//============================================

How to become machine learining expert!

https://www.kaggle.com/discussions/general/262128

//==================================================================

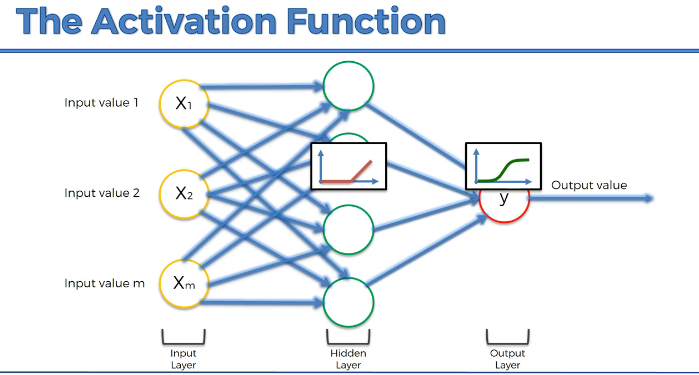
recall simple blogs for linear regression, multiple linear regression and logistic regression

<https://www.superdatascience.com/blogs/the-ultimate-guide-to-regression-classification>

Deep learning Notes:

Here we 4 types functions that are used in neural networks architectures

1. Thershold function
2. Sigmoid function
3. Rectify function
4. Hyperparabolic tangent function



=============================================

## curse of dimensionality example ?

-> The curse of dimensionality is like adding more and more ingredients to a recipe.

-> In low dimensions, you can easily manage and understand the flavor. However, as you add more and more ingredients (dimensions),

-> the overall taste becomes more complex, and it becomes challenging to maintain a balanced and reliable recipe.

-> Similarly, in high-dimensional data, maintaining a clear understanding and making accurate predictions becomes more challenging

gradient descent with example ?

-> In a real-world scenario, you might be trying to find the best settings for a recipe.

-> You close your eyes, take a small taste, and adjust the ingredients based on whether it tastes better or worse.

-> You repeat this process until you have the perfect recipe.

Stochastic gradient descent with example ?

-> In a real-world scenario, think of each cookie as a customer review for a product.

-> Instead of waiting for all reviews to come in, you continuously read and adjust your product based on

-individual reviews until you have a product that satisfies most customers

* The first thing we can observe is that the data augmentation was performed succesfully,
* as the feature distribution for faulty instancies have not been significantly distorted(mishappend).
* It should also be noted that in Rotational Speed, Torque and Tool Wear the observations relating to failures have a density peak in extreme zones of the distribution.
* This implies that the outliers we discussed in Section 2.3 are not to be imputed(assign) to mistakes in the dataset building but rather to the natural variance of the same.
* This becomes even clearer when observing the distributions relative to the single causes of failure: in particular, an almost symmetrical behavior is recognized in Rotational Speed and Torque while in Tool Wear a clear separation is observed between PWF and HDF failures on lower values, and the peaks that are found at higher values relative to TWF and OSF. This is perfectly consistent with the description of the targets reported in the "Task and dataset description" section.

## Principal component analysis ( PCA ):

Simplify the complex scenes, identifying the underline patterns, reducing dimensionality of reduction

How it works PCA:

1. We have to ensure the scaling all features.
2. Covariance of matrix. It shows the relationships features in all variables
3. Find out eigen values and eigen vectors. With help of eigen values to represent the principal component and eigen vectors are represent the direction of principle component.
4. Select the most data falls in which category ( component)
5. Project the original data into selected components to reduce the dimensionality.

The idea of PCA is simple — **reduce the number of variables of a data set, while preserving as much information as possible.**

For better understand PCA read this article

<https://builtin.com/data-science/step-step-explanation-principal-component-analysis>

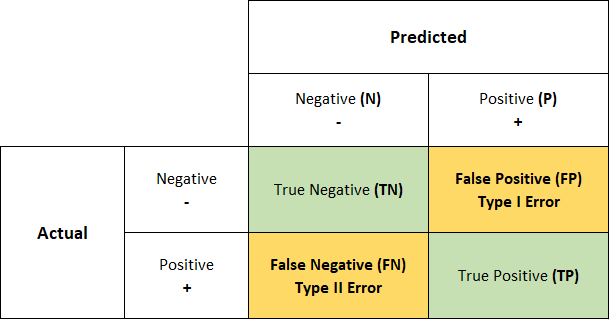
* we observe that the features related to temperature, as well as those related to power, are widely correlated.
* Furthermore, Tool Wear correlates well with both of our targets, confirming what we have observed by studying PCA.
* Finally, a less strong correlation is also observed between the torsion and the two targets.

