

6CS4-02:Machine Learning**Credit: 3**
3L+0T+0P**Max. Marks: 150(IA:30, ETE:120)****End Term Exam: 3 Hours**

SN	Contents	Hours
1	Introduction: Objective, scope and outcome of the course.	01
2	Supervised learning algorithm: Introduction, types of learning, application, Supervised learning: Linear Regression Model, Naive Bayes classifier Decision Tree, K nearest neighbor, Logistic Regression, Support Vector Machine, Random forest algorithm	09
3	Unsupervised learning algorithm: Grouping unlabelled items using k-means clustering, Hierarchical Clustering, Probabilistic clustering, Association rule mining, Apriori Algorithm, f-p growth algorithm, Gaussian mixture model.	08
4	Introduction to Statistical Learning Theory , Feature extraction - Principal component analysis, Singular value decomposition. Feature selection – feature ranking and subset selection, filter, wrapper and embedded methods, Evaluating Machine Learning algorithms and Model Selection.	08
5	Semi supervised learning, Reinforcement learning: Markov decision process (MDP), Bellman equations, policy evaluation using Monte Carlo, Policy iteration and Value iteration, Q-Learning, State-Action-Reward-State-Action (SARSA), Model-based Reinforcement Learning.	08
6	Recommended system , Collaborative filtering, Content-based filtering Artificial neural network, Perceptron, Multilayer network, Backpropagation, Introduction to Deep learning.	08
	Total	42

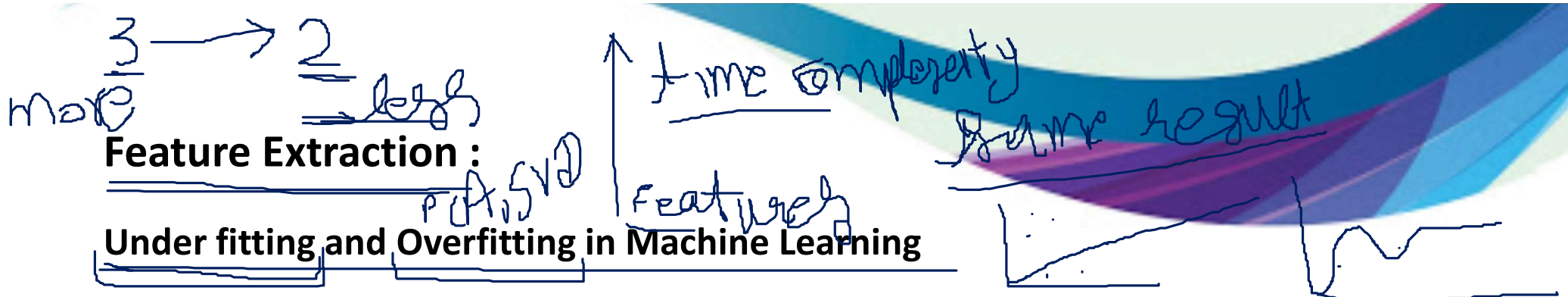


Machine Learning

Unit-3

Introduction to Statistical Learning Theory:

B K BIRLA INSTITUTE OF ENGINEERING & TECHNOLOGY, PILANI



Under fitting and Overfitting in Machine Learning

A machine learning model. A model is said to be a good machine learning model if it generalizes any new input data from the problem domain in a proper way. This helps us to make predictions in the future data, that data model has never seen.

we want to check how well our machine learning model learns and generalizes to the new data. For that we have overfitting and under fitting, which are majorly responsible for the poor performances of the machine learning algorithms.

Before diving further let's understand two important terms:

Bias – Assumptions made by a model to make a function easier to learn.

Variance – If you train your data on training data and obtain a very low error, upon changing the data and then training the same previous model you experience high error, this is variance.



Underfitting:

A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data. (It's just like trying to fit undersized pants!) Underfitting destroys the accuracy of our machine learning model. Its occurrence simply means that our model or the algorithm does not fit the data well enough. It usually happens when we have less data to build an accurate model and also when we try to build a linear model with a non-linear data. In such cases the rules of the machine learning model are too easy and flexible to be applied on such minimal data and therefore the model will probably make a lot of wrong predictions. Underfitting can be avoided by using more data and also reducing the features by feature selection.



Techniques to reduce underfitting :

- ✓ 1. Increase model complexity
- ✓ 2. Increase number of features, performing feature engineering
- ✓ 3. Remove noise from the data.
- ✓ 4. Increase the number of epochs or increase the duration of training to get better results.

