# Classification

## Agenda

Key Takeaways-

- Classification, its types and algorithms
- k Nearest Neighbors (kNN)
- Types of distance measures
- Hyperparameter Tuning, GridSearchCV
- Cross validation and its types
- K-fold Cross Validation
- Evaluation measures for Classification

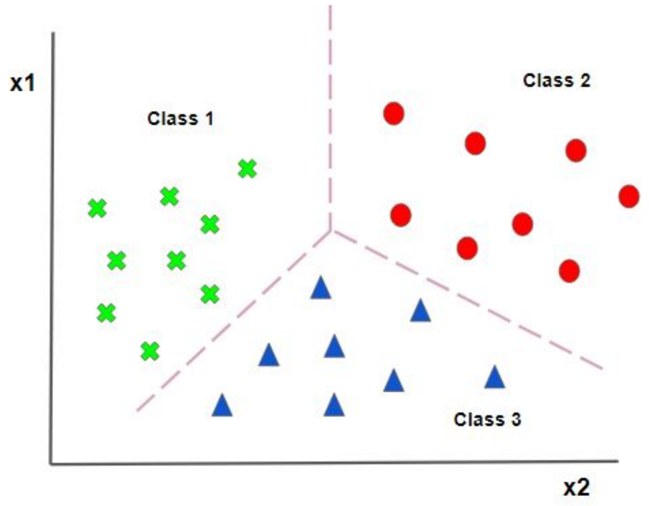
## Classification

Classification is a Supervised ML technique that helps in identifying/predicting the class/category for (new) instances.

If the target variable is categorical/qualitative in nature, then we use classification algorithms.

For example, Identifying

- Whether an email is spam or ham.
- Whether a transaction is fraudulent or not
- The handwritten digits 0-9
- Types of debit cards
- Whether a patient is cancerous or non-cancerous



## Types of Classification Tasks

Based upon the number of unique classes in the target variable, there are 3 types of classification tasks.

1. Binary Classification : The target variable has only two unique classes and the classification task is to identify/predict one of two classes for each instance.
   E.g. Identifying whether a transaction is fraudulent or not.

2. Multi-class Classification : The target variable has more than two unique classes and the classification task is to identify/predict one of more than two classes for each instance.
   E.g. Handwritten digit recognition

3. Multi-label Classification : Each instance of the labelled data belongs to one or more classes and the classification task is to identify/predict one or more classes for each instance.
   E.g. Predicting tags for Quora questions

## Types of Classification Algorithms

The most commonly used classification algorithms are -

1. k Nearest Neighbors (kNN)
2. Decision Tree
3. Random Forest
4. Logistic Regression
5. Support Vector Machine (SVM)

## k Nearest Neighbors (kNN)

k-NN algorithm assumes the similarity/distance between the new data point and available data points and put the new data point into a category having maximum votes in the neighborhood.

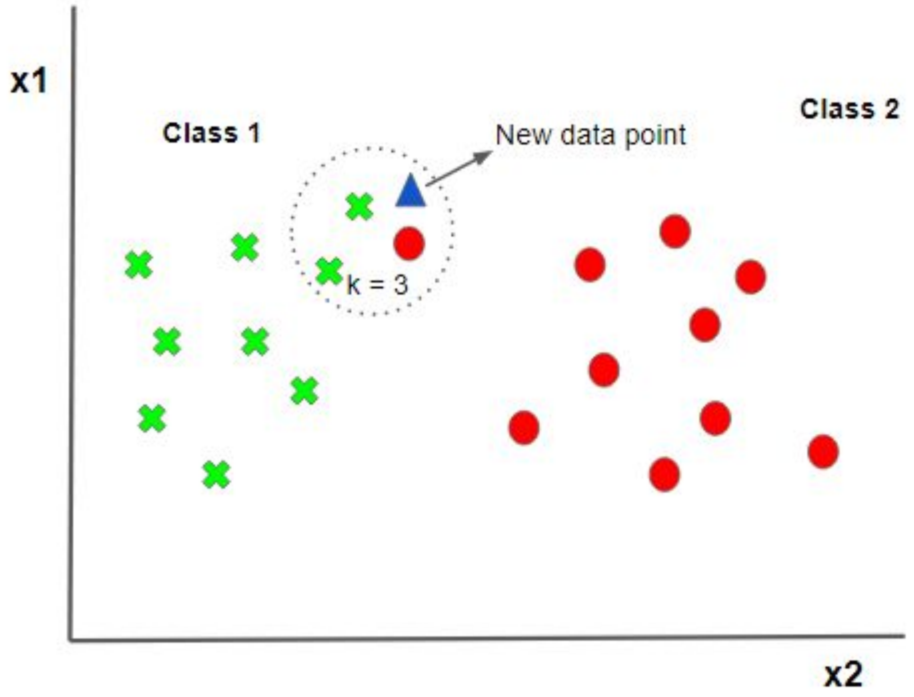
**Step 1**. Choose the k (number of nearest neighbors).

**Step 2**. Take the k nearest neighbors of new data point, based on distance (Euclidean)

**Step 3**. Among the k neighbors, count the number of data points in each category.

**Step 4**. Assign the new data point to the category with maximum counts.

## k Nearest Neighbors (kNN) [Contd.]



## Distance measures

The most commonly used distance measures in kNN are -

- Euclidean distance
- Manhattan distance
- Minkowski distance
- Cosine distance

## Euclidean distance

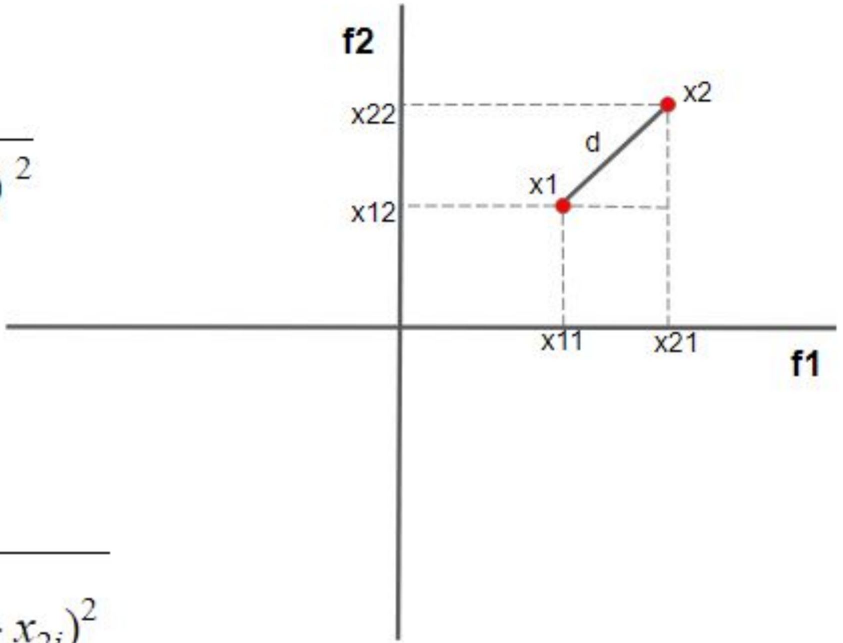
- Euclidean distance (d)

  =length of shortest line from x1 to x2

$$d = \sqrt{(x_{21} - x_{11})^2 + (x_{22} - x_{21})^2}$$

- Euclidean distance is also represented as L2 norm.

$$d = \left\|x_1 - x_2\right\|_2 = \sqrt{\sum_{i=1}^{d} (x_{1i} - x_{2i})^2}$$



## Manhattan Distance

- Manhattan distance (d)

  = Sum of absolute differences

$$d = |x_{21} - x_{11}| + |x_{22} - x_{21}|$$

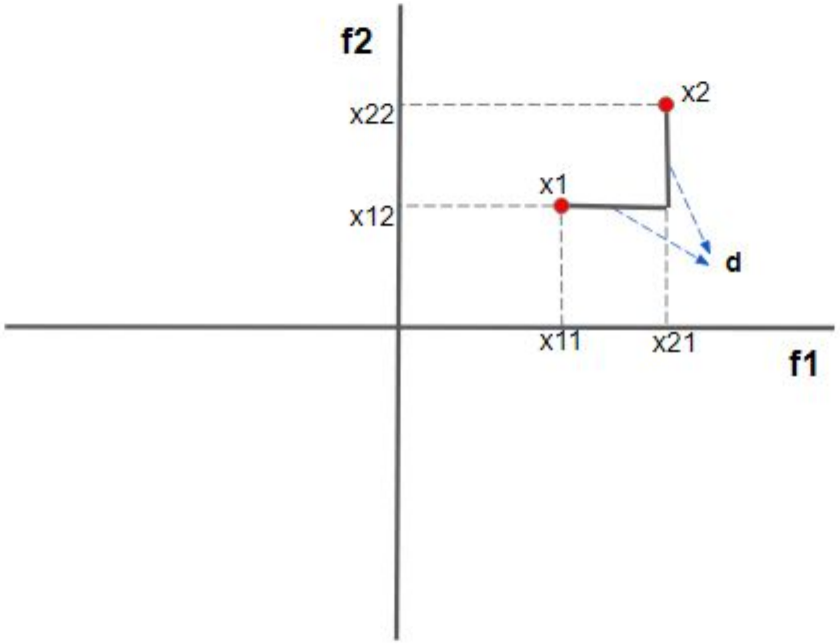
- Its name is from the city "Manhattan"

  Where roads are perpendicular to each other

- Manhattan distance is also represented as

  L1 norm.

$$d = \|x_1 - x_2\|_1 = \sum_{i=1}^{d} |x_{1i} - x_{2i}|$$



## Minkowski Distance

- It is a generalized distance measure.
- Minkowski distance is also represented as Lp norm.

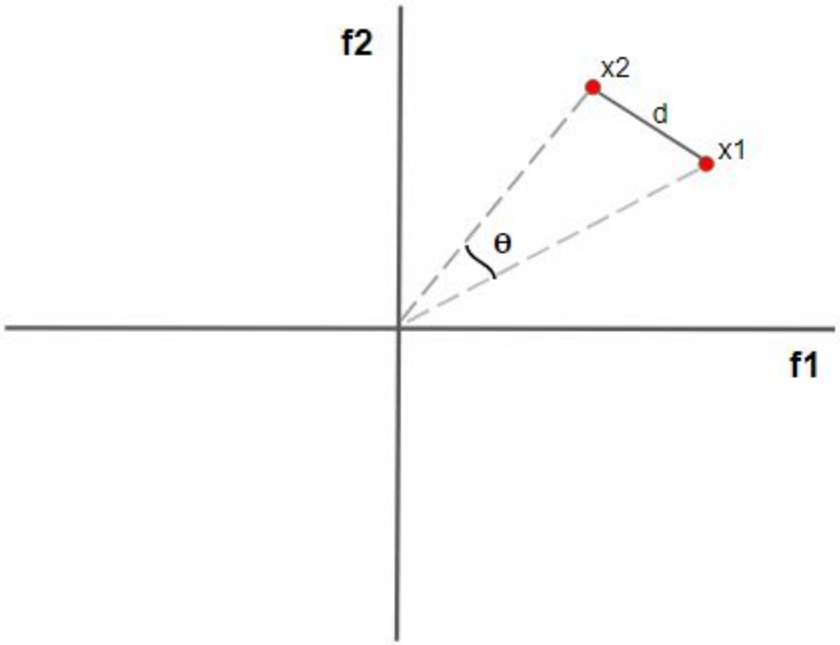
$$\|x_1 - x_2\|_p = \left(\sum_{i=1}^{d} |x_{1i} - x_{2i}|^p\right)^{1/p}$$

- If $p = 1$, then Minkowski distance = Manhattan distance
- Similarly, if $p = 2$ then Minkowski distance = Euclidean distance

## Cosine Distance

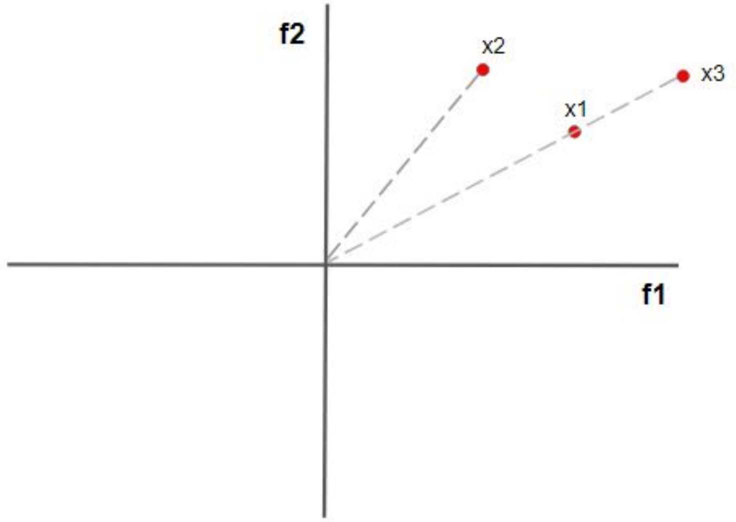
- Cosine distance = (1 - Cosine similarity)

  Where Cosine similarity ≅ Cos θ and θ is the angle between two vectors.



