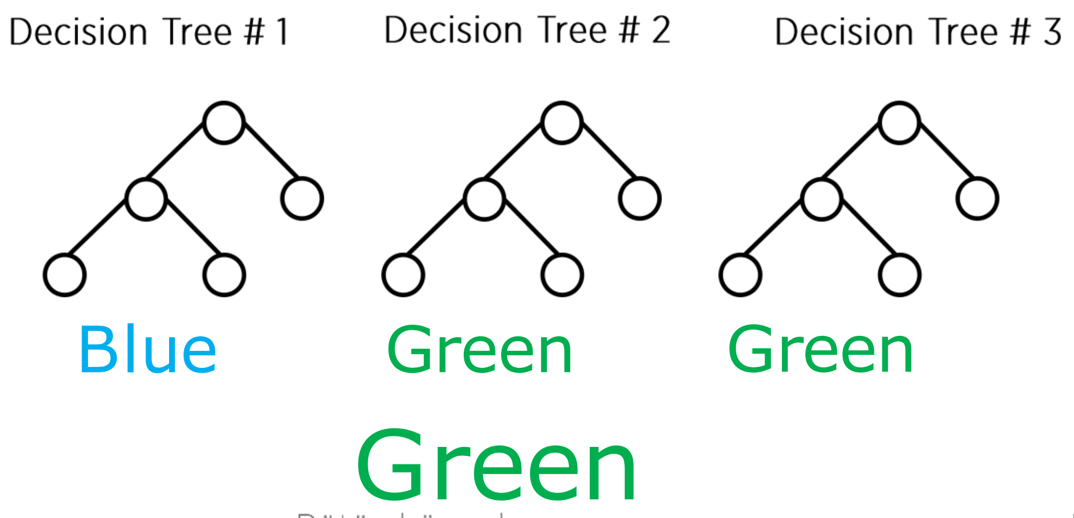
Pros & Cons of Decision Tree

- Decision trees require less effort for data preparation during pre-processing. A decision tree does not require normalization of data.
- A decision tree does not require scaling of data.
- Missing values in the data also does NOT affect the process of building decision tree to any considerable extent.
- A Decision trees model is very intuitive and easy to explain to technical teams as well as stakeholders.

- A small change in the data can cause a large change in the structure of the decision tree causing instability.
- For a Decision tree sometimes calculation can go far more complex compared to other algorithms.
- Decision tree often involves higher time to train the model.
- Decision tree training is relatively expensive as complexity and time taken is more.
- Decision Tree algorithm is inadequate for applying regression and predicting continuous values.

Random Forests

- Once we understand Decision Trees the intuition for Random Forests makes sense.
- Random forests are simply a the output of multiple decision tree classifiers.
- For Random Forests we sample with replacement from our training dataset, then train our decision tree on n set of samples and repeat this t times. Note some samples will be used multiple times in a single tree (hence the random part of Random forests)
- This process of using multiple classifiers and finding the most common output (majority vote) or average (if regression) is called Bagging or Boostrapping.



Random Forests

- The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is:

- A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

Random Forests

- The low correlation between models is the key.
- Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions.
- The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction).
- While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction.

III

Main performance metrics: Accuracy, Precision, Recall, ROC & AUC Curve, Gini Coefficient

Data Science Academy

Main Performance Metrics

- **Accuracy rate -** is the number of correctly predicted data points out of all the data points
- **Confusion matrix -** is a table that is often used to describe the performance of a classification model on a set of test data
- **ROC curve** - plots the performance of a binary classifier under various threshold settings
- **AUC** - is the area enclosed by the ROC curve
- **GINI coefficient** - its purpose is to normalize the AUC

Assessing Model Accuracy

- Accuracy is simply a measure of how much of our training data did our model classify correctly

$$\text{Accuracy} = \frac{\text{Correct Classification}}{\text{Total number of classification}}$$

- Our model should try to minimize these False Negatives. For these cases, we don't use just only accuracy but also F1-Score