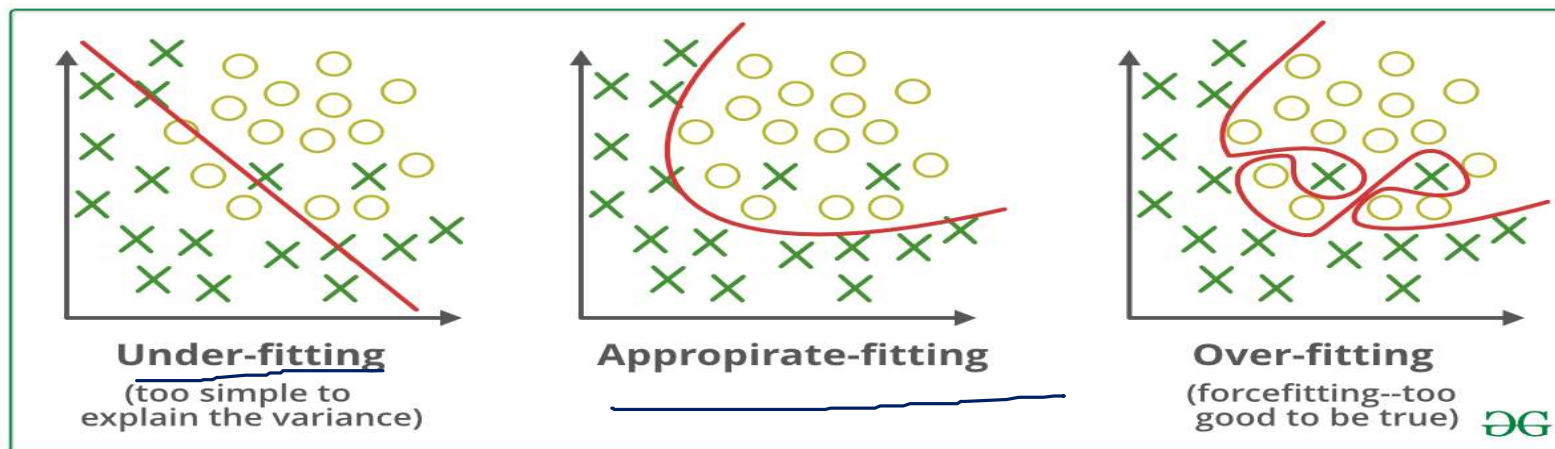


Overfitting:

A statistical model is said to be overfitted, when we train it with a lot of data (*just like fitting ourselves in oversized pants!*). When a model gets trained with so much of data, it starts learning from the noise and inaccurate data entries in our data set. Then the model does not categorize the data correctly, because of too many details and noise. The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic models. A solution to avoid overfitting is using a linear algorithm if we have linear data or using the parameters like the maximal depth if we are using decision trees.

Techniques to reduce overfitting :

- ✓ 1. Increase training data.
- ✓ 2. Reduce model complexity.
- ✓ 3. Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).
- ✓ 4. Ridge Regularization and Lasso Regularization
- ✓ 5. Use dropout for neural networks to tackle overfitting.





Principal Component Analysis (PCA)

PCA is a statistical procedure that uses an orthogonal transformation which converts a set of correlated variables to a set of uncorrelated variables. PCA is a most widely used tool in exploratory data analysis and in machine learning for predictive models. Moreover, PCA is an unsupervised statistical technique used to examine the interrelations among a set of variables. It is also known as a general factor analysis where regression determines a line of best fit.

PCA



Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process.

So to sum up, the idea of PCA is simple — reduce the number of variables of a data set, while preserving as much information as possible.



STEP 1: STANDARDIZATION

The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis.


More specifically, the reason why it is critical to perform standardization prior to PCA, is that the latter is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges (For example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1), which will lead to biased results. So, transforming the data to comparable scales can prevent this problem.

0-100	0-100
-------	-------

STEP 2: COVARIANCE MATRIX COMPUTATION

The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them. Because sometimes, variables are highly correlated in such a way that they contain redundant information. So, in order to identify these correlations, we compute the covariance matrix.

$$\begin{aligned}\text{Cov}(A) &= \begin{bmatrix} \frac{\sum (x_i - \bar{X})(x_i - \bar{X})}{N} & \frac{\sum (x_i - \bar{X})(y_i - \bar{Y})}{N} \\ \frac{\sum (x_i - \bar{X})(y_i - \bar{Y})}{N} & \frac{\sum (y_i - \bar{Y})(y_i - \bar{Y})}{N} \end{bmatrix} \\ &= \begin{bmatrix} \text{Cov}(X, X) & \text{Cov}(Y, X) \\ \text{Cov}(X, Y) & \text{Cov}(Y, Y) \end{bmatrix}\end{aligned}$$



Since the covariance of a variable with itself is its variance ($\text{Cov}(a,a)=\text{Var}(a)$), in the main diagonal (Top left to bottom right) we actually have the variances of each initial variable. And since the covariance is commutative ($\text{Cov}(a,b)=\text{Cov}(b,a)$), the entries of the covariance matrix are symmetric with respect to the main diagonal, which means that the upper and the lower triangular portions are equal.

What do the covariances that we have as entries of the matrix tell us about the correlations between the variables?

It's actually the sign of the covariance that matters :

- if positive then : the two variables increase or decrease together (correlated)
- if negative then : One increases when the other decreases (Inversely correlated)

X	$x - \bar{x}$	$(x - \bar{x})(x - \bar{x})$
2.5	0.69	0.4761
1.5	-1.31	1.7161
2.2	0.39	0.1521
1.9	0.09	0.0081
3.1	1.29	1.6641
2.3	0.49	0.2401
2	0.19	0.0361
1	-0.81	0.6561
1.5	-0.31	0.0961
1.1	-0.71	0.5041
		Sum = 5.548

$$\text{Cov}(x, x) = \sum_{i=1}^n \frac{(x_i - \bar{x})(x_i - \bar{x})}{n-1}$$

$$\Rightarrow \frac{5.548}{9} = 0.6165$$

Y	$y - \bar{y}$	$(y - \bar{y})(y - \bar{y})$
2.4	0.49	0.2401
0.7	-1.21	1.4641
2.9	0.99	0.9801
2.2	0.29	0.0841
3.0	1.09	1.1881
2.7	0.79	0.6241
1.6	-0.31	0.0961
1.1	-0.81	0.6561
1.6	-0.31	0.0961
0.9	-1.01	1.0201
		Sum = 6.448

$$\text{Cov}(y, y) = \sum_{i=1}^n \frac{(y_i - \bar{y})(y_i - \bar{y})}{n-1}$$

$$\Rightarrow \frac{6.448}{9} = 0.7165$$

X	Y	$(X - \bar{X})$	$(Y - \bar{Y})$	$(X - \bar{X})(Y - \bar{Y})$
2.5	2.4	0.69	0.49	0.338
0.5	0.7	-1.31	-1.21	1.585
2.2	2.9	0.39	0.99	0.386
1.9	2.2	0.09	0.29	0.026
3.1	3.0	1.29	1.09	1.406
2.3	2.7	0.49	0.79	0.387
2	1.6	-0.19	-0.79	0.058
1	1.1	-0.81	-0.31	0.656
1.5	1.6	-0.31	-0.81	0.096
1.1	0.9	-0.71	-1.01	0.717
Sum =				5.539

$$\underline{\text{Cov}(X, Y)} = \underline{\text{Cov}(Y, X)} = \frac{5.539}{9} = \underline{0.6154}$$