1. The goal of this section is to find the best model for binary classification of the dataset to predict whether or not there will be Machine Failure.
2. Classification algorithms are part of data mining and use supervised machine learning methods to make predictions about data. In particular, a set of data already divided (”labeled”) into two or more classes of belonging is provided as input thanks to which a classification model is created, which will than be used on new (”unlabeled”) data to assign them to the appropriate class.
3. The starting dataset is usually divided into three groups: the training dataset, i.e. the sample of data used to fit the model, the validation dataset, i.e. the sample of data used to provide an evaluation of a model fit on the training dataset while tuning model hyperparameters and the test dataset, which has the purpose of testing the model. At the beginning of a project a data scientist must make this division and the common ratios used are:

## Metrics types from sklearn.

1. Confusion Matrix
2. Accuracy Score.
3. F1 Score
4. ROC score.

**Confusion Matrix,**



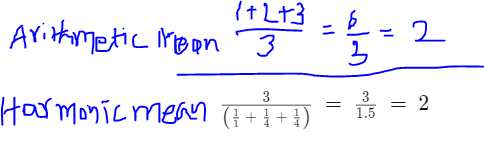
========================================

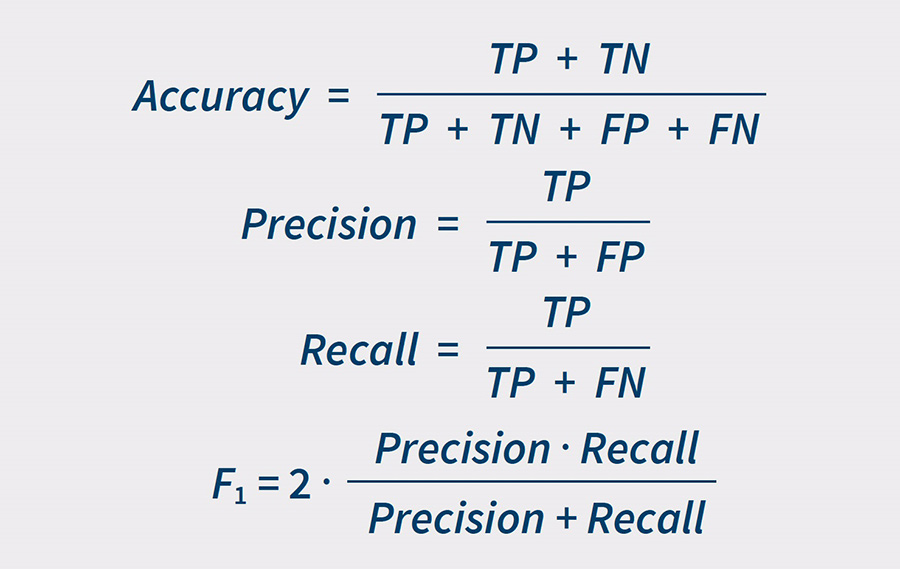
**Accuracy Score and F1 Score**

Accuracy : where the classes are balanced we can use accuracy score.

F1 Score : Especially useful when there is an imbalance between classes or when both precision and recall need to be considered.

The reason the harmonic mean is commonly used in the F1 score is because it naturally balances precision and recall. This is particularly valuable in situations where both false positives and false negatives have significant consequences, and you want to avoid favoring one at the expense of the other.



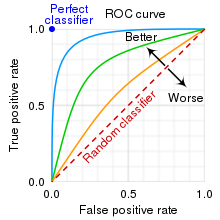


**ROC** **(Receiver Operating Characteristic Curve)**

Particularly relevant in industries where distinguishing between true positives and false positives is critical, such as in medical diagnostics or fraud detection..we can use the ROC when binary classification problems.

Refer this youtube video link for better understand

<https://www.youtube.com/watch?v=4jRBRDbJemM&pp=ygUDcm9j>



=========================================

OVR (One Vs Rest):

* In roc\_auc\_score, multi - class level we have multi-class parameters. It's a default behavior. For each class , it treats that class as the positive and rest as the negative class.
* Then it calculates ROC AUC for each class independently.

Average:

* In fbeta we have an average parameter that is used to specify how the scores should be aggregated across different classes in multi class classification problems.
* why are we using beta parameters in Fbeta\_score ?

The beta parameter determines the weight of recall in the combined score. beta < 1 lends more weight to precision, while beta > 1 favors recall (beta -> 0 considers only precision, beta -> +inf only recall).

Why we using stratify parameter in train\_test\_split()?

X\_train\_val, X\_test, y\_train\_val, y\_test = train\_test\_split(X,y, test\_size=0.1,

stratify=df\_pre['Failure Type'],

random\_state= 0)

* By using stratify we can split the data and ensure that both the training and testing sets have a similar proportion of passing and failing students. This helps your model learn from both types of students, making it better at predicting whether any student, passing or failing, will do well.

Below code is Stratify example.