## Quiz 1

Choose the correct statement(s) as per the following vectors.



- Cosine distance (x1, x3) < Euclidean distance (x1, x3)
- Euclidean distance (x2, x3) > Euclidean distance (x1, x2)
- Cosine distance (x1, x2) != 0
- All of the above

## Quiz 2

Choose the correct statement(s).

- As the distance increases the similarity b/w two vectors(entities) also increases.
- As the distance decreases the similarity b/w two vectors(entities) also increases.
- Euclidean distance is similar to L2 norm.
- None of the above

## Model Parameter vs Hyperparameter

| Parameters | Hyperparameters |
|---|---|
| It is a configuration variable that is internal to the model | It is a configuration that is external to the model |
| They are estimated or learned from data (often not set manually) | They are often specified by the practitioner (or often set manually) |
| They are required by the model when making predictions | They are often tuned for a given predictive modeling problem. This process is known as Hyperparameter tuning. |
| E.g. Regression coefficients, neural network weights, etc. | E.g. test_size in train-test splitting process, k (# of nearest neighbors) in kNN, Kernel type in SVM, Maximum depth of tree in decision tree, etc. |

## Cross Validation

- Validation techniques are used to evaluate how well a model would generalize to new/unseen data.

- Cross-Validation is a validation technique used to estimate how well and accurately a predictive model will perform to an independent dataset (real time scenarios).

- Most commonly used cross-validation strategies are -
  - Validation set approach
  - Leave one out cross validation (LOOCV)
  - K-fold cross validation

## Cross Validation [Contd.]

1. **Validation set approach**

- This approach divides the dataset into two equal parts, 50% of the dataset is reserved for training purpose, whereas the remaining 50% is reserved for validation purpose.

- **Disadvantage -** This approach generally leads to a high bias model as there always remains a possibility of missing out on relevant and meaningful information due to considering only 50% of the data.

2. **Leave one out cross validation (LOOCV)**

- As the name suggests, here we reserve only one data point for validation purpose and use rest of the data for training purpose.
- In this approach, number of folds = number of data points in the dataset.
- **Disadvantages -** 1. Higher execution time as it is repeated for n times.

  2. This approach leads to a high variance model.

## K-fold Cross Validation

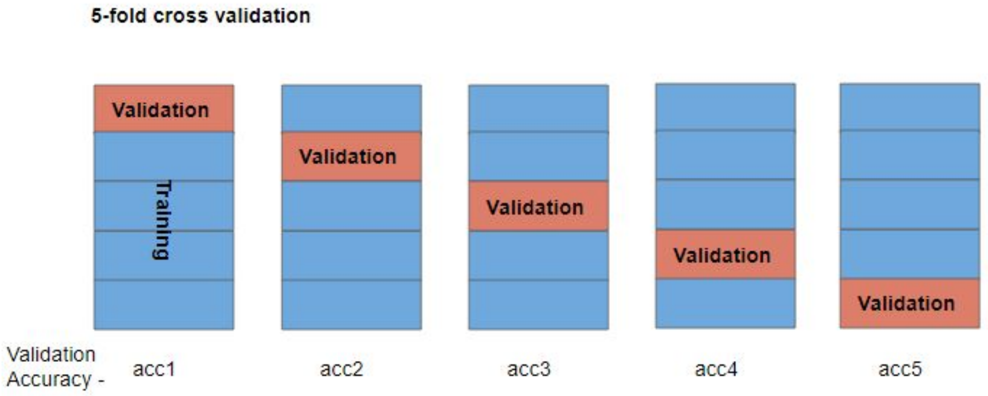
**Step 1**. Randomly split the entire dataset into **k** folds/subsets.

**Step 2**. In each iteration(or kth round), train the model using (k – 1) folds of the dataset and validate/test the model using the kth fold.

**Step 3**. Calculate the accuracy for this iteration.

**Step 4**. Repeat this process until each of the k-folds has served as the validation/test set.

**Step 5**. Take the average of all **k** such accuracies to get the final validation accuracy.



**Final Accuracy** = **Average**(acc1,acc2,acc3,acc4,acc5)

## Quiz 3

In k-fold cross validation, how many subsets are used for validation purpose in each iteration?

- (k - 1) subsets
- K subsets
- One fixed subset in each iteration
- One randomly selected subset in each iteration

## GridSearchCV

- Grid Search is the process of performing hyperparameter tuning in order to determine the optimal values of the hyperparameters for a given model.

- Manually performing hyperparameter tuning -
  - is a time consuming process.
  - Also, it is very hard to keep continuous track of hyperparameters which we have tried and still have to try.

- GridSearchCV is a built-in function in sklearn's model_selection package to perform hyperparameter tuning more efficiently and effectively.

- GridSearchCV tries all the combinations of the values passed and evaluates the estimator(model) for each combination.

- So GridSearchCV is used to find the optimal hyperparameters of a model which results in the most accurate predictions.

## kNN Time Complexity

- Test time complexity, for a dataset with n samples and d features =
  **O(nd) + O(1) + O(1) ≅ O(nd)**

  Where, **O(nd)** for comparing n data points and each point is a d dimensional vector.

  **O(1)** + **O(1)** - to perform majority voting.

- So kNN consumes too much time in making predictions.

## Evaluation metrics for Classification

- The most commonly used and simplest evaluation measure for classification is Accuracy.

- Accuracy is the ratio of total number of correctly classified observations to the total observations (total predictions made).

Mathematically,

$$Accuracy = \frac{Number\ of\ correct\ classifications}{Total\ number\ of\ predictions\ made}$$

**Caveats** - Accuracy gives a false sense of evaluation while dealing with imbalanced data, where most of observation belongs to one class(majority class).

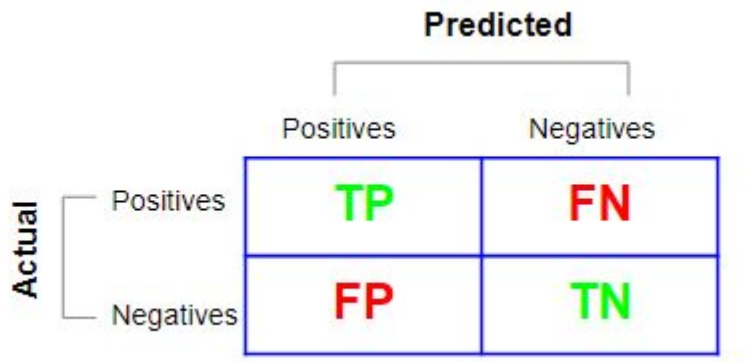
Therefore, we use various other types of evaluation metrics for classification tasks.

- Precision and Recall
- F1-score

## Confusion Matrix

- Confusion matrix provides a more intuitive way to count/know the number of correct and incorrect classifications for all the classes.
- Confusion matrix is a NxN matrix, where N is the total number of classes.

For a binary classification problem, the confusion matrix is as shown below.



- **True Positive (TP)** - Actual class is positive and the model also predicted as positive.
- **False Positive (FP) -** Actual class is negative but the model predicted as positive.
- **False Negative (FN) -** Actual class is positive but the model predicted as negative.
- **True Negative (TN)** - Actual class is negative and the model also predicted as negative.

## Evaluation measures using Confusion Matrix

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

**Precision -**

Precision tells how precise(sure) we are about the predictions. In other words, out of total positive predictions how many are actually positive.

$$Precision = TP / (TP + FP)$$

**Recall -**

Recall is out of total actual positives how many our model predicted as positive.

$$Recall = TP / (TP + FN)$$

**F1-score -**

F1-score combines both precision and recall. It is a weighted average (harmonic mean especially) of both precision and recall.

$$F1\,score = (2 * Precision * Recall) / (Precision + Recall)$$

## Quiz 4

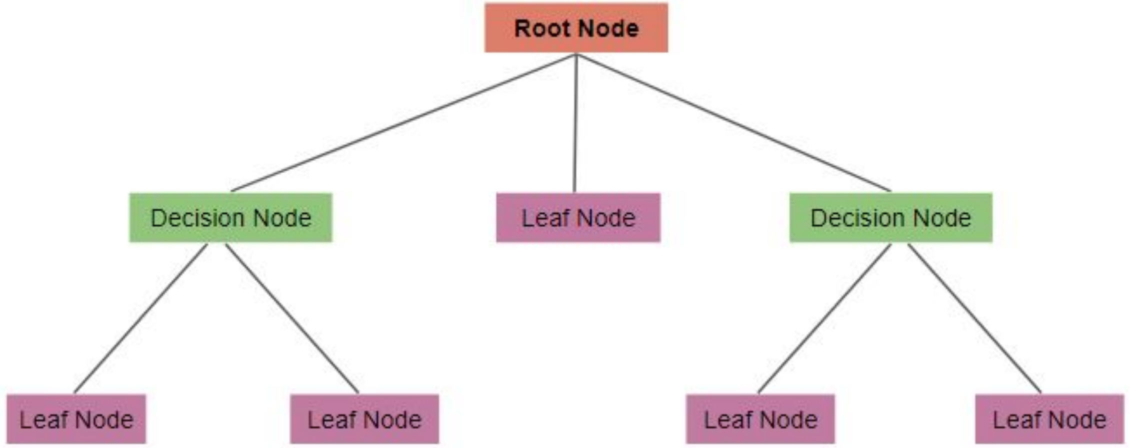
What is the precision for defaulter class based on the below confusion matrix?
Here, the rows (horizontal records) indicate the actual values and the columns (vertical records) indicate the predicted values.

| | Defaulter | Non-defaulter |
|---|---|---|
| **Defaulter** | 55 | 8 |
| **Non-defaulter** | 12 | 25 |

- 55 / (55+8)
- 55 / (55 +12)
- 12 / (55 + 12)
- 12 / (12 + 25)

## Decision Tree

- A tree-like structure having decision nodes and leaf nodes.
  - Decision nodes - To make decisions based on certain conditions
  - Leaf nodes - contain the outcome(class labels)



- To make a prediction for an instance, the path from the root to the leaf is followed as per the conditions.

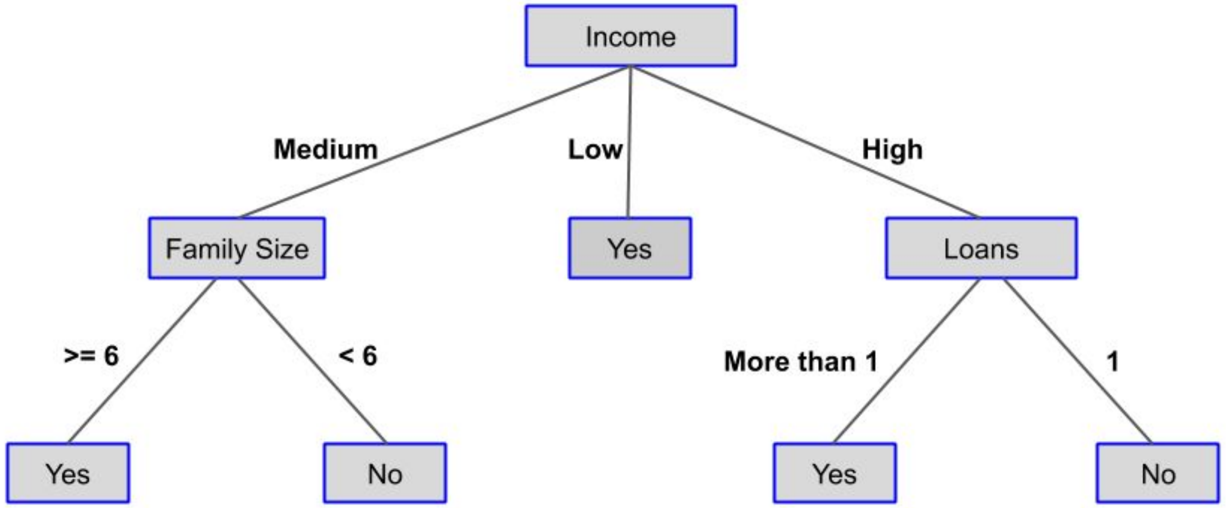
## Decision Tree [Contd.]