for $\lambda_1 = 0.4908$

$$\begin{bmatrix} .5674 & .6154 \\ 0.6154 & 0.6674 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = 0$$

$$\begin{aligned} .5674x_1 + .6154y_1 &= 0 \rightarrow y_1 = \frac{-.5674}{0.6154}x_1 \\ 0.6154x_1 + .6674y_1 &= 0 \\ x^2 + y^2 &= 1 \end{aligned}$$

$$y_1 = -.9220x_1$$

$$x_1^2 + 85008x_1^2 = 1$$

$$x_1 = .7351$$

$$y_1 = -.9220$$

$$\lambda_2 = \underline{1.2840}$$

A

$$\begin{bmatrix} -0.6675 & 0.6154 \\ 0.6154 & 0.5675 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = 0$$

$$-0.6675x_2 + 0.6154y_2 = 0$$

$$0.6154x_2 - 0.5675y_2 = 0$$

$$y_2 = \frac{0.6675}{0.6154}x_2 \Rightarrow 1.0846x_2$$


$$x_2^2 + 1.1764x_2^2 = 1$$

$$x_2^2 = \frac{1}{2.1764} \Rightarrow x_2 = 0.6778$$

$$y_2 = 0.7351$$

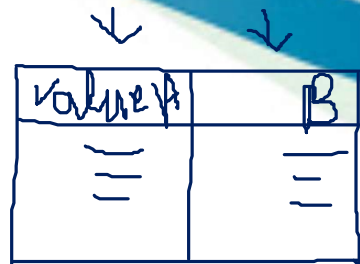
$$\begin{bmatrix} 0.7351 & 0.6778 \\ 0.6778 & 0.7351 \end{bmatrix}$$





If we rank the eigenvalues in descending order, we get $\lambda_2 > \lambda_1$, which means that the eigenvector that corresponds to the first principal component (PC1) is v_1 and the one that corresponds to the second component (PC2) is v_2 .

After having the principal components, to compute the percentage of variance (information) accounted for by each component, we divide the eigenvalue of each component by the sum of eigenvalues. If we apply this on the example above, we find that PC1 and PC2 carry respectively 96% and 4% of the variance of the data.



Singular Value Decomposition


Linear Algebra makes the core foundation for Machine Learning algorithms ranging from simple linear regressions to Deep Neural Networks. The major reason for that is the set of data can be represented using a 2-D matrix where the column represents the features and the row represents the different sample data points. Having said that, the Matrix computations using all of the values in a matrix are sometimes redundant or rather computationally expensive. We need to represent the matrix in a form such that, the most important part of the matrix which is needed for further computations could be extracted easily. That's where the **Singular Value Decomposition(SVD)** comes into play.



SVD states that **any** matrix A can be factorized as

$$\underline{A} = \underline{U} \underline{S} \underline{V}^T$$

where U and V are orthogonal matrices with orthonormal eigenvectors chosen from $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ respectively. S is a diagonal matrix with r elements equal to the root of the positive eigenvalues of $\mathbf{A}\mathbf{A}^T$ or $\mathbf{A}^T\mathbf{A}$ (both matrices have the same positive eigenvalues anyway). The diagonal elements are composed of singular values.




$$\begin{pmatrix} \sqrt{\lambda_1} & & & & \\ & \sqrt{\lambda_2} & & & \\ & & \ddots & & \\ & & & \sqrt{\lambda_r} & \\ & & & & \ddots \\ & & & & & 0 \end{pmatrix} \equiv \begin{pmatrix} \sigma_1 & & & & \\ & \sigma_2 & & & \\ & & \ddots & & \\ & & & \sigma_r & \\ & & & & \ddots \\ & & & & & 0 \end{pmatrix}$$

σ_2 : singular value

i.e. an $m \times n$ matrix can be factorized as:

$$\underline{A_{m \times n}} = \underline{U_{m \times m}} \underline{S_{m \times n}} \underline{V_{n \times n}^T}$$



We can arrange eigenvectors in different orders to produce U and V . To standardize the solution, we order the eigenvectors such that vectors with higher eigenvalues come before those with smaller values.

$$\begin{array}{c} \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_m \\ \left(\begin{array}{ccc} & & \\ | & & | \\ \hline u_1 & \dots & u_m \\ | & & | \end{array} \right) \end{array}$$

Comparing to eigen decomposition, SVD works on non-square matrices. U and V are invertible for any matrix in SVD and they are orthonormal which we love it. Without proof here, we also tell you that singular values are more numerical stable than eigenvalues.

SVD computation example

Example: Find the SVD of A , $U\Sigma V^T$, where $A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$.

First we compute the singular values σ_i by finding the eigenvalues of AA^T .

$$AA^T = \begin{pmatrix} 17 & 8 \\ 8 & 17 \end{pmatrix}.$$

The characteristic polynomial is $\det(AA^T - \lambda I) = \lambda^2 - 34\lambda + 225 = (\lambda - 25)(\lambda - 9)$, so the singular values are $\sigma_1 = \sqrt{25} = 5$ and $\sigma_2 = \sqrt{9} = 3$.

Now we find the right singular vectors (the columns of V) by finding an orthonormal set of eigenvectors of $A^T A$. It is also possible to proceed by finding the left singular vectors (columns of U) instead. The eigenvalues of $A^T A$ are 25, 9, and 0, and since $A^T A$ is symmetric we know that the eigenvectors will be orthogonal.

For $\lambda = 25$, we have

$$A^T A - 25I = \begin{pmatrix} -12 & 12 & 2 \\ 12 & -12 & -2 \\ 2 & -2 & -17 \end{pmatrix}$$

which row-reduces to $\begin{pmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$. A unit-length vector in the kernel of that matrix

is $v_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{pmatrix}$.