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import make\_classification

# Generate a synthetic dataset with imbalanced classes

X, y = make\_classification(n\_samples=1000, n\_features=5, n\_classes=2, weights=[0.9, 0.1], random\_state=42)

# Without stratify

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# With stratify

X\_train\_stratify, X\_test\_stratify, y\_train\_stratify, y\_test\_stratify = train\_test\_split(X, y, test\_size=0.2, stratify=y, random\_state=42)

# Print class distribution in the original dataset

print("Original class distribution:")

print("Class 0:", sum(y == 0))

print("Class 1:", sum(y == 1))

print()

# Print class distribution in the datasets without stratify

print("Class distribution without stratify:")

print("Training Set - Class 0:", sum(y\_train == 0))

print("Training Set - Class 1:", sum(y\_train == 1))

print("Testing Set - Class 0:", sum(y\_test == 0))

print("Testing Set - Class 1:", sum(y\_test == 1))

print()

# Print class distribution in the datasets with stratify

print("Class distribution with stratify:")

print("Training Set (Stratified) - Class 0:", sum(y\_train\_stratify == 0))

print("Training Set (Stratified) - Class 1:", sum(y\_train\_stratify == 1))

print("Testing Set (Stratified) - Class 0:", sum(y\_test\_stratify == 0))

print("Testing Set (Stratified) - Class 1:", sum(y\_test\_stratify == 1))

Output for above code

Original class distribution:

Class 0: 895

Class 1: 105

Class distribution without stratify:

Training Set - Class 0: 710

Training Set - Class 1: 90

Testing Set - Class 0: 185

Testing Set - Class 1: 15

Class distribution with stratify:

Training Set (Stratified) - Class 0: 716

Training Set (Stratified) - Class 1: 84

Testing Set (Stratified) - Class 0: 179

Testing Set (Stratified) - Class 1: 21

## Classification models

1. Logistic regression
2. KNN(K-Nearest Neibors)
3. Support vector machines
4. Random Forest
5. XGBoost

XGBoost:

XGBoost is a Gradient boosted decision tree (GBDT)is machine learning library.

GBDT is decision tree ensemble learning algorithm, similar like random forest.

Difference is boosting technique, it trains the ensemble of shallow decision trees

With each iteration error residuals of the previous model to fit into the next model.

The final prediction is sum of all the tree predictions.

Purpose of Permutation feature importance:

For good accuracy depends on selecting features. Which feature is best for accuracy to get this we using permutation feature importance.

How its works:

* Original dataset features compared to shuffled the values in single feature. Observe the model performance
* Main purpose why we using permutation is how model is depends on which feature is important to get good accuracy.

Multi - class task description

* We now proceed to the second task of this project, that is predict not only if there will be a failure, but also the type of failure that will occur.
* So we are in the case of multiclass classification problems that make the assumption that each sample is assigned to one and only one label.
* This hypothesis is verified because in data preprocessing we removed all the ambiguous observations that belonged to more than one class.

==========================

* For multiclass targets, when we calculate the values of AUC, F1 and F2 scores, we need to set the parameter "average". We choose "average=weighted", in order to account for class imbalance: in fact, at the end of data preprocessing, we have 80% WORKING machine and 20% that fail.
* As for binary classification task, we choose Logistic Regression as baseline model
* and we look for models that get higher values for the chosen metrics.
* In particular, we adapt to the multiclass case the models developed in the previous section.
* While many classification algorithms (such as K-nearest neighbor, Random Forest and XGBoost) naturally permit the use of more than two classes,
* some (like Logistic Regression and Support Vector Machines) are by nature binary algorithms;
* these can, however, be turned into multiclass classifiers by a variety of strategies.
* For our project, we decide to use "OnevsRest" approach, who involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives. We choose it because it is computationally more efficient than other types of approach.

## Conclusion about the Predictive Maintenance Project:

According to the analyses carried out and the results obtained, it is possible to make some conclusive considerations related to this project.

We decided to tackle two tasks:

1. predict whether a machine will fail or not and
2. predict the type of failure that will occur.
3. Before developing the models we did data preprocessing to ensure the validity of the assumptions of applicability of the models and ensure the best performances.
4. Briefly, in the preprocessing phase we have deleted some ambiguous samples,
5. We applied a label encoding to the categorical columns and then we performed the scaling of the columns with StandardScaler. We also noticed the presence of some data points which at first we referred to as outliers but later turned out to be part of the natural variance of the data and played an important role in the classification task.
6. Then we ran PCA and found that most of the variance is explained by the first three components, that can be represented as the following features:

* combination of the two Temperatures,
* Machine Power (product of Rotational Speed and Torque) and
* Tool Wear.

1. In according to this, we found that these are the features that contribute the most in the predictions when apply the models.
2. Contrary to logical predictions, we demonstrated that the machine’s type does not affect the presence of failure.