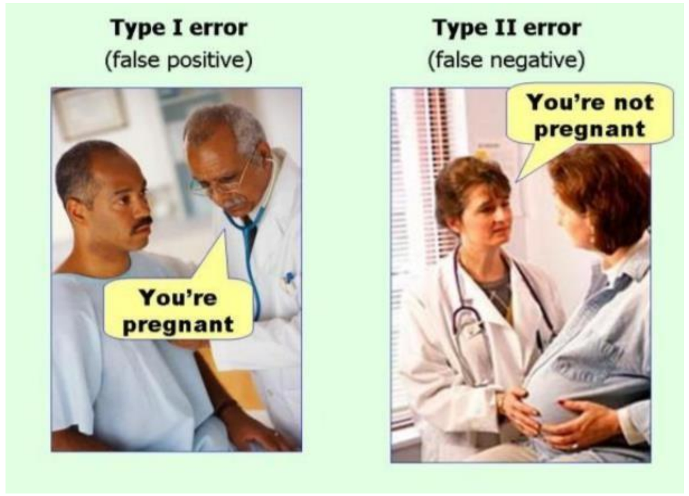
Is Accuracy the only way to assess a model’s performance?

- While very important, accuracy alone doesn’t tell us the whole story.
- Imagine we’re using a Model to predict whether a person has a life-threatening disease based on a blood test.
- There are now 4 possible scenarios.
  - TRUE POSITIVE- Test Predicts Positive for the disease and the person has the disease
  - TRUE NEGATIVE - Test Predicts Negative for the disease and the person does NOT have the disease
  - FALSE POSITIVE - Test Predicts Positive for the disease but the person does NOT have the disease (Type I error)
  - FALSE NEGATIVE - Test Predicts Negative for the disease but the person actually has the disease (Type II Error)

Confusion matrix

Based on the way classifier has performed, four more metrics are derived:

- True Positive (TP)
- False Positive (FP)
- True Negative (TN)
- False Negative (FN)



		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 $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$

Metrics of Confusion Matrix

- **Accuracy** - Use precision when it is needed to know preciseness of predictions between each other not train dataset labels
- **Precision** - Use precision when it is needed to know preciseness of predictions between each other not train dataset labels
- **Recall (Sensitivity) -** Use recall when it is needed to know how model has performed relative to the train dataset labels
- **Specificity -** Use in cases where classification of false negatives is a priority.
- **F1-Score** - Use F1-Score when distribution of class labels is uneven or in other words FP and FN have different weights for overall evaluation of performance

Metrics of Confusion Matrix

- Recall(Sensitivity) – How much of the positive classes we got correct
  - Recall = $\dfrac{\text{True Positive}}{\text{True Positive + False Negative}}$
- Specificity – How much of the negative classes we got correct
  - Specificity = $\dfrac{\text{True Negative}}{\text{True Negative + False Positive}}$
- Precision – When predicting positive, how many of our positive predictions were right?
  - Precision = $\dfrac{\text{True Positive}}{\text{True Positive + False Positive}}$
- F1-Score – Is a metric that attempts to measure both Recall & Precision
  - F1-Score = $\dfrac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall + Precision}}$

Accuracy or F1-Score?

To summarize the differences between the F1-score and the accuracy,

- Accuracy is used when the True Positives and True negatives are more important while F1-score is used when the False Negatives and False Positives are crucial
- Accuracy can be used when the class distribution is similar while F1-score is a better metric when there are imbalanced classes as in the above case.
- In most real-life classification problems, imbalanced class distribution exists and thus F1-score is a better metric to evaluate our model on

F1-Score and Support

- F1 Score is the weighted average of Precision and Recall. As a result, this score takes both false positives and false negatives into account It is useful if you have an uneven class distribution as accuracy works best if false positives and false negatives have similar cost.
- The f1-scores corresponding to every class will tell you the accuracy of the classifier in classifying the data points in that particular class compared to all other classes
- The support is the number of samples of the true response that lie in that class.