For $\lambda = 9$ we have $A^T A - 9I = \begin{pmatrix} 4 & 12 & 2 \\ 12 & 4 & -2 \\ 2 & -2 & -1 \end{pmatrix}$ which row-reduces to $\begin{pmatrix} 1 & 0 & -\frac{1}{4} \\ 0 & 1 & \frac{1}{4} \\ 0 & 0 & 0 \end{pmatrix}$.

A unit-length vector in the kernel is $v_2 = \begin{pmatrix} 1/\sqrt{18} \\ -1/\sqrt{18} \\ 4/\sqrt{18} \end{pmatrix}$.

For the last eigenvector, we could compute the kernel of $A^T A$ or find a unit vector perpendicular to v_1 and v_2 . To be perpendicular to $v_1 = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ we need $-a = b$.

Then the condition that $v_2^T v_3 = 0$ becomes $2a/\sqrt{18} + 4c/\sqrt{18} = 0$ or $-a = 2c$. So $v_3 = \begin{pmatrix} a \\ -a \\ -a/2 \end{pmatrix}$ and for it to be unit-length we need $a = 2/3$ so $v_3 = \begin{pmatrix} 2/3 \\ -2/3 \\ -1/3 \end{pmatrix}$.


So at this point we know that

$$A = U \Sigma V^T = U \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{pmatrix}.$$

Finally, we can compute U by the formula $\sigma u_i = A v_i$, or $u_i = \frac{1}{\sigma} A v_i$. This gives

$$U = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}. \text{ So in its full glory the SVD is:}$$

$$A = U \Sigma V^T = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{pmatrix}.$$



SVD is basically a matrix factorization technique, which decomposes any matrix into 3 generic and familiar matrices. It has some cool applications in Machine Learning and Image Processing. To understand the concept of Singular Value Decomposition the knowledge on **eigenvalues** and **eigenvectors** is essential. If you have a pretty good understanding on eigenvalues and eigenvectors, scroll down a bit to experience the Singular Value Decomposition.

Singular Value Decomposition:-

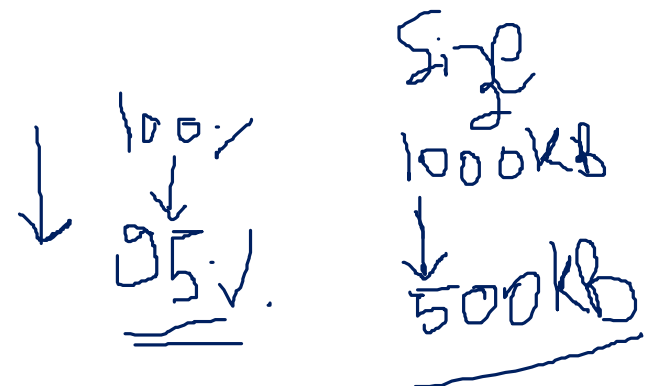
30-

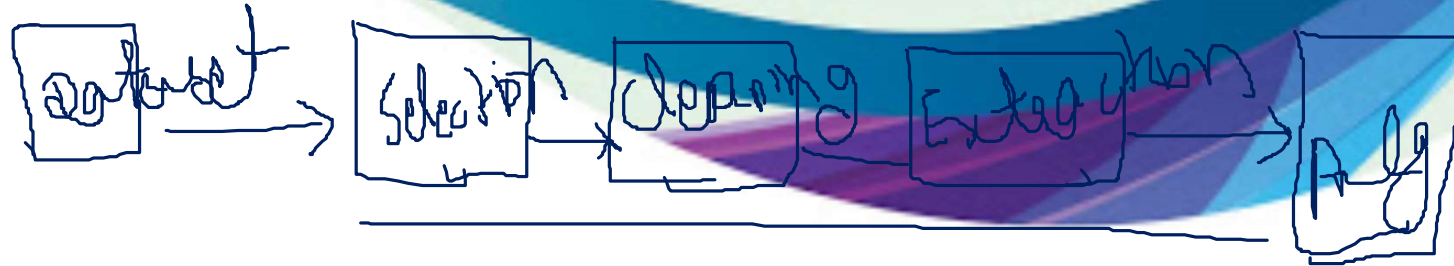
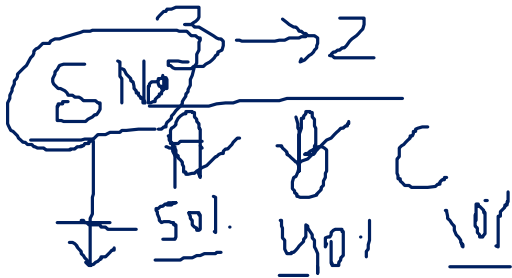
1. SVD is decomposition of a matrix A into 3 Matrices
- U, S, V
2. S is the diagonal matrix of Singular Values. Think of
Singular value as the importance values of different
features in the matrix.
3. The rank of a matrix is a measure of the unique
information stored in a matrix. Higher the
rank, more the information.
4. Eigen Vectors of a matrix are directions of
maximum spread or variance of data.

SVD for Image Compression:

We have smartphone. We love clicking images
with our smart phone cameras and saving
random photos off the web. And then
one day no space! Image compression
helps. dear

it minimize the size of an image in bytes to
an acceptable level of quality. This means that
you are able to store more image in the same
disk space as compared to before





Feature Selection:

Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model. The data features that you use to train your machine learning models have a huge influence on the performance you can achieve.

Irrelevant or partially relevant features can negatively impact model performance.

Feature selection and Data cleaning should be the first and most important step of your model designing.

In this post, you will discover feature selection techniques that you can use in Machine Learning.

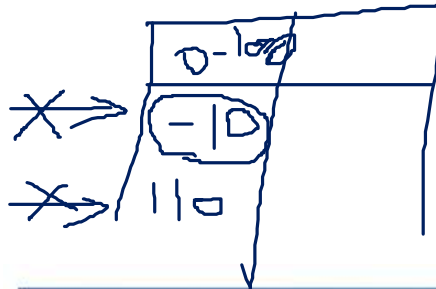
Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in.

Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.