* At the end, we can conclude that for both task the chosen models perform very well
* For both tasks the best model is XGBoost and the worst is KNN;
* However, the response time of KNN is instant while XGBoost takes more time and this further increases when we proceed with the multi-class classification task.
* The choice of the model depends on the needs of the company:
* For faster applications one can use KNN while if one cares more about accuracy one can use XGBoost.

// —---------------------------------------------------------------------

Why are we using feature scaling in support vector regression for both independent and dependent variables separately.

* The primary reason is SVR algorithm is very sensitive and how relies distance between data points to the feature space.
* When we scaling the independent variables, you ensure the each feature contributes equally distance of calculations.
* If one feature much larger than the another, it might dominates the distance computations and disproportionate influencer the model.
* By scaling the features, you make SVR algo more robust and less sensitive to scale individual features
* Scaling is necessary for dependent variables when dependent variable data points are very different to the independent variables.

Robust meaning - strong, healthy

What happens if i use same scaling object in SVR for both independent and dependent variables ?

* If we use same scaling object for both independent and dependent variables, SVR may leads as potential issues.
* Primary reason is scaling the feature is treating the all variables are same during learning process
* Separate scaling objects are necessary when algorithms relies on particular in distance calculations points.
* Lets see what happenes when we using same scaling object for both independent and dependent variables
  + Dominance features : if the features and targets are on different scale. SVR might give more importance to the variable with larger scale. This could be results biased
  + Numerical instabilities : SVR involves solving optimization problems. If we use same object in svr it gives us numerical instability during optimization process.
  + Model Interpretability : scaling the features and targets are important for interpretability. If the variables are different scale. It may be challenging to interpret the coefficients or support vectors accurately.

## RBF - Radial basis function:

Rbf kernel allows SVR to complex model, non linear relationship of the data by mapping into the higher dimensional data.

*This choice of the σ affects the smoothness of the decision boundary.*

*The smaller values are leads to more complex (less smooth) boundary.*

*The higher values are resulting smooth boundary*

Logistic regression :- its predict the categorical dependent variable value from number of independent variable

Logistic regression is type of supervised learning. It predict the values between 0 and 1. Its like true or false, Yes or no, up or down etc…

Logistic regression have sigmoid function to determines the curve about the logistic regression

Logistic regression only use classification problems.

// - - - - - – - - - - —---------------------------------------------------

Simple Linear Regression Assumptions:

* Linearity
* Homoscedasticity
* Independence
* Normality

Multiple Linear Regression Assumptions:

* No multi collinearity
* Future selection
* Over fitting

import pandas as pd

import numpy as np

# import the dataset

dataset = pd.read\_csv('/content/drive/MyDrive/Cloud Lab/Lab Experiments/Colab Notebooks/Part 3 - Classification/1. Logistic Regression/Copy of Social\_Network\_Ads.csv')

X = dataset.iloc[:,:-1].values

y = dataset.iloc[:,-1].values

print(X)

print(y)

# splitting the dataset into train set and test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

print(X\_train)

print(X\_test)

print(y\_train)

print(y\_test)

# now we doing the feature scaling for independent variables

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

print(X\_test)

print(X\_train)

# train the logistic regression model

from sklearn.linear\_model import LogisticRegression

lr = LogisticRegression(random\_state=0)

lr.fit(X\_train, y\_train)

lr.predict([[19,40000]])

y\_pred = lr.predict(X\_test)

print(y\_pred)

# checking the prediction results and actual results

np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1)

ANN:

<https://www.superdatascience.com/blogs/the-ultimate-guide-to-artificial-neural-networks-ann>

ann.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

Details explanation about above code

* optimizer=’adam’
  + By using this adam optimizer algorithm used to minimize the loss function by adjusting the weights of the model
  + Adam optimizer is popular algorithm that combines of RMSprop and Momentum.
* loss function - by using loss function we can measure performance of the model. binary classification problems using binary\_crossentropy. If we have more than two classed we might use categorical\_crossentropy for multi-classification problem.
* Metrics (metrics=['accuracy']): Metrics are used to evaluate the performance of the model during training. 'accuracy' is a common metric for classification problems. It represents the proportion of correctly classified instances.

PPT Notes for ANN:

<https://www.slideshare.net/KirillEremenko/deep-learning-az-artificial-neural-networks-ann-module-1>

<https://www.slideshare.net/KirillEremenko/deep-learning-az-artificial-neural-networks-ann-plan-of-attack>

ANN Task:

**Build an ANN Regression model to predict the electrical energy output of a Combined Cycle Power Plant.**

**https://www.udemy.com/course/linear-regression-with-artificial-neural-network/?referralCode=2E5BC40C4E666DC53A22**

**CNN**

Refer this below article link

[*https://serokell.io/blog/introduction-to-convolutional-neural-networks#why-do-we-use-convolution-in-neural-networks%3F*](https://serokell.io/blog/introduction-to-convolutional-neural-networks#why-do-we-use-convolution-in-neural-networks%3F)

What is problem about normal NN ?