More on Recall vs. Precision

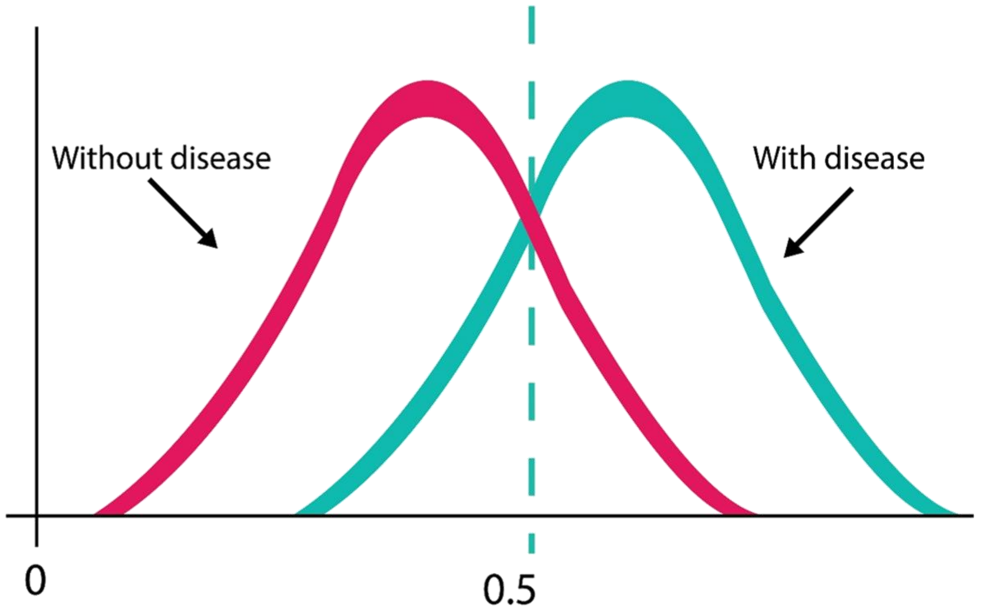
- High recall (or sensitivity) with low precision.
  - This tells us that most of the positive examples are correctly recognized (low False Negatives) but there are a lot of false positives i.e. other classes being predicted as our class inquestion.

- Low recall (or sensitivity) with high precision.
  - Our classifier is missing a lot of positive examples (high FN) but those we predict as positive are indeed positive (low False Positives).

ROC (Receiver Operating Characteristic)

- The ROC curve is very important as it tells us how good our model is at distinguishing between two classes.
- Imagine we have a model that predicts whether someone has a disease or not.
- In the diagram on the right we use 0.5 as our threshold to identify someone as having the disease.
- However, look at the area around 0.5, there can be a lot of ambiguity if a sample point lies in that region.

Distribution of probabilities predicted by our model for people with and without the disease



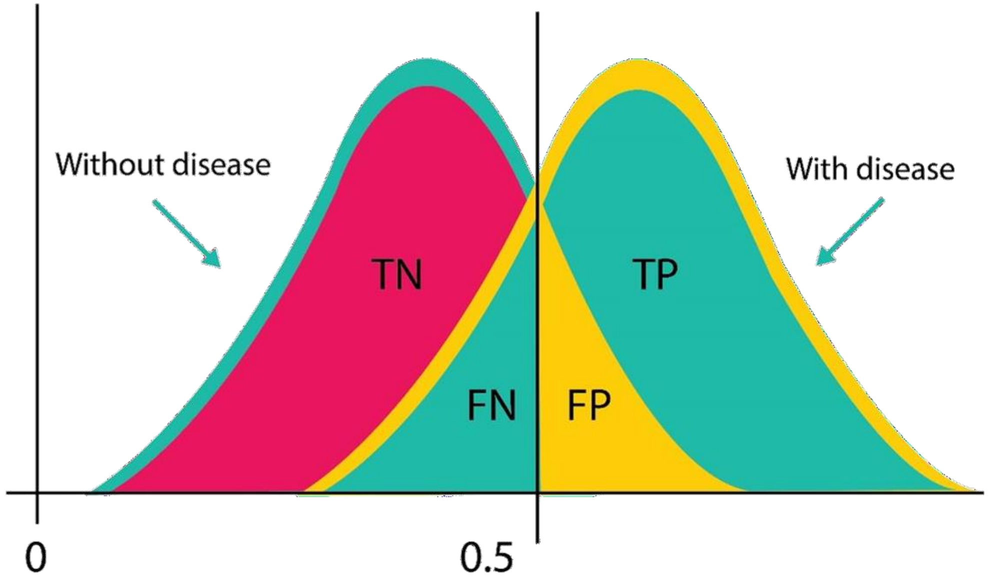
Remember our Recall and Precision

- Recall- in this example, Recall would be the number of patients who our model identified as having the disease (TP) over the total number of patients that actually have the disease.