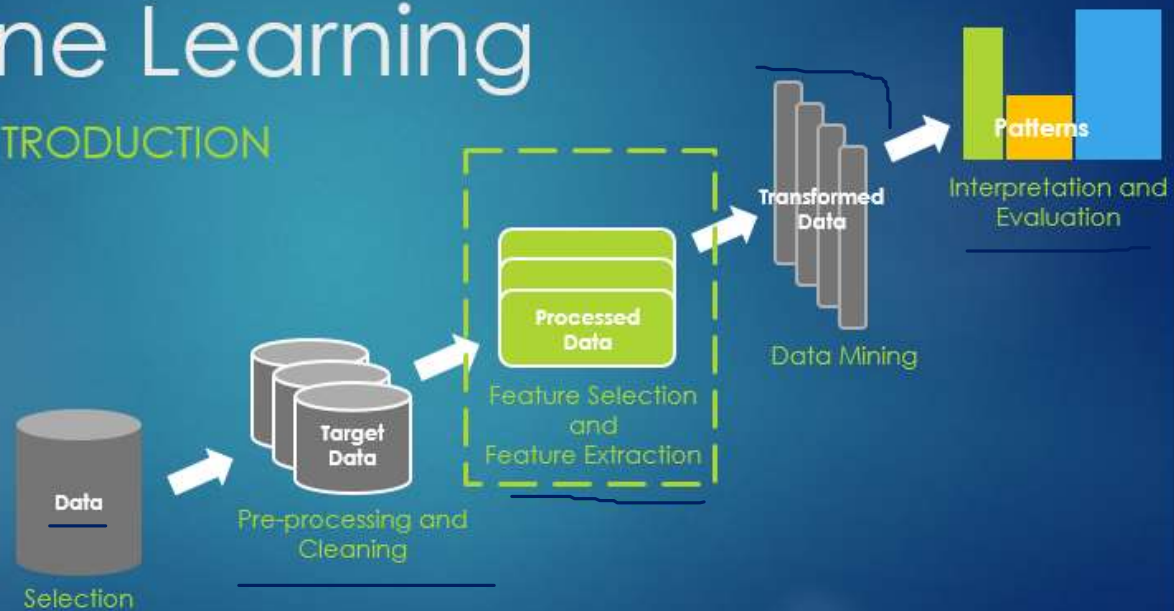
Feature Selection

- ✓ How to select features and what are Benefits of performing feature selection before modeling your data?
- ✓ Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise.
- ✓ Improves Accuracy: Less misleading data means modeling accuracy improves.
- ✓ Reduces Training Time: fewer data points reduce algorithm complexity and algorithms train faster.



Feature Selection in Machine Learning

SERIES 1: AN INTRODUCTION





Variable Ranking :

Variable Ranking is the process of ordering the features by the value of some scoring function, which usually measures feature-relevance.

Resulting set: The score $S(f_i)$ is computed from the training data, measuring some criteria of feature f_i . By convention a high score is indicative for a valuable (relevant) feature.

A simple method for *feature selection* using **variable ranking** is to select the k highest ranked features according to S . This is usually not optimal, but often preferable to other, more complicated methods. It is computationally efficient — only calculation and sorting of n scores.

①

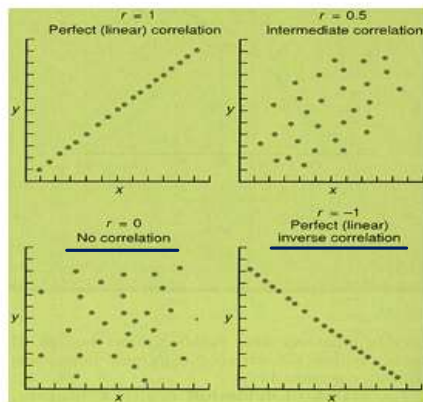
RANKING CRITERIA

CORRELATION

The higher the correlation between the feature and the target, the higher the score.

Pearson Correlation Coefficient

$$R(f_i, y) = \frac{\text{cov}(f_i, y)}{\sqrt{\text{var}(f_i) \text{var}(y)}}$$



-0.424

②

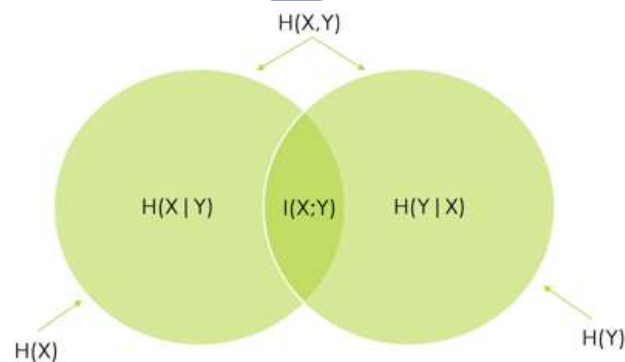
INFORMATION THEORETIC CRITERIA

Mutual information can also detect non-linear dependencies among variables

But harder to estimate than correlation

It is a measure for "how much information (in terms of entropy) two random variables share"

entropy



Pearson Correlation

The Pearson correlation coefficient is denoted by the letter " r ". The formula for Pearson correlation coefficient r is given by:

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n \sum x^2 - (\sum x)^2][n \sum y^2 - (\sum y)^2]}}$$

Where,

r = Pearson correlation coefficient

x = Values in the first set of data

y = Values in the second set of data

n = Total number of values.

Solved Example

Question: Marks obtained by 5 students in algebra and trigonometry as given below:

<u>Algebra</u>	15	16	12	10	8
<u>Trigonometry</u>	18	11	10	20	17

Calculate the Pearson correlation coefficient.