Consider a case where a bank has to decide whether a customer will enroll for credit card or not based on the features like Income, Family Size and Loans.

| Income | Family Size | Loans | Credit Card (Yes/No) |
|---|---|---|---|
| Medium | >= 6 | 1 | Yes |
| Medium | >= 6 | More than 1 | Yes |
| Low | >= 6 | 1 | Yes |
| High | >= 6 | 1 | No |
| High | < 6 | More than 1 | Yes |
| Low | < 6 | More than 1 | Yes |
| Medium | < 6 | 1 | No |
| High | < 6 | 1 | No |
| Medium | < 6 | More than 1 | No |
| Low | >= 6 | More than 1 | Yes |
| Low | < 6 | 1 | Yes |
| High | >= 6 | More than 1 | Yes |
| Medium | < 6 | More than 1 | No |
| High | >= 6 | More than 1 | Yes |

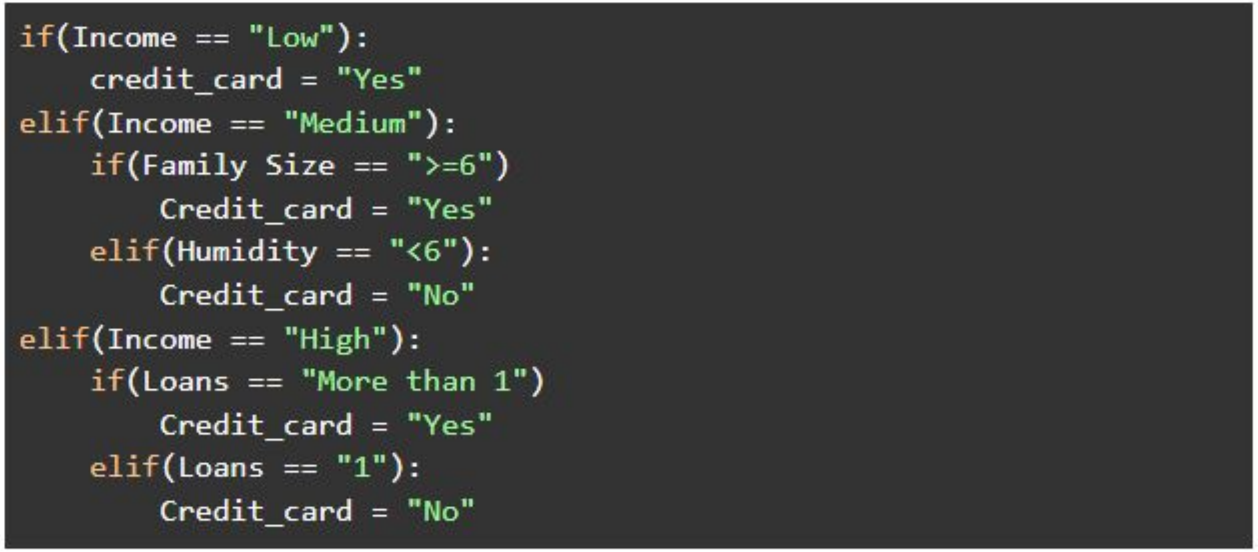
## Decision Tree [Contd.]

So if we try to predict the credit card enrollment based on the features like Income, Family Size and Loans using decision tree then the tree may look like - .



## Decision Tree [Contd.]

The previous decision tree can be further simplified as -



```
if(Income == "Low"):
    credit_card = "Yes"
elif(Income == "Medium"):
    if(Family Size == ">=6")
        Credit_card = "Yes"
    elif(Humidity == "<6"):
        Credit_card = "No"
elif(Income == "High"):
    if(Loans == "More than 1")
        Credit_card = "Yes"
    elif(Loans == "1"):
        Credit_card = "No"
```

So the decision tree can also be referred as a nested if-else classifier.

## Quiz 5

Which of the following nodes in Decision Tree contains the conditions (decisions).

- Root Node
- Internal Nodes
- Leaf Nodes
- All the nodes

## How does the Decision Tree algorithm work?

**Step 1.** Select the best attribute using Attribute Selection Measures(ASM), for e.g. Information Gain (mostly used).

**Step 2.** Make that attribute a decision node and split the dataset into smaller subsets.

**Step 3.** Start tree building by repeating this process recursively for each child until one of the condition is satisfied -

- There are no more remaining attributes.
- All the tuples belong to the same attribute value.
- There are no more instances.

## Attribute Selection Measures (ASM)

- To build a decision tree for a dataset having **d** features, the major challenge is to decide which feature to use at root node and at other decision nodes. To fix this, Attribute Selection Measures (ASM) are used.

- Attribute Selection Measures (ASM) compare different attributes/predictors and rank them for the purpose of tree/model building.

- The most commonly used attribute selection measures are -
  - Information Gain
  - Entropy
  - Gini Index

## Information Gain

- Information Gain measures how much information about the classes(categories) can be gained using a feature.

- The feature with highest information gain is taken at the root node (or decision node).

Information gain further depends on **Entropy**.

**Entropy :** It measures randomness in the data. Higher the entropy, more difficult to draw conclusions from the information.

- Entropy for a dataset(S) with C classes is computed as -

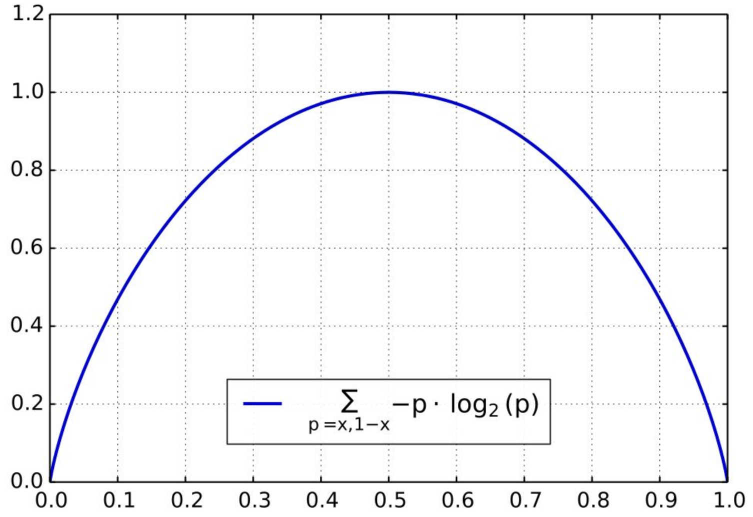
$$Entropy(S) = -\sum_{i}^{C} p_i \log_2(p_i)$$

Where, $p_i$ is the probability of class *i*.

## Properties of Entropy

- If all the class labels are equally probable (or equally divided) then entropy is maximum.
  E.g. In a binary classification, if (y+ = 50% and y- = 50%) then
  H(y) = -0.5 log(0.5) - 0.5 log(0.5) = 1

- If only one class is fully dominating (most probable) then entropy is minimal.
  E.g. In a binary classification, if (y+ = 99% and y- = 1%) then
  H(y) = -0.99 log(0.99) - 0.01 log(0.01) = 0.08



## Attribute Selection Measures [Contd.]

- Information Gain is computed as -

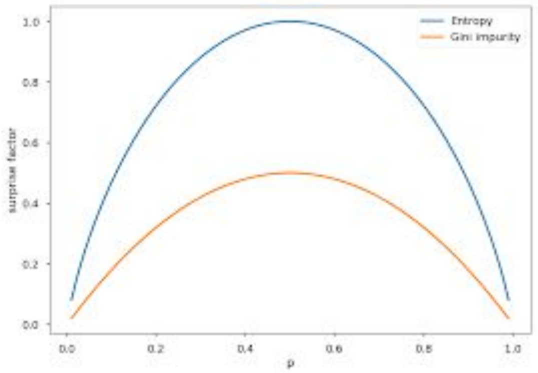
$$Information\ Gain = Entropy(S) - [\{weighted\ average) * Entropy(each\ feature)]$$

### Gini Index

- It measures how often a randomly picked instance would be incorrectly classified.
- It is a measure of impurity and an attribute with low Gini index is preferred first.

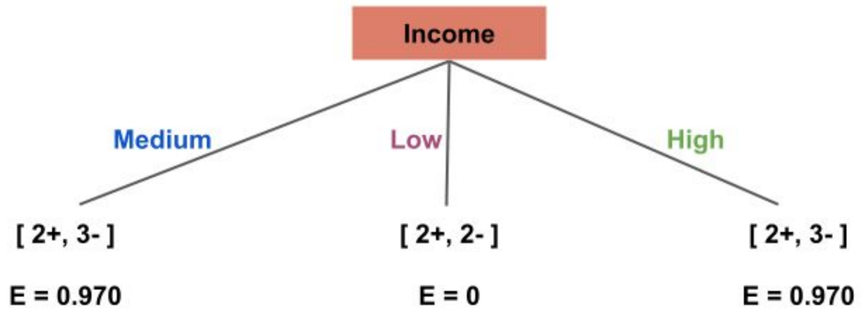
Mathematically,

$$Gini\ Index = 1 - \sum_{i}^{C}(p_i)^2$$



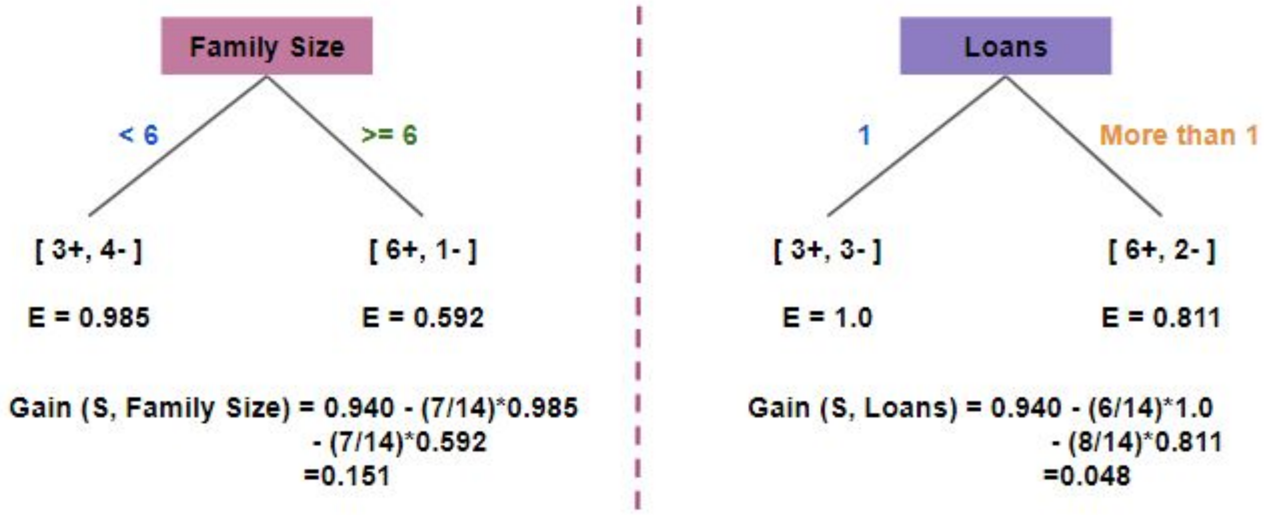
## Attribute at the Root Node

- Attribute with highest Information Gain is selected at the root node.



Gain (S, Income) = 0.940 - (5/14)*0.970 - (4/14)*0 - (5/14)*0.970 = 0.247

## Attribute at the Root Node [Contd.]

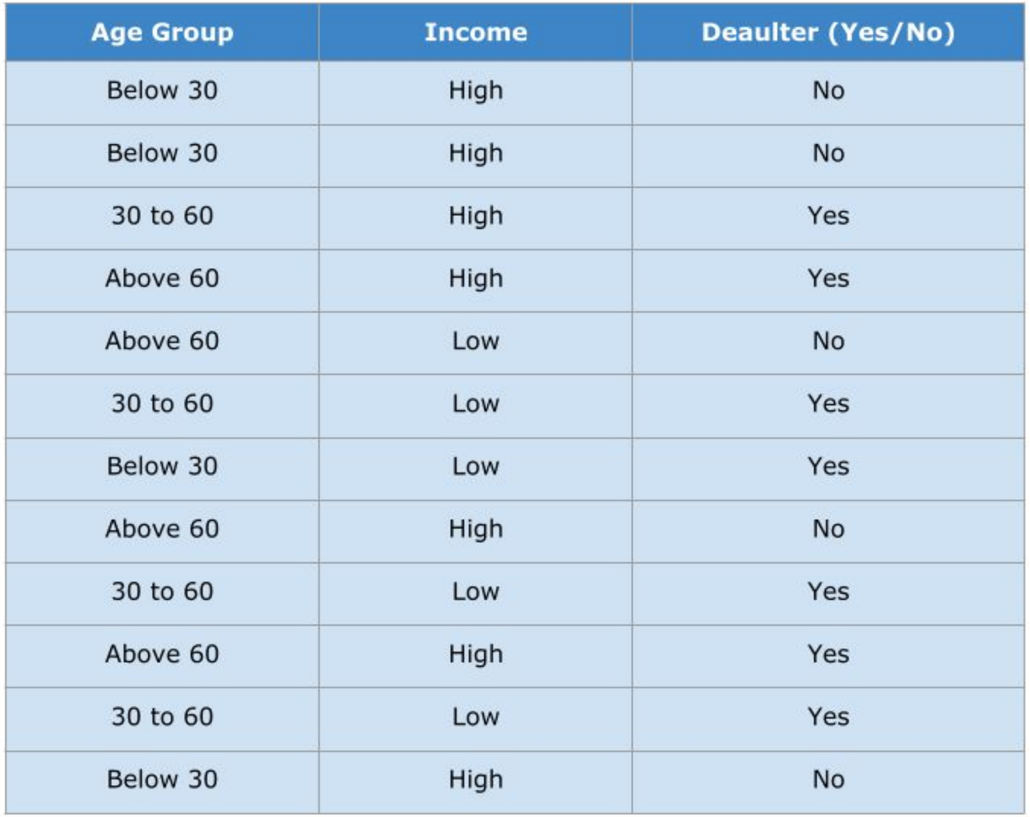


- So the attribute with highest information gain is **Income**.