According to CIFAR dataset, we used to images for deep learning problems. For example we have one image like 32x32 px have 3 colurs. It meant by single fully connected neuron in hidden neural networks its look like 32x32x3 = 3072 weights. It is still manageable

What if we have more bigger image 310x310, it would have 270000 weights. Training this much of data is requiring demand of computational power.

Huge neural networks requires lot of resources, but even then remains the prone to overfitting because the larger number of parameters are enable it just memorize the dataset.

CNN uses the parameter sharing. All the neurons are particular in feature map share weights which makes whole system less computational intense.

CNN means **convolutional neural networks**.

Why do we using convolutional in Neural Networks ?

Its just basically merging the two set of information

Cnn can give to us like pattern recognition, identifying the image categories.

Cnn have three types of layers:

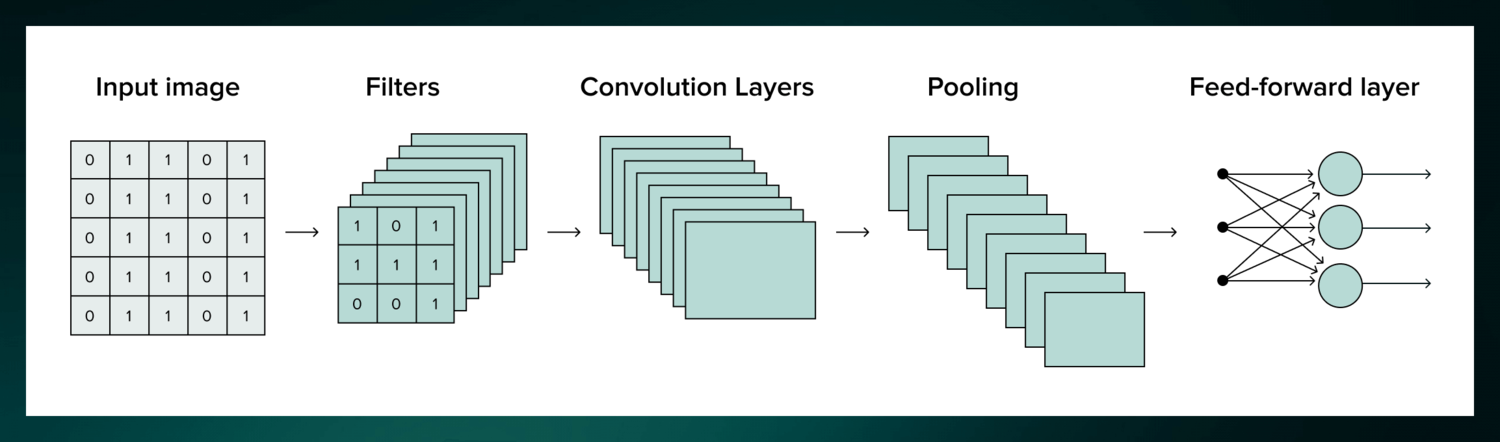
* Convolutional layers
* Pooling layers
* Fully connected layers

Filters also called as feature detectors or kernal

Convolutional layers: it is responsible for recognition the features from pixels

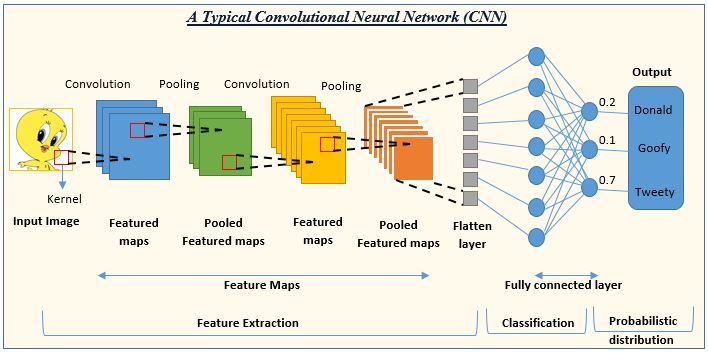
Pooling layers: it is responsible for making these features more abstractable

Fully connected layers: it is responsible for acquired features from predictions



Applications of CNN:

* Object detection
* Image classification
* Audio visual matching
* Object reconstruction
* Speech recognition



Cnn points:

* For CNN strengths come from convolutional layer
* 3 or 4 convolutional layers it can recognize the handwritten digits
* 25 convolutional layers it can recognize the human faces

CNN design:

* Cnn is a multi layered feed forward neural networks, made by many unseen neural networks in particular order
* Its a sequential design that gives permission to CNN to learn hierarchical attributes

What is meant by Spatial dimensions -> different directions in event or object measuring distance or locating length, height and this process called spatial dimensions.