Pearson Correlation

Solution:

Construct the following table:

<u>x</u>	<u>y</u>	x^2	y^2	xy
15	18	225	324	270
16	11	256	121	176
12	10	144	100	120
10	20	100	400	200
8	17	64	289	136
$\sum x = 61$	$\sum y = 76$	$\sum x^2 = 789$	$\sum y^2 = 1234$	$\sum xy = 902$

Formula for Pearson correlation coefficient is given by:

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n \sum x^2 - (\sum x)^2][n \sum y^2 - (\sum y)^2]}}$$

$$r = \frac{5 \times 902 - 61 \times 76}{\sqrt{[5 \times 789 - (61)^2][5 \times 1234 - (76)^2]}}$$

$$\underline{\underline{r = -0.424}}$$



Feature Subset Selection

The Goal of Feature Subset Selection is to find the optimal feature subset. Feature Subset Selection Methods can be classified into three broad categories

- ✓ •Filter Methods
- ✓ •Wrapper Methods
- ✓ •Embedded Methods



For **Feature Subset Selection** you'd need:

- ✓ • A measure for assessing the goodness of a feature subset (scoring function)
- ✓ • A strategy to search the space of possible feature subsets
- ✓ • Finding a minimal optimal feature set for an arbitrary target concept is hard. It would need Good Heuristics.



Filter Methods

- In this method, select subsets of variables as a pre-processing step, independently of the used classifier
- It would be worthwhile to note that **Variable Ranking-Feature Selection** is a Filter Method.





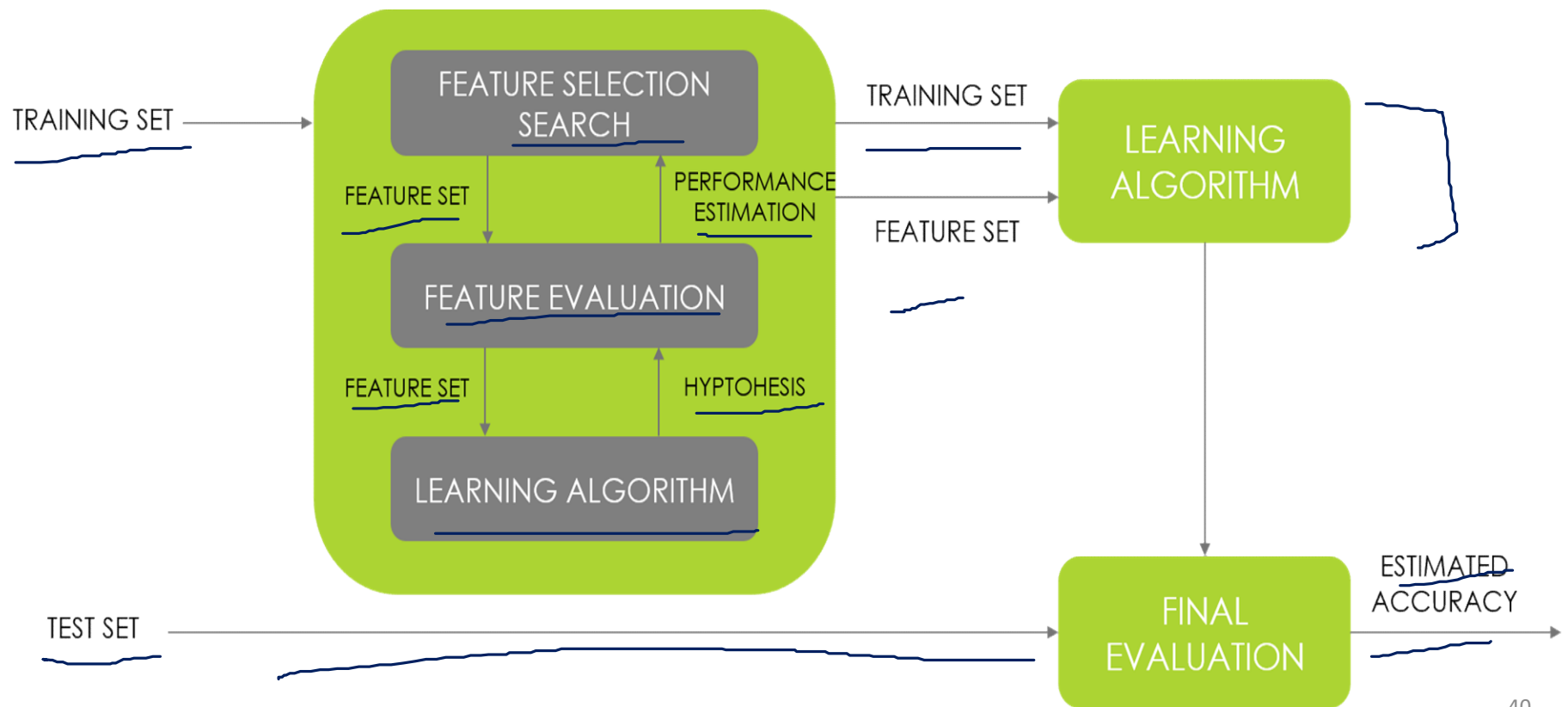
Key features of Filter Methods for **Feature Subset Selection**:

- ✓ •Filter Methods are usually fast
- ✓ •Filter Methods provide generic selection of features, not tuned by given learner (universal)
- ✓ •Filter Methods are also often criticized (feature set not optimized for used classifier)
- ✓ •Filter Methods are sometimes used as a pre-processing step for other methods.



Wrapper Methods

- ✓ • In Wrapper Methods, the Learner is considered a black-box. Interface of the black-box is used to score subsets of variables according to the predictive power of the learner when using the subsets.
- ✓ • Results vary for different learners
- ✓ • One needs to define: – how to search the space of all possible variable subsets ?– how to assess the prediction performance of a learner ?





Embedded Methods

- ✓ • Embedded Methods are specific to a given learning machine
- ✓ • Performs variable selection (implicitly) in the process of training
- ✓ • E.g. WINNOW-algorithm (linear unit with multiplicative updates).



Evaluating machine learning algorithms and model selection

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and over fitted models. There are two methods of evaluating models in data science, Hold-Out and CrossValidation. To avoid overfitting, both methods use a test set (not seen by the model) to evaluate model performance.