  - $\text{Recall} = \dfrac{TP}{TP+FN}$

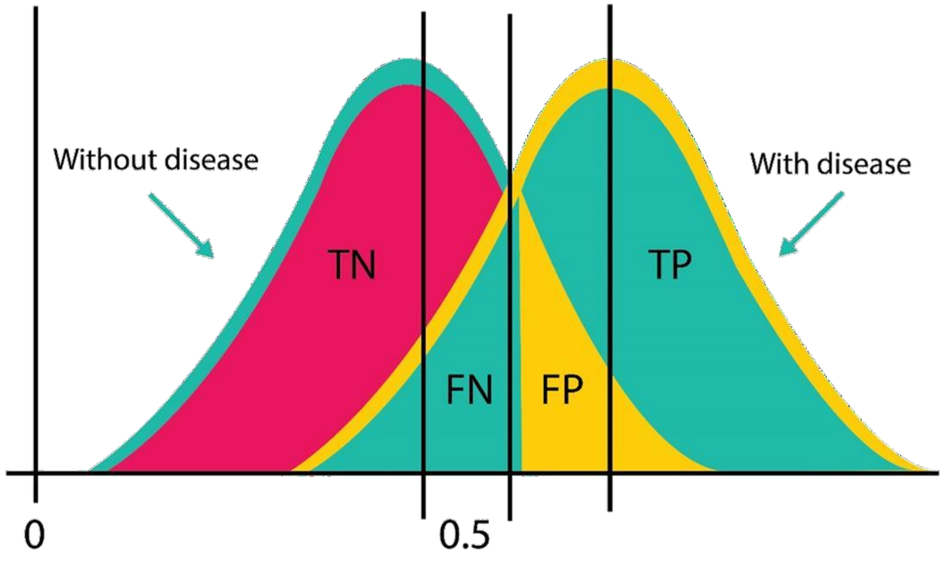
- Precision - in this example would be the number of patients who our model identified as NOT having the disease (TN) over the total number of patients that actually did NOT have the disease.

  - $\text{Precision} = \dfrac{TN}{TN+FP}$



How Thresholds affect Recall and Precision

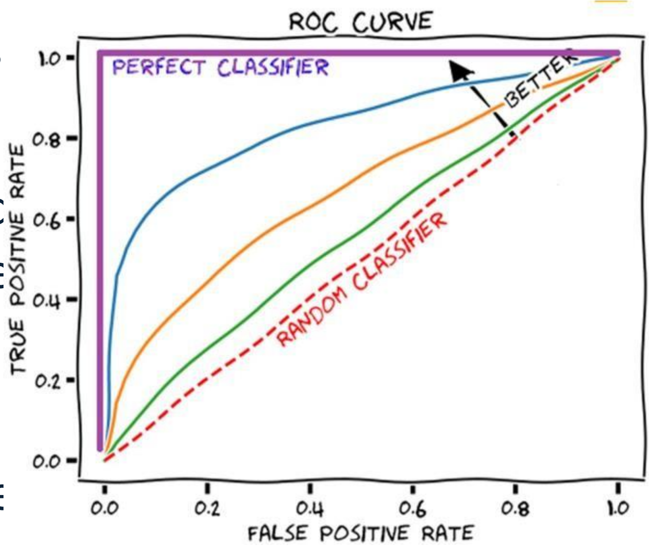
- Let’s make a lower threshold moving it from 0.5 to 0.4.
  - We get more positive predictions of a patient having the disease
  - Increase in FP
  - Decrease in FN
- This will decrease the Precision and increase Recall
- If we reversed and increased it to 0.6 we get:
- More negative predictions (i.e. Less classifications saying we found the disease)
- Decrease in FP
- Increase in FN
- This will increase the Precision and reduce Recall.