Types of CNN :

1. LeNet - ( it can recognize the handwritten digit. It was developed by late 1990 )
2. Alexnet - (it can recognize or classify the images. It was developed by 2012 )
3. ResNet
4. GoogleNet
5. MobleNet
6. VGG

Applications of CNN:

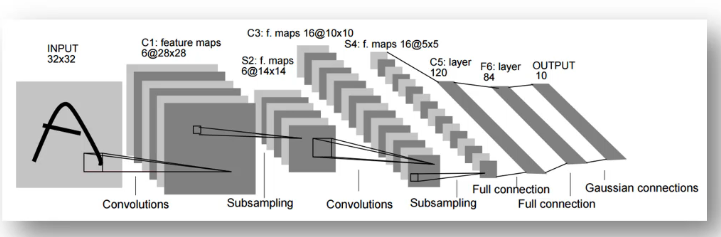
* Decoding facial recognition
* Understanding climate
* Collecting historic and environmental elements

Case study about CNN for diabetic ratinopathy:

* Diabetic ratinopathy also know has diabetic eye disease.which destruction occurs to the retina due to diabetes mellitus, It is a major cause of blindness in advance countries.
* Diabetic retinopathy influence up to 80 percent of those who have had diabetes for 20 years or more.
* The overlong a person has diabetes, the higher his or her chances of growing diabetic retinopathy.
* It is also the main cause of blindness in people of age group 20-64.
* Diabetic retinopathy is the outcome of destruction to the small blood vessels and neurons of the retina.

Udemy CNN Notes:

CNN steps 😀

1. Convolutional 2. Max Pooling 3. Flattening. 4 Full connection.
2. 

Relu Layer:

By using relu layer is used to we will get the non linearity. Because of the most of images will works as non linearity.

<https://arxiv.org/pdf/1609.04112.pdf>

Spatial dimensions: it means height, width, depth of the images is called spatial dimensions.

***Pooling :***

Pooling is a down sampling operation most commonly used in cnn to reduce the spatial dimensions of the input volume

Pooling is mainly performed after convolutional layers and achieve several objectives.

Udemy CNN notes:

1. Images transformation only applied in training set not test set. Reason why we using transformation only for trainset is to avoid overfitting. If we not use transformation in trainset we will get huge difference between accuracy and prediction results.

//===================================

Dimensionality Reduction:

Projecting higher dimensional data into lower dimensionality data.

There are two types of techniques we have.

1. Feature Selection
2. Feature Extraction

Feature extraction we have three types of analysis

1. PCA ( Principle Component Analysis )
2. LDA ( Linear Discriment Analysis )
3. Kernal PCA

PCA

PCA achieves this by identifying the directions in which the data varies the most, and these directions become the principal components.

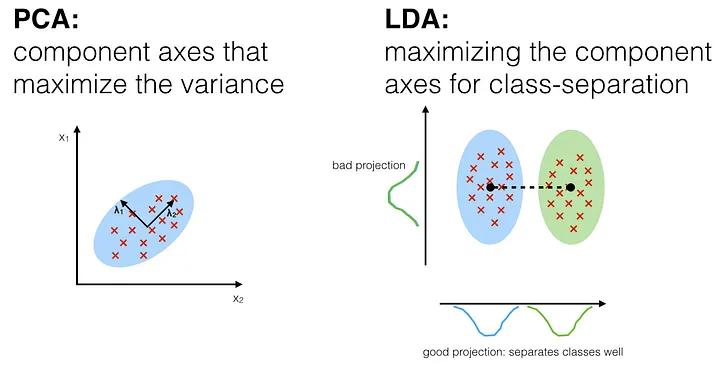
Here's a step-by-step explanation of what PCA does:

1. Centering the Data:
2. Covariance Matrix Calculation:
3. Eigendecomposition:
4. Sorting and Selecting Components:
5. Projection:
6. Variance Retention:
7. Dimensionality Reduction (Optional):

Summary about PCA:

In summary, PCA provides a way to simplify and represent high-dimensional data by identifying the most important directions in the data space. It allows for a reduction in dimensionality while preserving the most significant patterns and variations in the data. PCA is widely used for visualization, noise reduction, and feature extraction in various fields, including machine learning and data analysis.

// =====================================================================



PCA can be used in Unsupervised learning

LDA can be used in Supervised learning

// =====================================================================

**LDA - Linear Discriminant Analysis**

**Unlike Principal Component Analysis (PCA), which focuses on maximizing the variance in the data, LDA is designed to find the linear combinations of features that best separate two or more classes in a supervised manner.**

Summary about LDA:-

LDA is often used as a preprocessing step for classification tasks. It is particularly effective when dealing with datasets where classes are well-separated, and it assumes that the features have a normal distribution within each class. LDA is also closely related to Fisher's Linear Discriminant, and the terms are often used interchangeably.