## Quiz 6

Which of the following age group subset has entropy = 0?

- Above 60
- 30 to 60
- Below 30
- None of these



| Age Group | Income | Deaulter (Yes/No) |
|---|---|---|
| Below 30 | High | No |
| Below 30 | High | No |
| 30 to 60 | High | Yes |
| Above 60 | High | Yes |
| Above 60 | Low | No |
| 30 to 60 | Low | Yes |
| Below 30 | Low | Yes |
| Above 60 | High | No |
| 30 to 60 | Low | Yes |
| Above 60 | High | Yes |
| 30 to 60 | Low | Yes |
| Below 30 | High | No |

## Decision Tree Time & Space Complexity

- Training time complexity

  for a dataset with n samples and d features = **O(n (log n) d)**

  Where, **n (log n)** corresponds to Sorting

  **d** - to evaluate d features every time

- Run (test) complexity
  - Time complexity : **O(depth)** = **O(k)**, where k is the maximum depth of the tree.

    In order to predict the class for an instance we need to make k decisions.

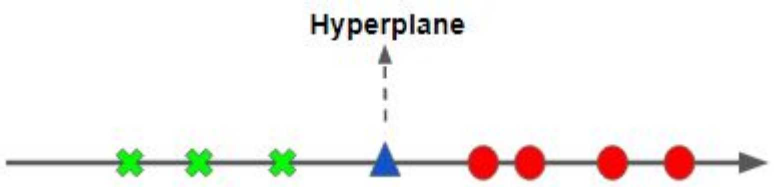
  - Space complexity : **O(nodes)** = # of internal nodes + # of leaf nodes

- So decision tree can handle large data with small(significant) number of features.

## Support Vector Machine (SVM)

- Support Vector Machine (SVM) tries to find out an optimal **hyperplane** (in a **d**-dimensional space) that can easily separate the data points into classes.

- Hyperplane is a decision boundary that classifies the data points. Its dimension varies as per the number of features varies. E.g. For a dataset having two features, the hyperplane will be a line whereas for a dataset having 3 features, the hyperplane will be a plane in 3d space.



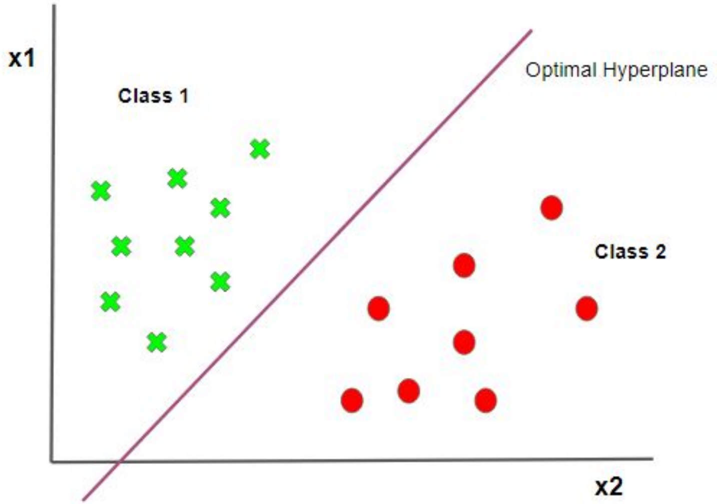
Class 1, for any datapoint to its left

Class 2, for any datapoint to its right

1d Hyperplane (Point)

## Hyperplane [Contd.]

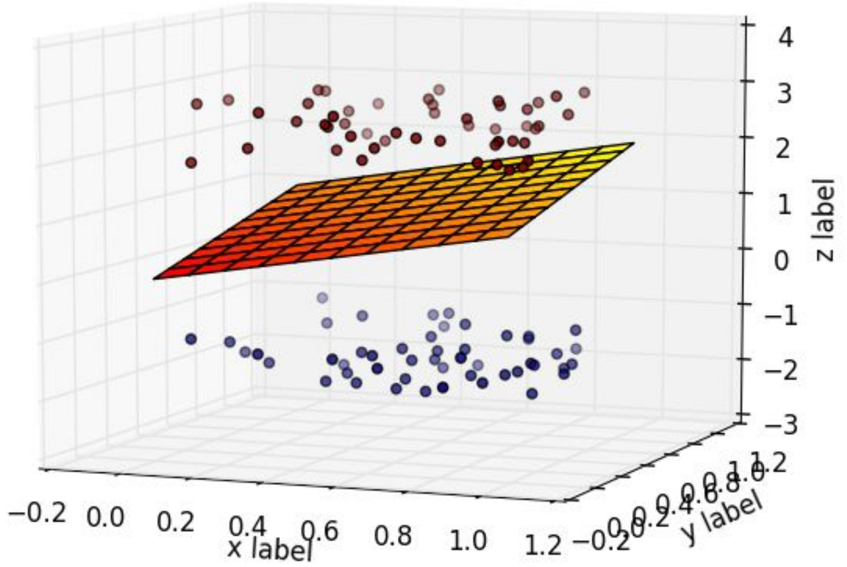
2d Hyperplane (Line)



Class 1, for any datapoint to its left

Class 2, for any datapoint to its right

3d Hyperplane (Plane)



Class 1, for any datapoint above the hyperplane.

Class 2, for any datapoint below the hyperplane.

## Quiz 9

Choose the correct statement(s) about Figure 1 and 2.

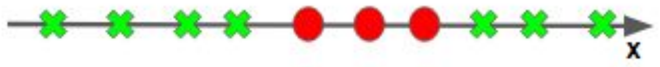


Figure 1

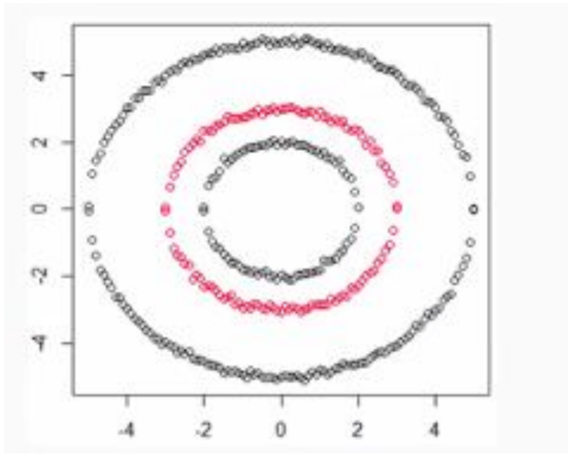
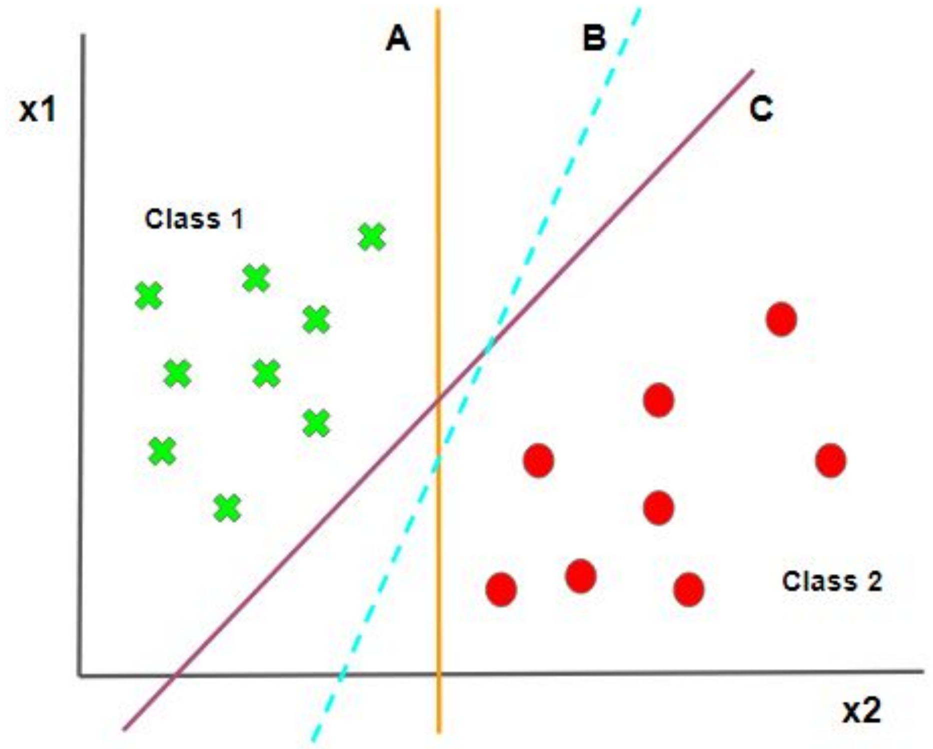


Figure 2

- Figure 1 and 2 both are linearly separable.
- Figure 1 is linearly separable but 2 is not.
- Figure 2 is linearly separable but 1 is not.
- Figure 1 and 2 both are non-linearly separable.

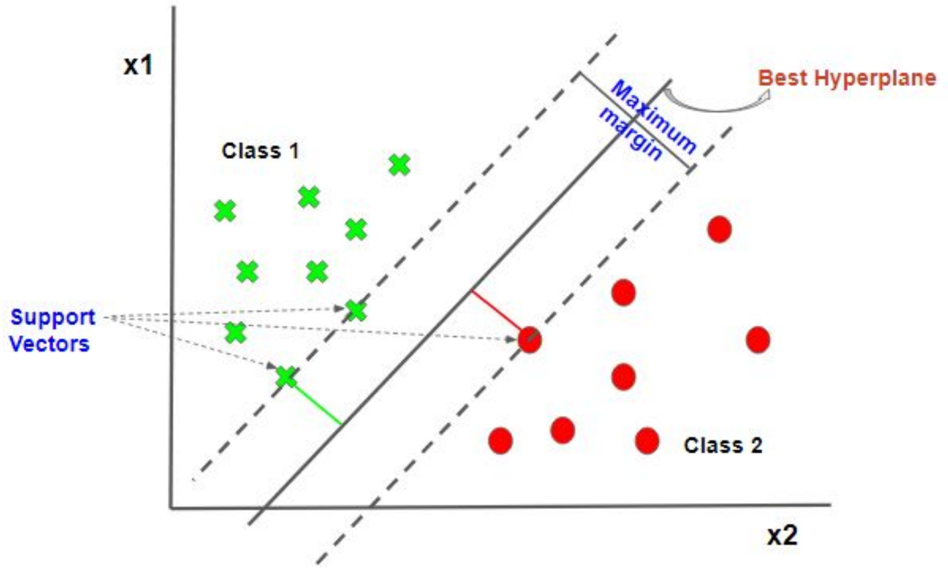
## How to find an optimal Hyperplane

Multiple hyperplanes like A, B and C are possible then how to find the optimal hyperplane?