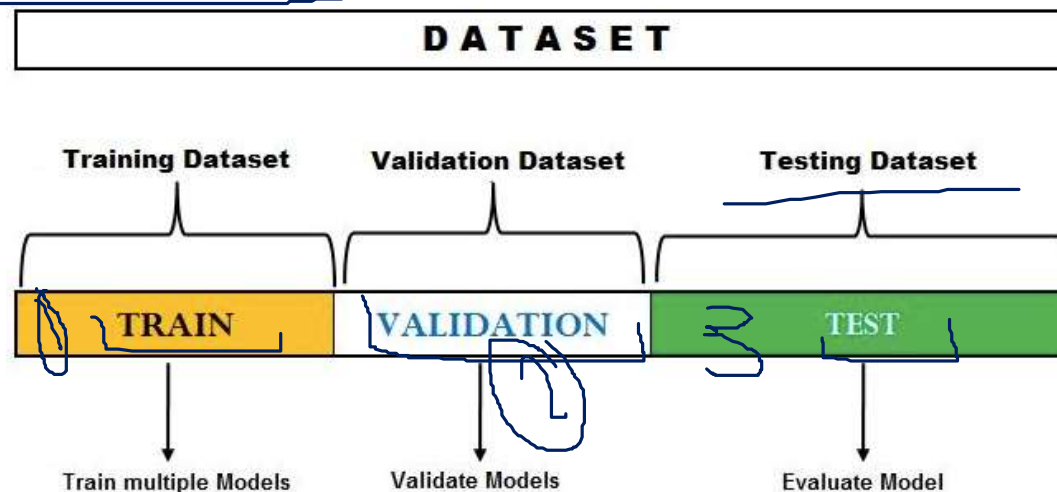
Hold-Out

In this method, the mostly large dataset is randomly divided to three subsets:

Training set is a subset of the dataset used to build predictive models.

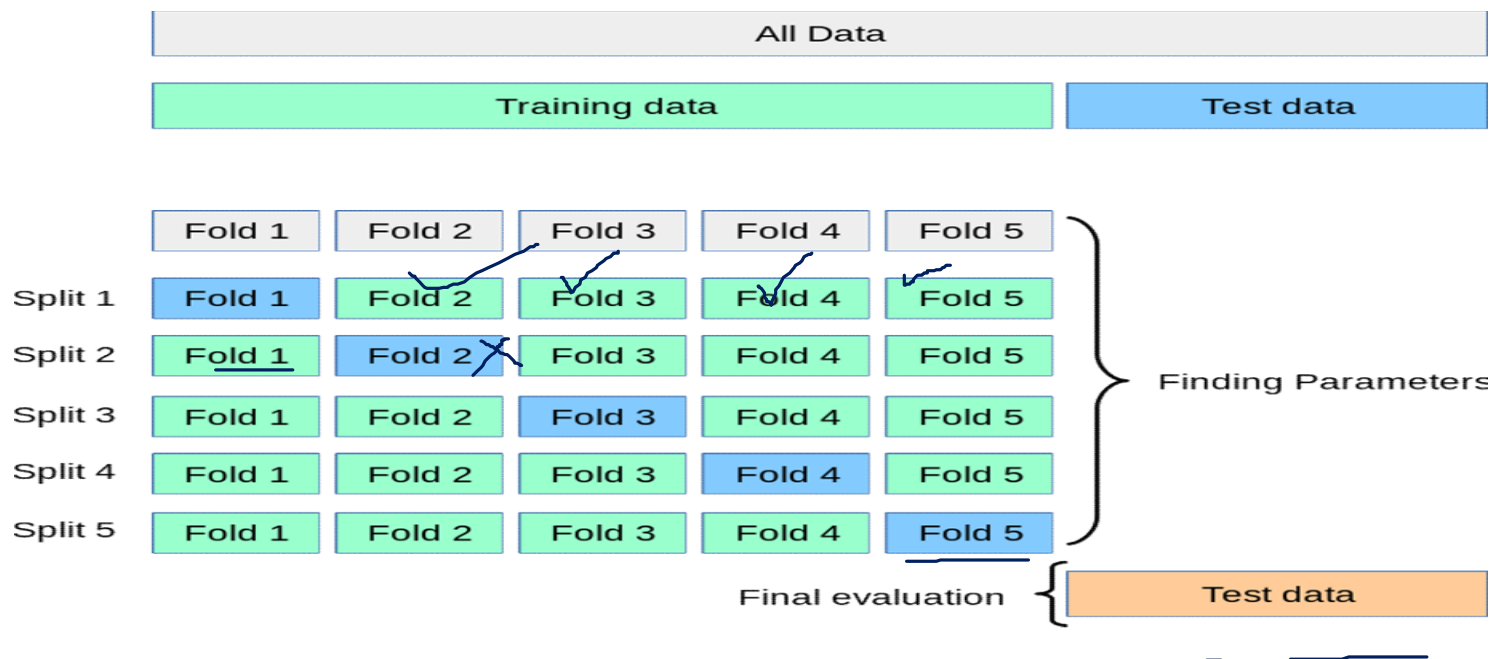
Validation set is a subset of the dataset used to assess the performance of model built in the training phase. It provides a test platform for fine tuning model's parameters and selecting the best-performing model. Not all modeling algorithms need a validation set.

Test set or unseen examples is a subset of the dataset to assess the likely future performance of a model. If a model fit to the training set much better than it fits the test set, overfitting is probably the cause.



Cross-Validation

When only a limited amount of data is available, to achieve an unbiased estimate of the model performance we use k-fold cross-validation. In k-fold cross-validation, we divide the data into k subsets of equal size. We build models k times, each time leaving out one of the subsets from training and use it as the test set. If k equals the sample size, this is called "leave-one-out".





Model Evaluation - Classification

Confusion Matrix

A confusion matrix shows the number of correct and incorrect predictions made by the classification model compared to the actual outcomes (target value) in the data. The matrix is $N \times N$, where N is the number of target values (classes). Performance of such models is commonly evaluated using the data in the matrix. The following table displays a 2×2 confusion matrix for two classes (Positive and Negative).