Here's an overview of how Linear Discriminant Analysis works:

* **Objective:**
  + LDA is primarily used for dimensionality reduction while preserving as much class discriminatory information as possible.
* **Assumptions:**
  + LDA assumes that the data follows a multivariate normal distribution and that the classes have the same covariance matrix.
* **Between-Class and Within-Class Scatter:**
  + LDA aims to maximize the distance between the means of different classes while minimizing the scatter (variance) within each class.
* **Eigenvalue Problem:**
  + LDA solves the generalized eigenvalue problem.
* **Selecting Discriminant Vectors**:
* **Projection:**
  + The original data is projected onto the selected discriminant vectors to obtain the new reduced-dimensional representation.

Kernel PCA:

This is an extension about PCA. the kernel PCA method enable us non linear dimensionality reduction.

While standard PCA is effective for linearly separable data, Kernel PCA allows for capturing complex, non-linear relationships in the data.

Summary about kernel PCA:

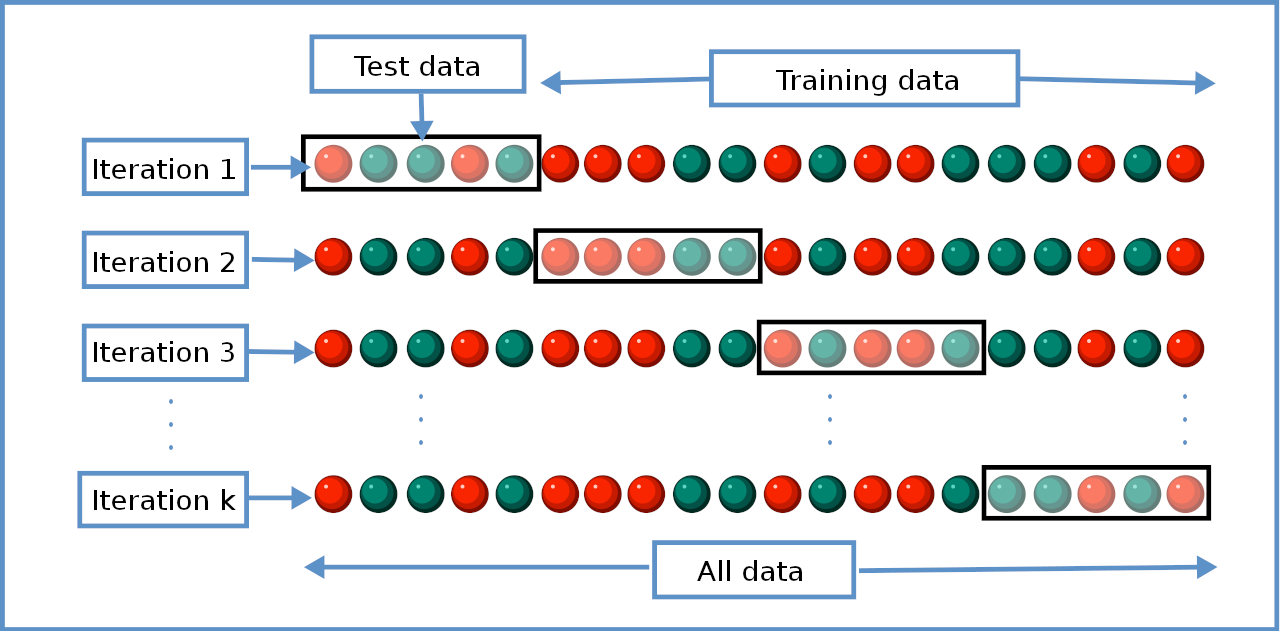
**Kernel PCA allows for capturing non-linear structures in the data, making it suitable for tasks where the underlying relationships are more complex than what linear methods can capture. The choice of the kernel function and its parameters (if applicable) plays a crucial role in the performance of Kernel PCA. Popular libraries, such as scikit-learn in Python, provide implementations of Kernel PCA for practical use**