## How to find an optimal Hyperplane

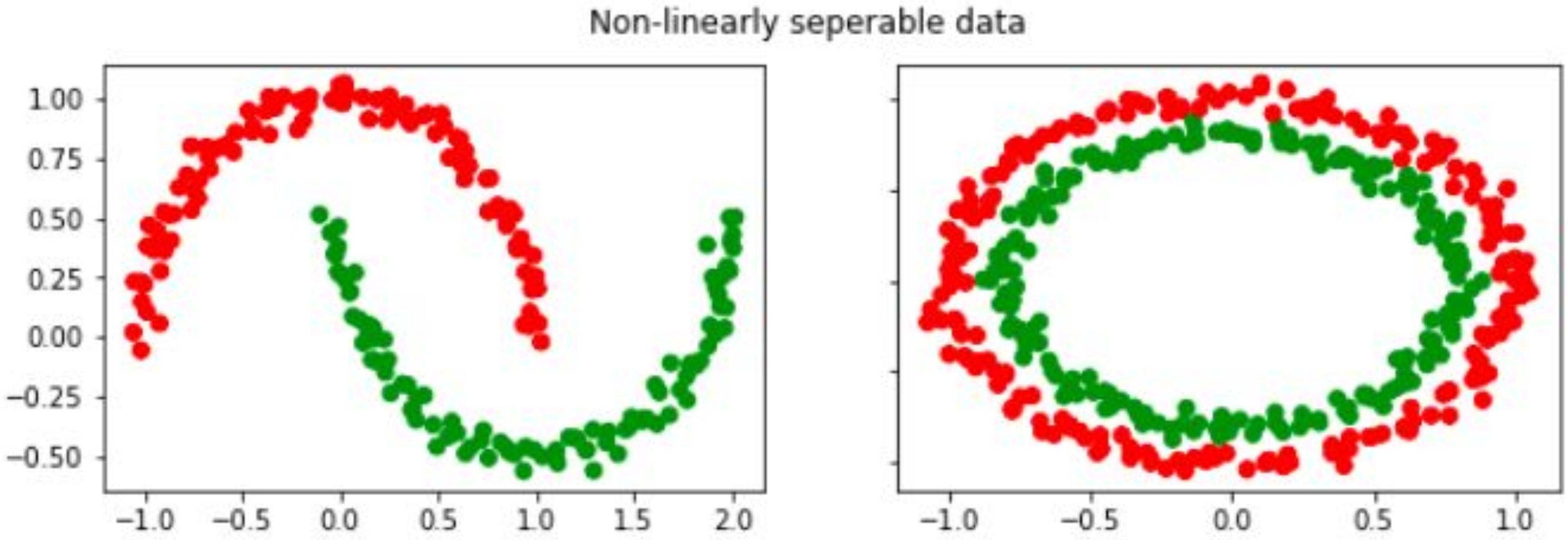
In finding the best hyperplane, the distance between the closest point of each class and the hyperplane plays an important role. This distance is known as **margin** and SVM tries to find a hyperplane with the maximum margin.



**Support vectors** - The data points(vectors) that are closest to the hyperplane and dominate in deciding the position and orientation of the hyperplane.

## Non-linearly Separable data

- The datasets which can not be easily separated into classes using a single line.



## Types of SVM

SVM can be further classified in two categories.

- **Linear SVM** - Best suits to linearly separable data. If a dataset can be easily separated in two classes using a single line then it is known as linearly separable data.

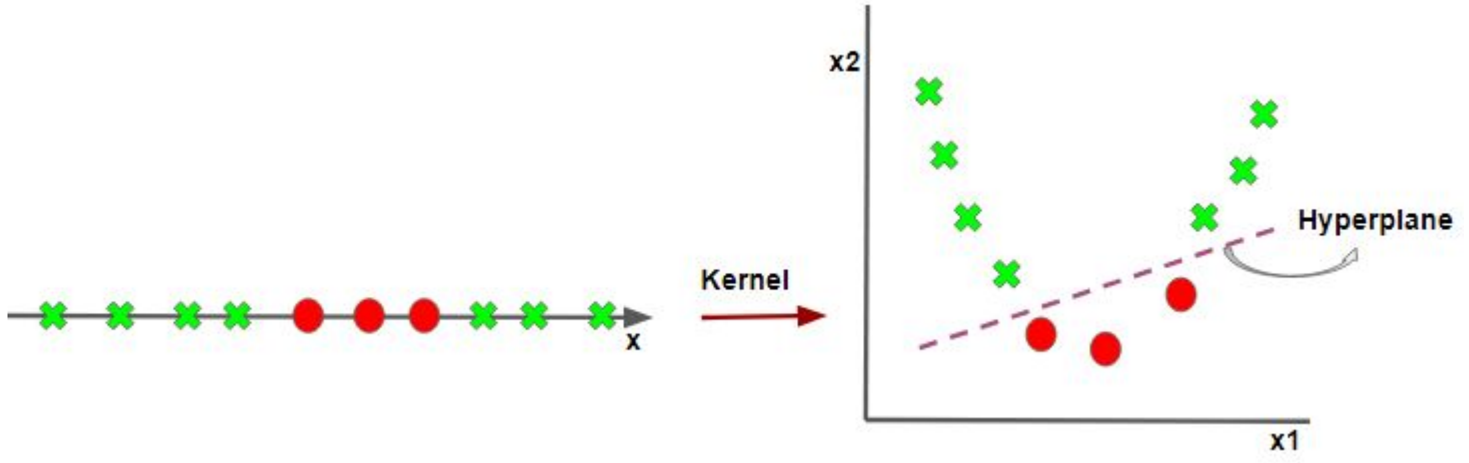
- **Non-Linear SVM -** As the name suggests, it is best suited to non-linearly separable data.In this, the dataset is mapped to a higher dimensional space using kernel trick to get an optimal hyperplane.

  Most commonly used kernels are polynomial and radial.

## Kernel Trick

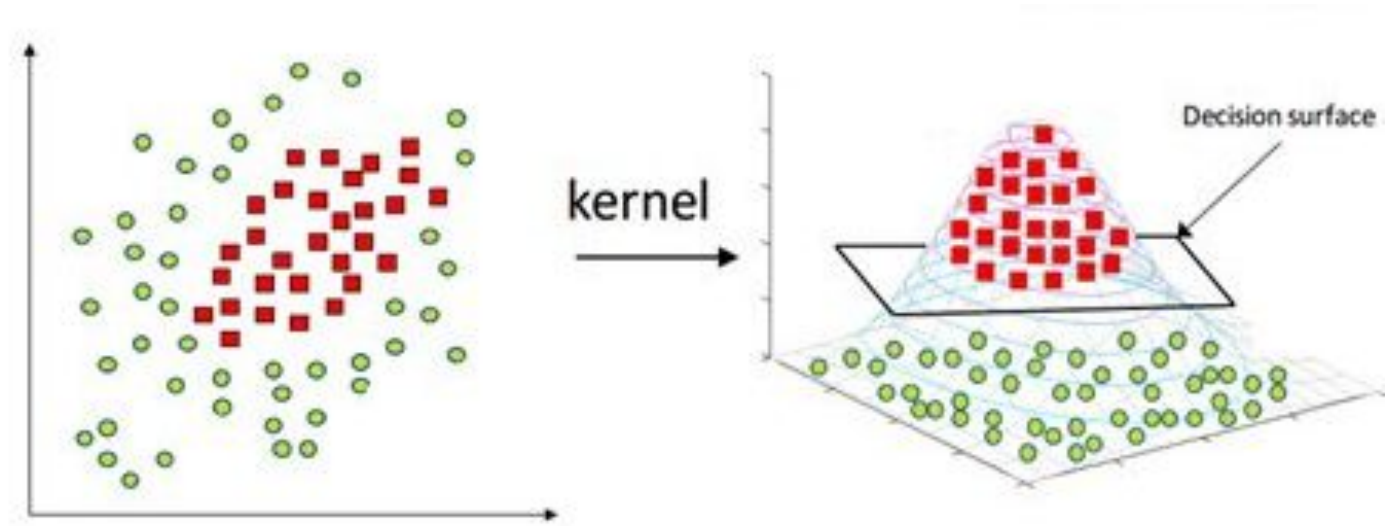
- The non-linearly separable dataset often becomes separable after mapping the original feature space to a higher dimensional space. This entire mapping process is known as kernelization.
- So Kernel methods are used to transform the data into a higher dimensional space such that the data becomes separable in the newer space.

**Example 1 -**



## Kernel Trick [Contd.]

**Example 2 -**



## Types of Kernel

The most commonly used kernels in SVM are -

1. Linear Kernel
2. Polynomial Kernel
3. Radial Basis Kernel (RBF)

### Polynomial Kernel

In polynomial kernel, the dot product by increasing the power of the kernel is computed.

.

$$K(X_1, X_2) = (1 + X_1{}^T X_2)^b$$

Here, **b** = degree of polynomial kernel

## Polynomial Kernel [Contd.]

Let’s say there are two points xa, xb in a original feature space (2-dimensional) as

$$x_a = (a_1, a_2) \qquad x_b = (b_1, b_2)$$

Now if we map them to a higher dimensional space using a polynomial kernel of degree 2

$$K(x_a, x_b) = (1 + x_a{}^T x_b)^2$$

$$= (1 + a_1 b_1 + a_2 b_2)^2$$

$$= (1 + a_1{}^2 b_1{}^2 + a_2{}^2 b_2{}^2 + 2\, a_1 b_1 + 2\, a_2 b_2 + 2 a_1 b_1 a_2 b_2)$$

$$\Rightarrow [1, a_1{}^2, a_2{}^2, a_1, a_2, a_1 * a_2] \rightarrow x_a'$$

$$[1, b_1{}^2, b_2{}^2, b_1, b_2, b_1 * b_2] \rightarrow x_b'$$

$$= (x_a')^T (x_b')$$

This new feature space is 6 dimensional.

## RBF (Radial Basis Function) Kernel

- This is the most popular and general purpose kernel.

Mathematically,

$$K_{RBF}(X_1, X_2) = e^{\left(\frac{-||X_1 - X_2||^2}{2\sigma^2}\right)}$$

Where $\|X_1 - X_2\|$ is the Euclidean distance between X1 and X2.

- This kernel is also known as Gaussian Kernel.

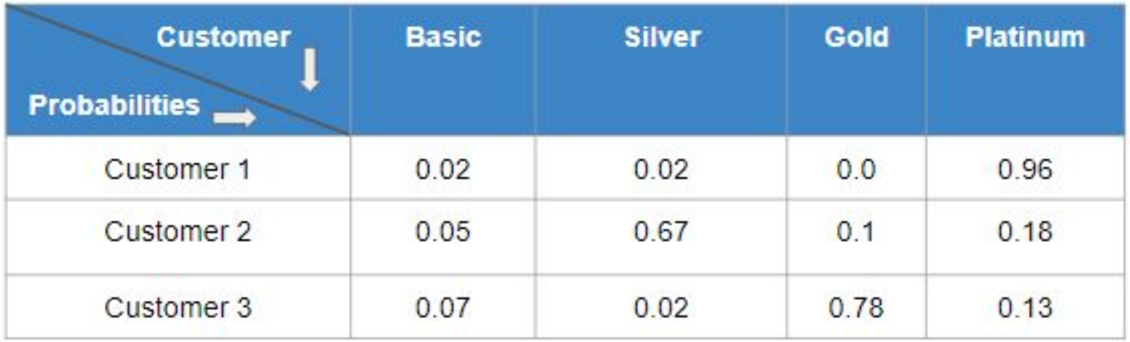
## Quiz 10

Choose the correct statement(s).

- In Kernelization process, the data is mapped to a lower dimensional space.
- SVM without any kernel methods is analogous to linear SVM.
- As the Euclidean distance increases in RBF kernel, its value decreases.
- All of the above

## Logistic Regression

Consider the below table which represents the probability distributions of debit cards for different customers.



| Customer ↓ Probabilities → | Basic | Silver | Gold | Platinum |
|---|---|---|---|---|
| Customer 1 | 0.02 | 0.02 | 0.0 | 0.96 |
| Customer 2 | 0.05 | 0.67 | 0.1 | 0.18 |
| Customer 3 | 0.07 | 0.02 | 0.78 | 0.13 |

Here,

- For customer 1, there is a 96% chance that he needs/deserves a Platinum debit card, 2% chance that he needs a Basic type of debit card and so on.
- For customer 2, there is a 67% chance that he needs/deserves a Silver debit card, 18% chance that he needs a Platinum debit card and so on.
- For customer 3, there is a 78% chance that he needs/deserves a Gold debit card, 13% chance that he needs a Platinum debit card and so on.