Confusion Matrix		Target				
		Positive	Negative			
Model	Positive	a	b	Positive Value	Predictive	$a/(a+b)$
	Negative	c	d	Negative Value	Predictive	$d/(c+d)$
		Sensitivity	Specificity	Accuracy = $(a+d)/(a+b+c+d)$		
		$a/(a+c)$	$d/(b+d)$			

- **Accuracy** : the proportion of the total number of predictions that were correct.
- **Positive Predictive Value or Precision** : the proportion of positive cases that were correctly identified.
- **Negative Predictive Value** : the proportion of negative cases that were correctly identified.
- **Sensitivity or Recall** : the proportion of actual positive cases which are correctly identified.
- **Specificity** : the proportion of actual negative cases which are correctly identified.

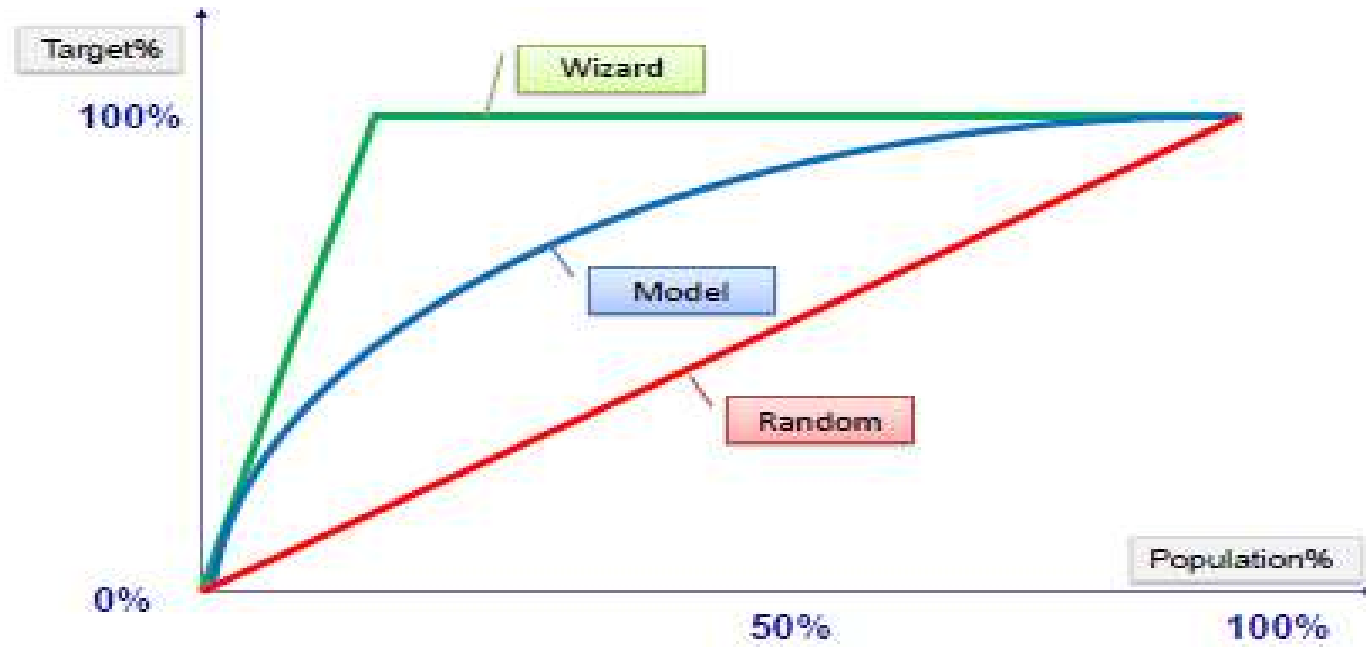
Example:

Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	70	20	Positive Predictive Value	0.78
	Negative	30	80	Negative Predictive Value	0.73
		Sensitivity	Specificity	Accuracy = 0.75	
		0.70	0.80		



Gain or lift Charts

Gain or lift is a measure of the effectiveness of a classification model calculated as the ratio between the results obtained with and without the model. Gain and lift charts are visual aids for evaluating performance of classification models. However, in contrast to the confusion matrix that evaluates models on the whole population gain or lift chart evaluates model performance in a portion of the population.





Model Evaluation - Regression

After building a number of different regression models, there is a wealth of criteria by which they can be evaluated and compared.


Root Mean Squared Error

RMSE is a popular formula to measure the error rate of a regression model. However, it can only be compared between models whose errors are measured in the same units.

$$\underline{RMSE} = \sqrt{\frac{\sum_{i=1}^n (\underline{p_i} - \underline{a_i})^2}{n}}$$

a = actual target

p = predicted target



Unlike RMSE, the relative squared error (RSE) can be compared between models whose errors are measured in the different units.

$$RSE = \frac{\sum_{i=1}^n (p_i - a_i)^2}{\sum_{i=1}^n (\bar{a} - a_i)^2}$$

Mean Absolute Error

The mean absolute error (MAE) has the same unit as the original data, and it can only be compared between models whose errors are measured in the same units. It is usually similar in magnitude to RMSE, but slightly smaller.

$$MAE = \frac{\sum_{i=1}^n |p_i - a_i|}{n}$$

Coefficient of Determination


The coefficient of determination (R^2) summarizes the explanatory power of the regression model and is computed from the sums-of-squares terms.

$$\text{Coefficient of Determination} \rightarrow R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

$$\text{Sum of Squares Total} \rightarrow SST = \sum (y - \bar{y})^2$$

$$\text{Sum of Squares Regression} \rightarrow SSR = \sum (y' - \bar{y}')^2$$

$$\text{Sum of Squares Error} \rightarrow SSE = \sum (y - y')^2$$



R² describes the proportion of variance of the dependent variable explained by the regression model. If the regression model is “perfect”, SSE is zero, and R² is 1. If the regression model is a total failure, SSE is equal to SST, no variance is explained by regression, and R² is zero.



Thanks!