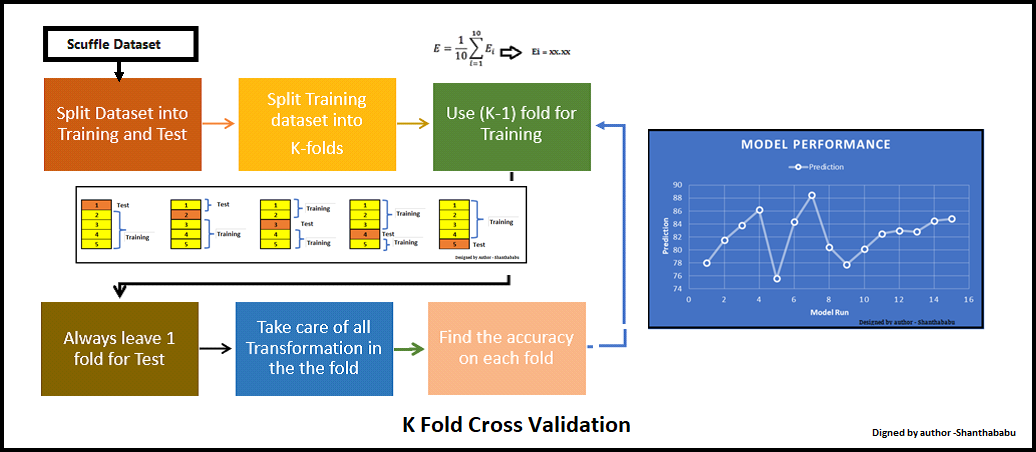
Calculatio

// ==================================================

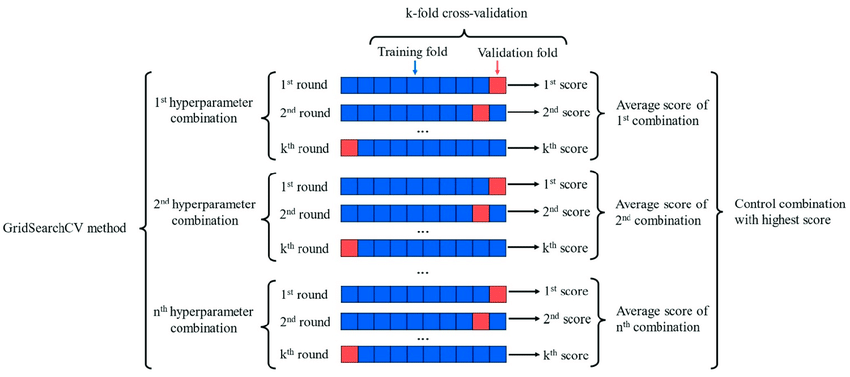
# Model selection and boosting

K Fold Cross validation:





Grid SearchCV



What is Grid search CV

* Grid Search CV is technique used in hyper parameter tuning in machine learning.
* Initially we given set of parameters to the algorithm.
* GSCV is used to get best hyperparameters in the given list.

ChatGPT notes for GridSearchCSV

GridSearchCV (Grid Search Cross-Validation) is a technique used for hyperparameter tuning in machine learning. It systematically searches through a predefined set of hyperparameter combinations for a given algorithm. This helps find the combination that gives the best performance for your specific problem. Let's break down the concept with an example:

### Example: Support Vector Machine (SVM) with GridSearchCV

Let's say you want to train a Support Vector Machine classifier, and you're using the popular library scikit-learn in Python.

```python

from sklearn import svm, datasets

from sklearn.model\_selection import GridSearchCV, train\_test\_split

# Load dataset (for example, the iris dataset)

iris = datasets.load\_iris()

X = iris.data

y = iris.target

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Define the SVM model

svm\_model = svm.SVC()

# Define the hyperparameter grid to search

param\_grid = {'C': [1, 10, 100],

'kernel': ['linear', 'rbf'],

'gamma': [0.1, 0.01, 0.001]}

# Create a GridSearchCV object

grid\_search = GridSearchCV(svm\_model, param\_grid, cv=3, scoring='accuracy')

# Fit the model to the data

grid\_search.fit(X\_train, y\_train)

# Get the best hyperparameters

best\_params = grid\_search.best\_params\_

# Get the best model

best\_model = grid\_search.best\_estimator\_

# Evaluate the best model on the test set

accuracy = best\_model.score(X\_test, y\_test)

print("Best Hyperparameters:", best\_params)

print("Best Model Accuracy:", accuracy)

```

In this example:

1. \*\*Load Data:\*\* Load the Iris dataset as an example dataset.

2. \*\*Split Data:\*\* Split the dataset into training and testing sets.

3. \*\*Define Model:\*\* Create an SVM model (`svm\_model` in this case).

4. \*\*Define Hyperparameter Grid:\*\* Specify the hyperparameters and their potential values that you want to search. In this example, we're looking at different combinations of `C` (penalty parameter), `kernel` (type of kernel), and `gamma` (kernel coefficient).

5. \*\*Create GridSearchCV Object:\*\* Instantiate a `GridSearchCV` object, providing the model, the hyperparameter grid, the number of cross-validation folds (`cv`), and the scoring metric to optimize (e.g., accuracy).

6. \*\*Fit Model:\*\* Fit the GridSearchCV object to the training data. It will systematically try all combinations of hyperparameters using cross-validation and select the combination that gives the best performance.

7. \*\*Retrieve Best Hyperparameters and Model:\*\* After fitting, you can retrieve the best hyperparameters and the best model.

8. \*\*Evaluate Best Model:\*\* Finally, you can evaluate the performance of the best model on the test set.

GridSearchCV helps automate the process of hyperparameter tuning, saving you from manually trying out different combinations and allowing you to find the optimal set efficiently.

// ================================================