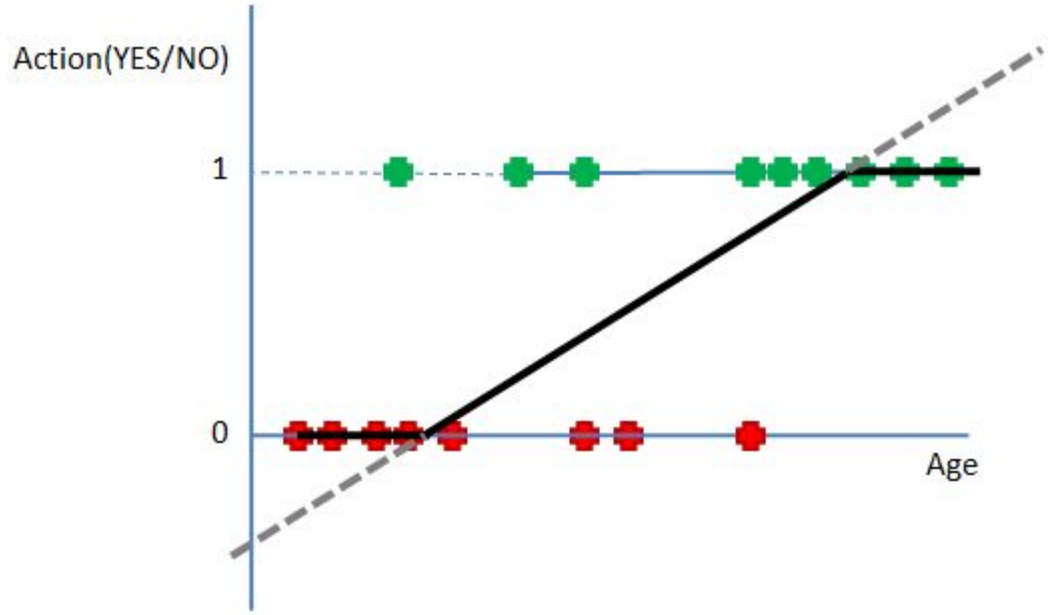
Can we have a classification model that computes the probability of an instance belonging to each of the classes?

## Logistic Regression [Contd.]

- Logistic Regression computes the probability for each class of the target variable.

**OR**

- It provides the probability distribution of the target variable across different class labels.



## Logistic Regression [Contd.]

- Logistic regression is a special case of linear regression where instead of predicting a real number (continuous), it computes the probability values for each of the class.

- In order to get the probability values, we pass the weighted sum of inputs through a squashing function that returns values between 0 and 1 (probabilities). This mapping function is known as **sigmoid** function.

Mathematically, the linear regression equation -

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots\dots\dots\dots + \beta_n X_n$$

And Sigmoid function equation -

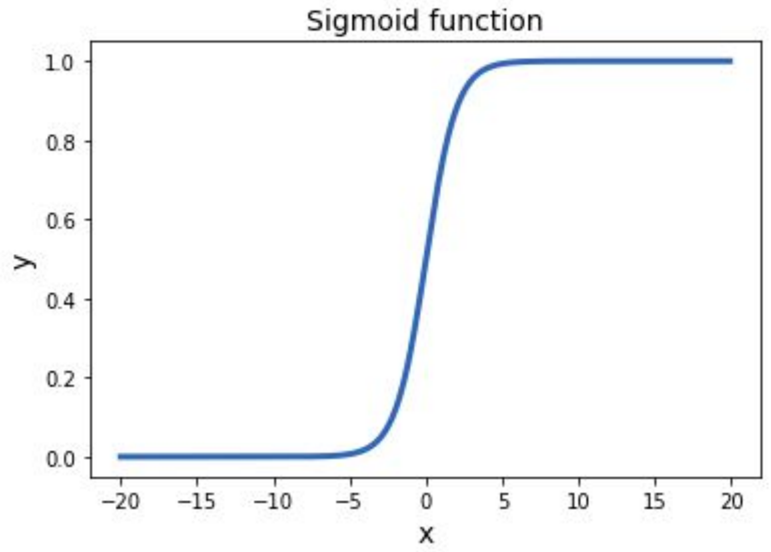
$$p = \frac{1}{(1 + e^{-y})}$$

After combining the sigmoid equation with linear regression equation, the logistic regression equation becomes

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots\dots\dots + \beta_n X_n)}}$$

## Sigmoid Function

- It is a "**S**" shaped curve as shown below.



**Properties**

- Its output ranges from 0 to 1.
- For a large positive input value, it tends towards 1 whereas for a large negative value, it tends towards 0.
- At x = 0, its value is 0.5.

## Quiz 11

Choose the correct statement(s).

- Logistic regression is a form of linear regression.
- The sigmoid function output values ranges from -1 to 1.
- Logistic regression provides normally distributed probability values of the target variable across different classes.
- None of the above

## Log Loss

- Log loss is a probability based evaluation measure for classification tasks.

- Log loss is the negative average of the log of probability value of correct class label.

Mathematically, for a binary classification problem -

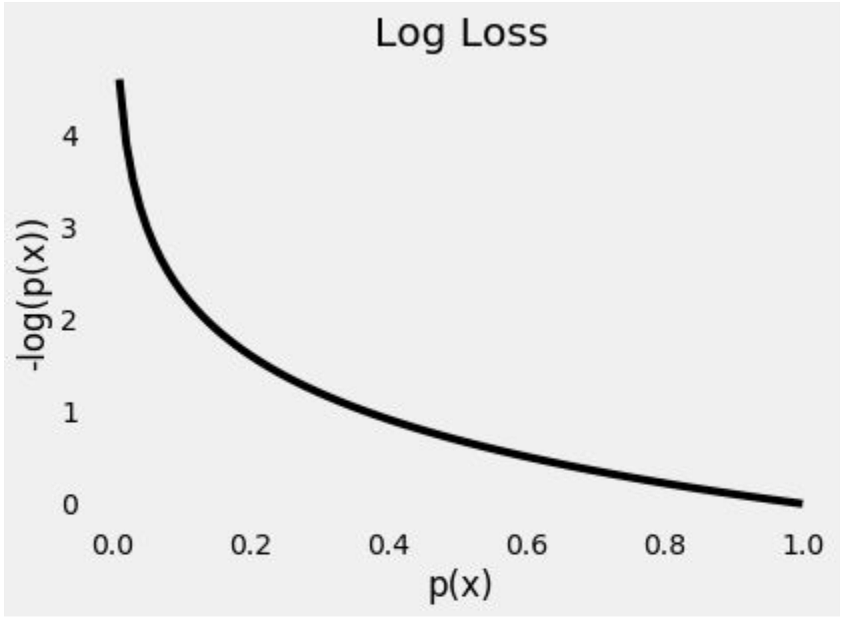
$$\text{Log loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Here $y_i$ represents the actual class and $p(y_i)$ is the probability of that class.

**Case 1**. When the actual class label is 1, then the 2nd term will be zero.

**Case 2**. When the actual class label is 0, then the 1st term will be zero.

## Log Loss [Contd.]



**Insights -**

- Log loss ranges from 0 to infinity.
- For a positive class (or class 1) if $p(y_i)$ is high then the log loss will be low.
- For a negative class (or class 0) if $p(y_i)$ is low then the log loss will be low.

## Log Loss for Multi-Class Classification problem

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

*Where,*

| | |
|---|---|
| $N$ | *No of Rows in Test set* |
| $M$ | *No of Fault Delivery Classes* |
| $Y_{ij}$ | *1 if observation belongs to Class j; else 0* |
| $P_{ij}$ | *Predicted Probability that observation belong to Class j* |

**Drawback of Log Loss-**

- Log loss is hard to interpret. It is a relative term.

## Quiz 12

Which of the following prediction subsets has a better log loss?

| $y_i$ | $p(y_i)$ |
|---|---|
| 1 | 0.95 |
| 0 | 0.02 |
| 1 | 0.84 |

Predictions Set1

| $y_i$ | $p(y_i)$ |
|---|---|
| 1 | 0.15 |
| 0 | 0.04 |
| 1 | 0.68 |

Predictions Set2

- Predictions Set1
- Predictions Set2
- Both

## AUC-ROC Curve

- The **Receiver Operator Characteristic (ROC)** curve is an evaluation metric for binary classification problems where the model predicts the probabilities of different classes for an instance.
- It is a plot of the True Positive Rate (y-axis) versus the False Positive Rate (x-axis) at different classification thresholds between 0.0 and 1.0.
- The model performance is determined by looking at the area under the ROC curve (or AUC).

True Positive Rate(Sensitivity) = True Positives / (True Positives + False Negatives) = TP / (TP + FN)

False Positive Rate = False Positives / (False Positives + True Negatives) = FP / (FP + TN)

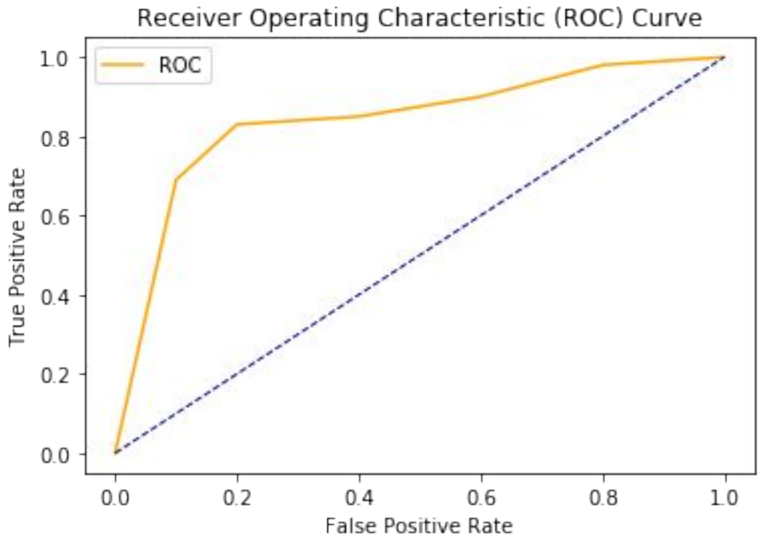
- TPR(True Positive Rate) describes how good the model is at predicting the positive class when the actual outcome is positive.
- FPR(False Positive Rate) describes how often a positive class is predicted when the actual outcome is negative.

FPR  = (1 - TNR)

True Negative Rate(Specificity) = True Negatives / (True Negatives + False Positives) = TN / (TN +FP)

So FPR = 1 - Specificity

## AUC-ROC Curve [Contd.]



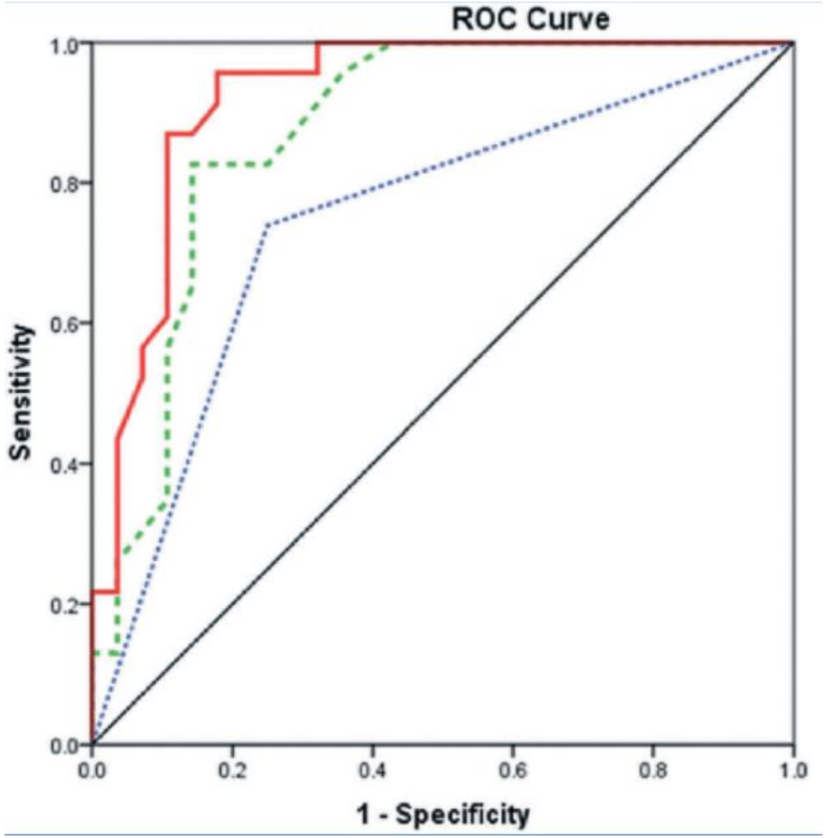
**Properties**

- AUC lies between **0** and **1**.
- For a random model (randomly predicting 0 or 1), the AUC = **0.5**
- For a good model (better than random model) the **0.5** < AUC < **1**
- For a worst model (worse than random model) the **0** < AUC < **0.5**

## Quiz 13

Choose the model with highest roc-auc score.