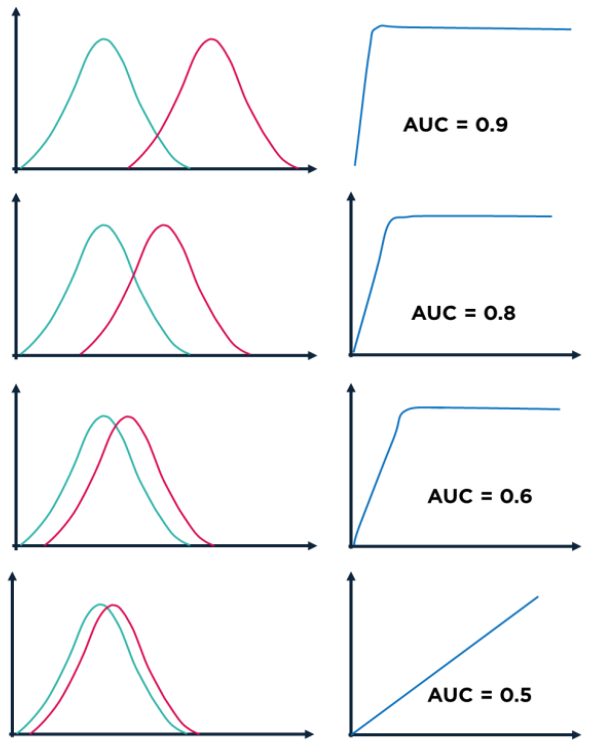


As Recall decreases Precision increases
As Precision decreases Recall increases

The ROC and AUC Curve

- The ROC Curve is the plot of Recall or our True Positive Rate (TPR) against (1- Precision) or our False Positive Rate (FPR).
- TPR is the proportion of people with the diseas that were correctly identified by our model.
- The FPR is the proportion of people identifiec by our model to not have the disease that were false positives.
- The AUC or Area under the Curve is an overall measure of performance, the greater area the better the classifier.





A Gini Coefficient

- **Gini = 2 * AUC -1**

- A Gini coefficient can be used to evaluate the performance of a classifier. A classifier is a model that identifies to which class or category a request belongs to. In credit risk, classifiers can identify if an applicant belongs to the creditworthy or the uncreditworthy categories.

- A Gini is the most commonly used for imbalanced datasets where the probability alone makes it difficult to predict an outcome. The Gini coefficient is a standard metric in risk assessment because the likelihood of default is relatively low.

- Gini is measured in values between 0 and 1, where a score of 1 means that the model is 100% accurate in predicting the outcome. A score of 1 only exists in theory. In practice, the closer the Gini is to 1, the better. Whereas a Gini score equal to 0 means the model is entirely inaccurate. To achieve a score of 0, the model would have to ascribe random values to every prediction

Dealing with Overfitting

- **Hyperparameter Tuning** is a meta-optimization task. The outcome of hyperparameter tuning is the best hyperparameter setting, and the outcome of model training is the best model parameter setting

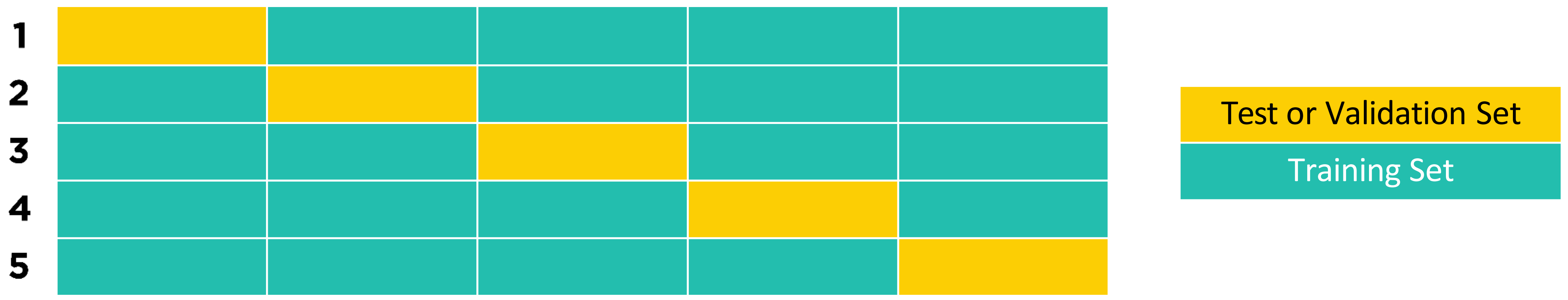
- **Cross Validation** is a procedure used to detect overfitting and estimate the skill of the model on new data.

- **Regularization** is technique used to reduce the error by giving penalty using different functions appropriately on the given training set and avoid overfitting
  - Lasso Regression
  - Ridge Regression
  - Elastic Net Algorithm

Cross Validation

- Cross validation or also known as k-fold cross validation is a method of training where we split our dataset into k folds instead of a typical training and test split.
- For example, let's say we're using 5 folds. We train on 4 and use the 5th final fold as our validation. We then train on the other 4 folds and validate on another.
- We then use the average weights across coming out of each cycle.
- Cross Validation reduces Overfitting but slows the training process

K-folds (5 shows)



Thank you!

Data Science Academy