- Red
- Green
- Blue



## Random Forest

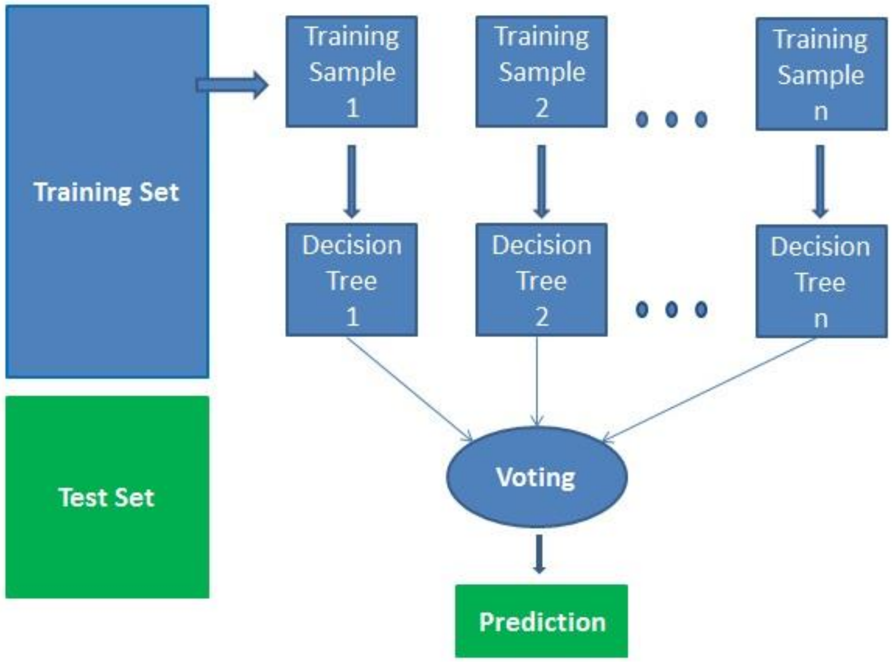
- A forest is comprised of trees, similarly random forest classification is a collection of different decision trees.
- Generally, a random forest model is robust if it has more(significant) number of trees.
- Random forest creates decision trees on randomly selected data samples, gets prediction from each tree and makes the final prediction by performing majority voting (in case of classification).
- Random forest often decreases the high variance of decision tree model.

# Random Forest [Contd.]

Step 1. Select random samples from a given dataset.

Step 2. Construct a decision tree for each sample and get a prediction result from each decision tree.

Step 3. Select the prediction result with the most votes as the final prediction.



## Random Forest Time & Space Complexity

- Training time complexity

  for a dataset with n samples and d features = **O(n (log n)*d*k)**

  Where, **n (log n)** corresponds to Sorting

  **d** - to evaluate d features every time

  **k** - number of decision trees used

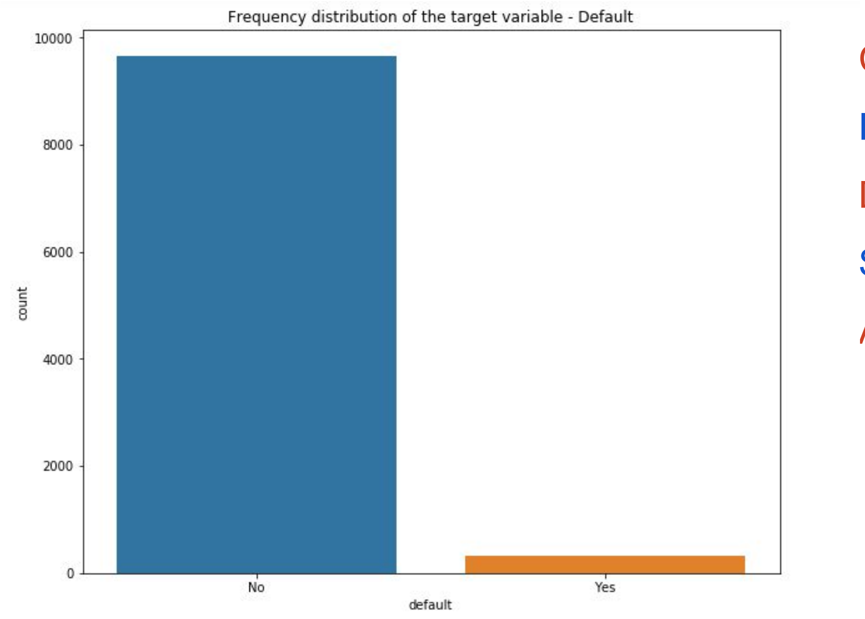
- Run (test) complexity
  - Time complexity : **O(depth*k)** = where k is the number of decision trees used.

  - Space complexity : **O(nodes*k)**

## Imbalanced Datasets

Imbalanced datasets are a special case in classification, where most of the observations belong to one class (majority class). In other words the class distribution is not uniform among the classes.

Example - Consider a case of 10,000 customers, where 9667 are non-defaulters and only 333 are defaulters.



Class imbalance may occur in many domains like -

Fraud detection

Disease screening

Spam Filtering

Advertising click-throughs

## Imbalanced Datasets [Contd.]

**Problem with Class Imbalance -**

- Since most of the ML algorithms are designed to maximize accuracy and reduce errors so the algorithms remain biased towards the majority class.

- Therefore, we may get a high accuracy but the model fails to capture minority class.

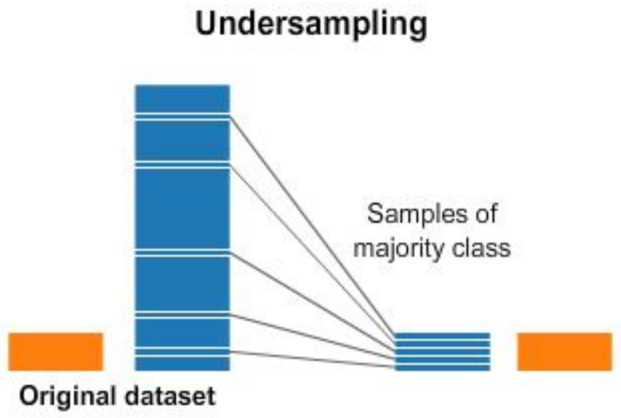
**Resampling techniques to tackle Class Imbalance**

Resampling is the most commonly used and adopted technique for dealing with highly imbalanced datasets.

1. UnderSampling (DownSampling)
2. OverSampling (UpSampling)
3. SMOTE (Synthetic Minority OverSampling Technique)

## UnderSampling (DownSampling)

Undersampling is the process of randomly removing observations from the majority class until the majority and minority class are balanced out.

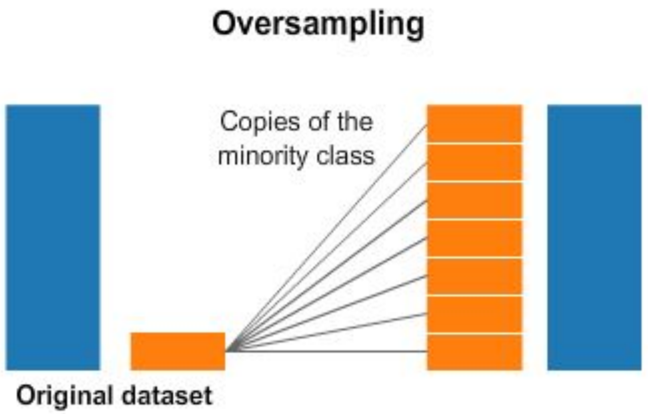


**Disadvantages**

- Since we are reducing the majority class instances here so the model may miss/fail to capture the important and meaningful information/pattern.

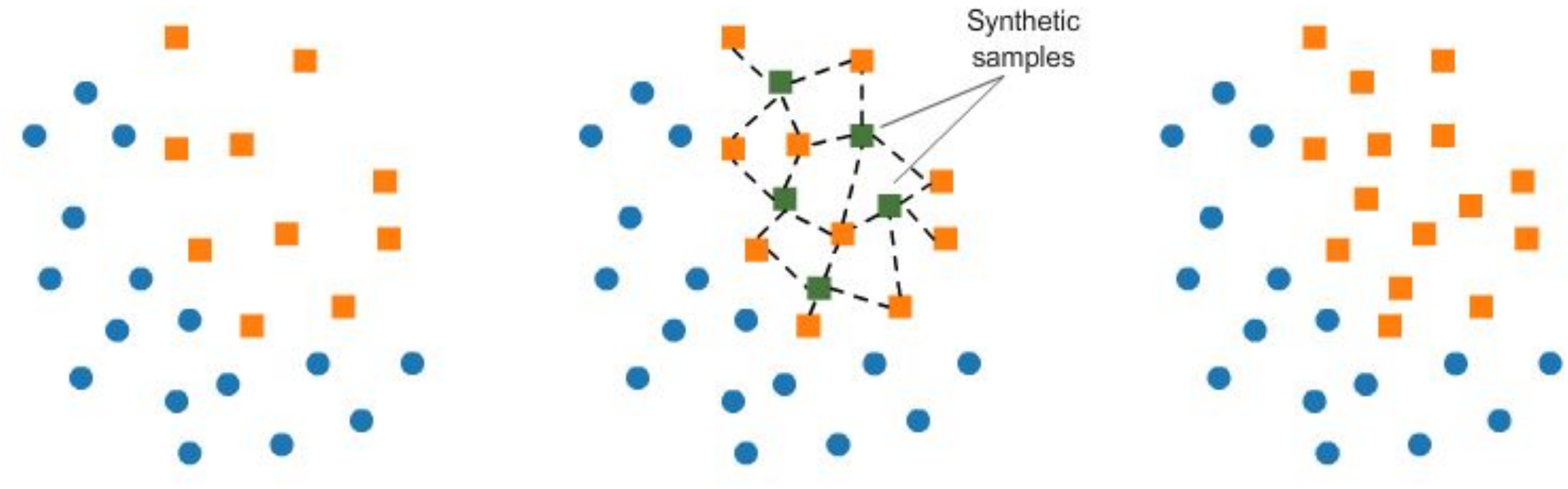
## OverSampling (UpSampling)

Oversampling is the process of randomly duplicating observations from the minority class.



## SMOTE (Synthetic Minority OverSampling Technique)

SMOTE (Synthetic Minority Oversampling Technique) first randomly picks a point from the minority class and computes the k-nearest neighbors for this point. Then the synthetic points are added between the chosen point and its neighbors.



## Quiz 14

Choose the correct statement(s).

- OverSampling leads to reduction in majority class observations.
- Data articulation in SMOTE is done in the proximity of a minority class data point.
- UpSampling process increases the instances of minority class.
- All of the above

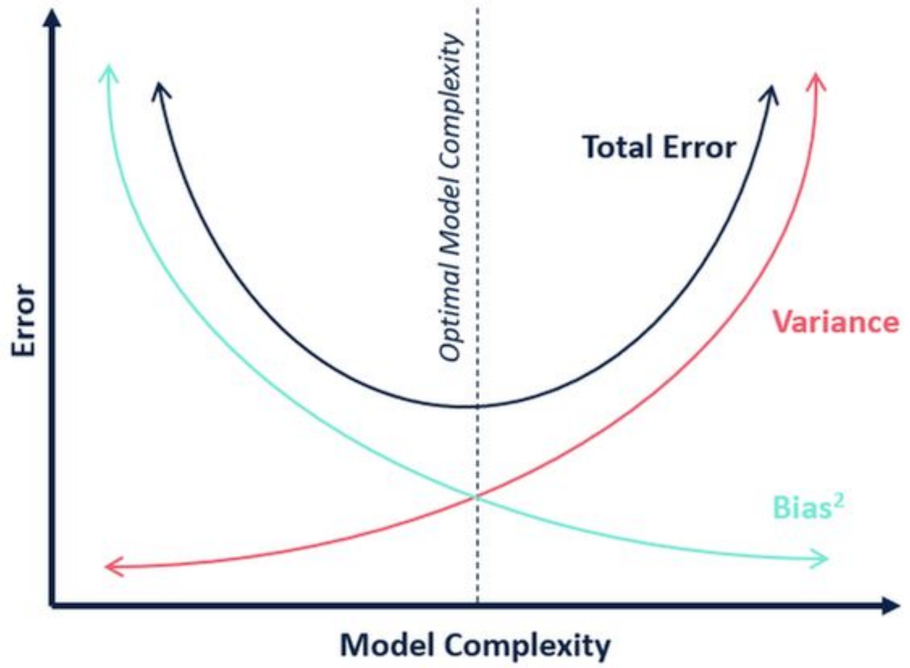
## Bias-Variance Trade-Off

The error of a Supervised model can be further decomposed as

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

Bias $\uparrow$ $\Rightarrow$ Error $\uparrow$

Variance $\uparrow$ $\Rightarrow$ Error $\uparrow$

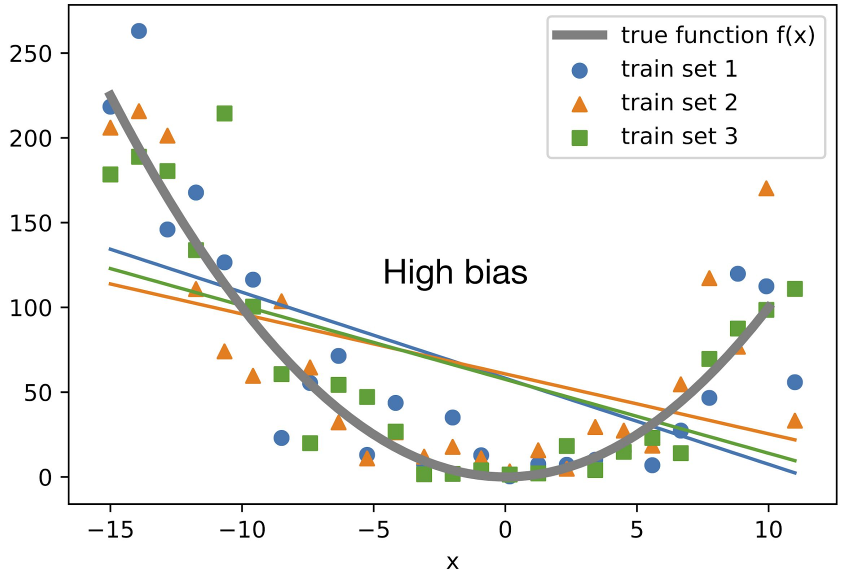


***What if we train the model on different subsets of data?***

## Bias

Bias is the difference between the average predicted value of the model and the actual value.

Mathematically, Bias = $E[\hat{f}(x)] - f(x)$



Model with high bias pays very little attention to the training data (oversimplifies the model).

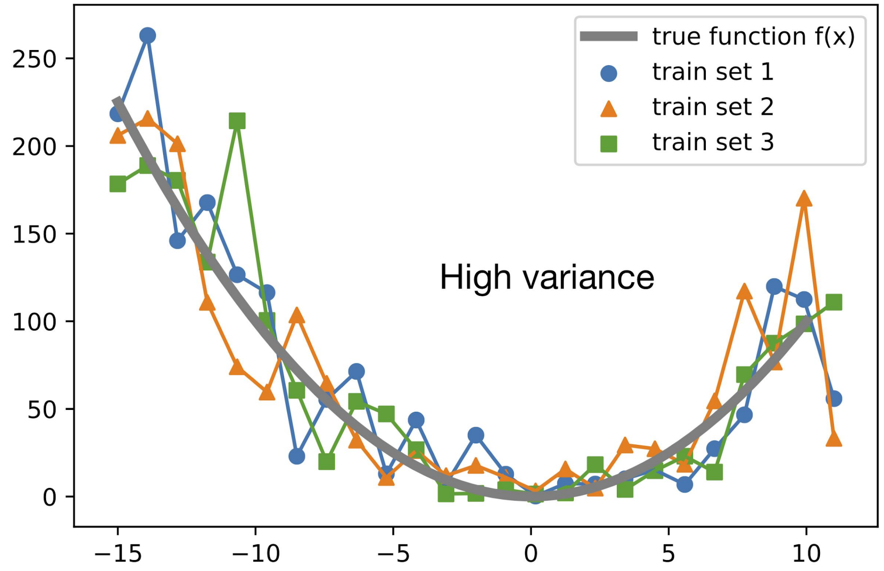
A high bias model leads to high error on both training and test data.

***What if we train the model on different subsets of data?***

## Variance

Variance is the variations in model predictions as training data changes.

Mathematically, Variance = $E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right]$

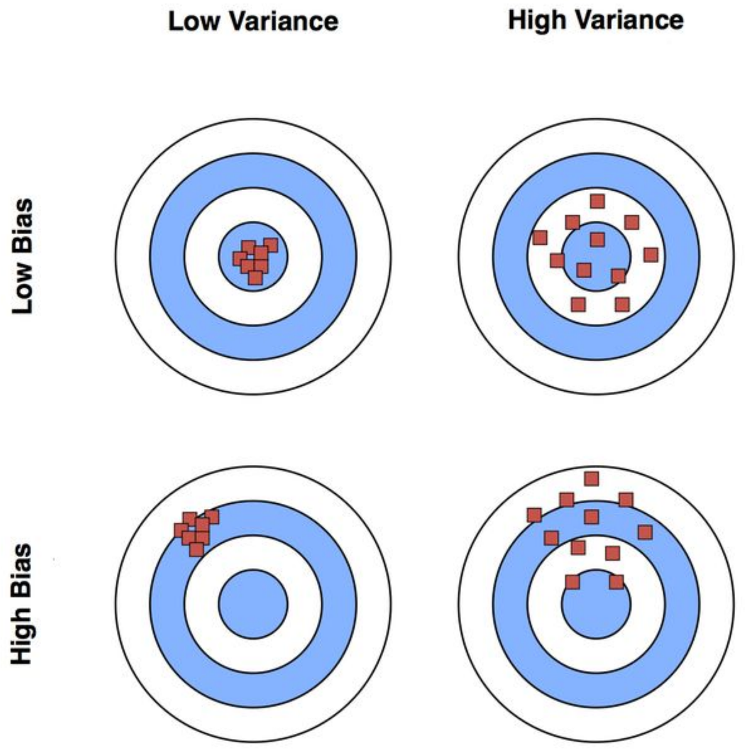


Model with high variance pays a lot of attention to the training data and does not generalize well on the test data.

A high variance model performs very well on training data but provides high error on test data.

## Bias-Variance using Bull’s eye diagram

- Here, center of the target represents a model that predicts all the points correctly.
- As we move away from the center, model starts making more and more wrong predictions.
- A model with low bias and high variance predicts points that are often around the center but far from each other.
- A model with high bias and low variance predicts points that are closer to each other but far from the center.
- A model with high bias and high variance predicts points that are far from the center and also far from each other.

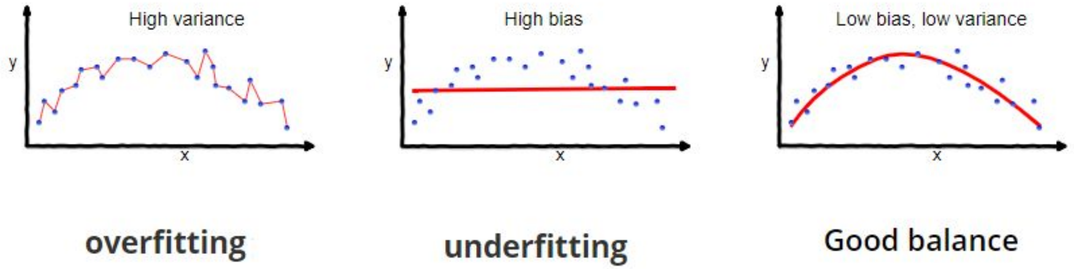


## Train-Test Error Combinations

| Train Error | Test Error | Causes |
|---|---|---|
| Low | Low | Good/perfect trade off |
| Low | High | High Variance (Overfitting) |
| High | Low | Uncommon (need to be examined) |
| High | High | High Bias (Underfitting) |

## Overfitting, Underfitting and Good fit

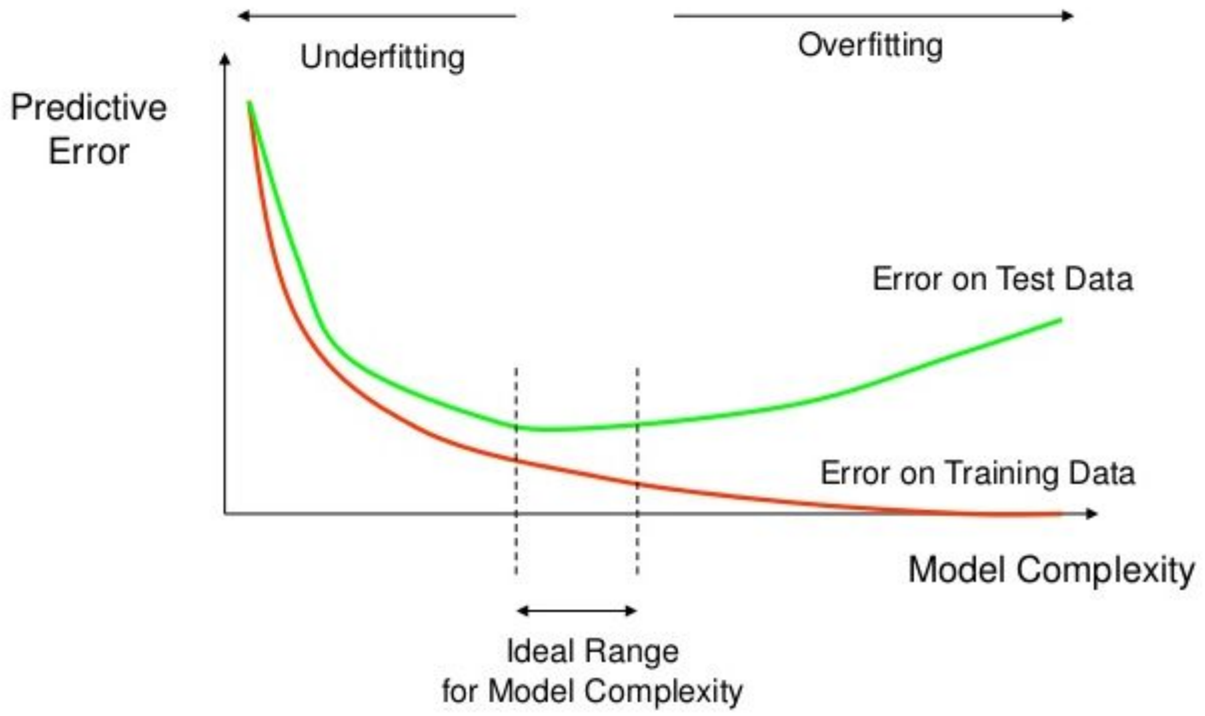
- **Overfitting -** Case where a ML model works well on the training data but does not generalizes well to test/unseen data.
- **Underfitting** - Case where a ML model neither works well on the training data nor generalizes well to test/unseen data.
- **Good fit -** Case where a ML model works well on the training data and also generalizes well to test/unseen data.



## Overfitting, Underfitting and Good fit [Contd.]

| | Underfitting | Good Fit | Overfitting |
|---|---|---|---|
| Train-Test error | High training error, High test error | Low training error, Low test error | Low training error, High test error |
| Complexity | Over Simplistic model | Intermediate | Highly complex model |
| Bias-Variance combination | High Bias, Low Variance | Low Bias, Low Variance | Low Bias, High Variance |
| Generalization power | Poor | Good | Poor |
| Capturing power | Unable to capture the underlying pattern/trend in the data. | Captures the underlying pattern/trend in the data being less prone to noise. | Captures noise (random fluctuations) along with the underlying pattern in the data. |
| Decision Surface | Model does not make any good attempt for decision surface. | Smooth | Non-smooth |

## Overfitting, Underfitting and Good fit [Contd.]



## Classification Algos Overfitting, Underfitting case

| Algorithm | Overfitting | Underfitting |
|---|---|---|
| kNN | Smaller values of k | Higher value of k |
| Decision Tree | Max_depth ↑ | Max_depth ↓ |
| SVM | C ↑ | C ↓ |
| Logistic Regression | C = (1 / ƛ) ↑ | C = (1 / ƛ) ↓ |

## Quiz 15

Choose the correct statement(s) about Overfitting.

- An overfitted model is more sensitive to noisy data points.
- An overfitted model has high bias and high variance.
- An overfitted model attempts to make a non-smooth decision surface by paying more attention to noisy data points along with ordinary data points.
- All of the above

## Ways to deal with Overfitting, Underfitting

### Underfitting

- Increase model complexity
- Increase number of features, Perform feature engineering
- Trying different ML algorithms, Hyperparameter tuning

### Overfitting

- Reduce model complexity
- Regularization
- Resampling techniques like k-fold cross validation
- Increase training data
- Ensemble methods

## L1 and L2 Regularization

- Regularization is the process of penalizing complex models by adding some penalty to the loss term.
- There are mainly two types of regularization techniques - L1 regularization and L2 regularization.

**L1 regularization (Lasso Regression)**

- Lasso Regression adds absolute value of magnitude of coefficient as penalty term to the loss function.

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$

**L2 regularization (Ridge Regression)**

- Ridge regression adds squared magnitude of coefficient as penalty term to the loss function.

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

The **key difference** between these techniques is that Lasso shrinks the least important features by setting the coefficients to zero thus, helps in feature selection.