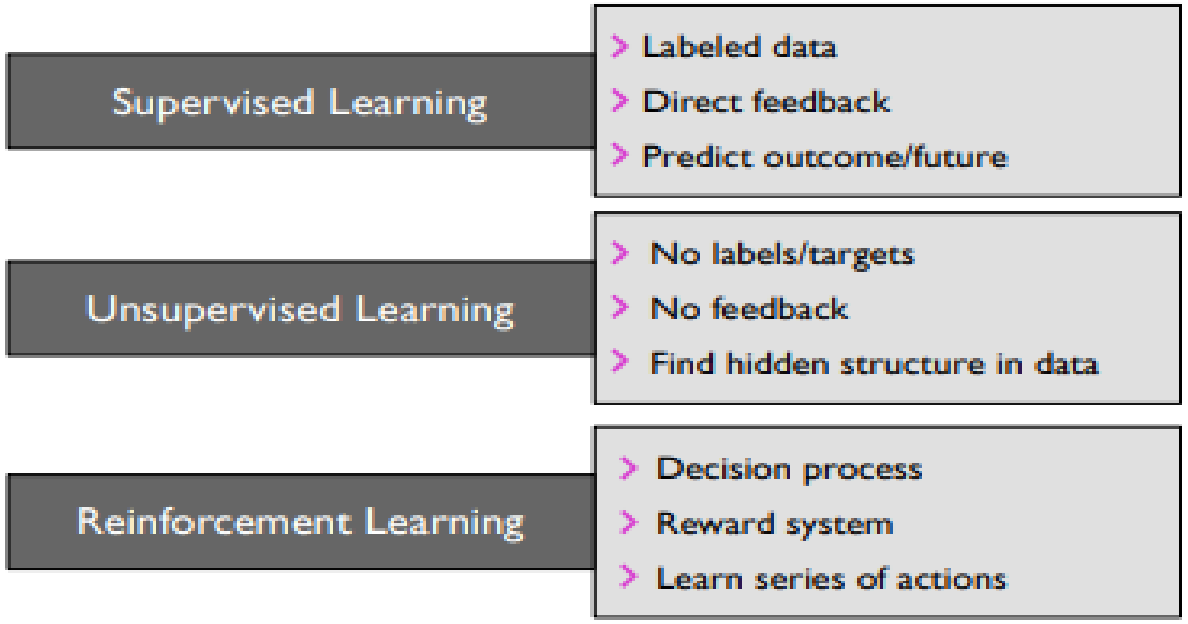
# Categories of machine learning-TRIZA

The three broad categories of machine learning are summarized in Figure 1.0: (1) supervised learning,

(2) unsupervised learning, and

(3) reinforcement learning

Fig 1.0



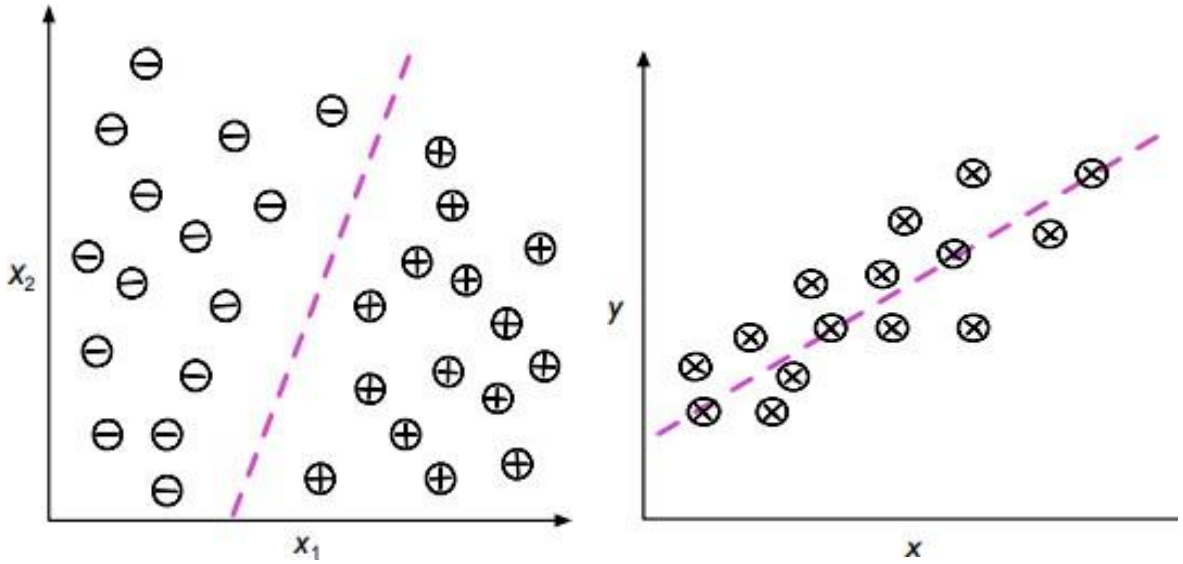
# 1. Supervised learning

Supervised learning is the subcategory of machine learning that focuses on learning a classification (Figure 2.0), or regression model (Figure 3.0), that is, learning from labeled training data (i.e., inputs that also contain the desired outputs or targets; basically, "examples" of what we want to predict).

Fig 2.0 and 3.0 respectively.

Fig 2.0. Illustration of a binary classification problem (plus and minus signs denote class labels) and two feature variables, (x1 and x2)

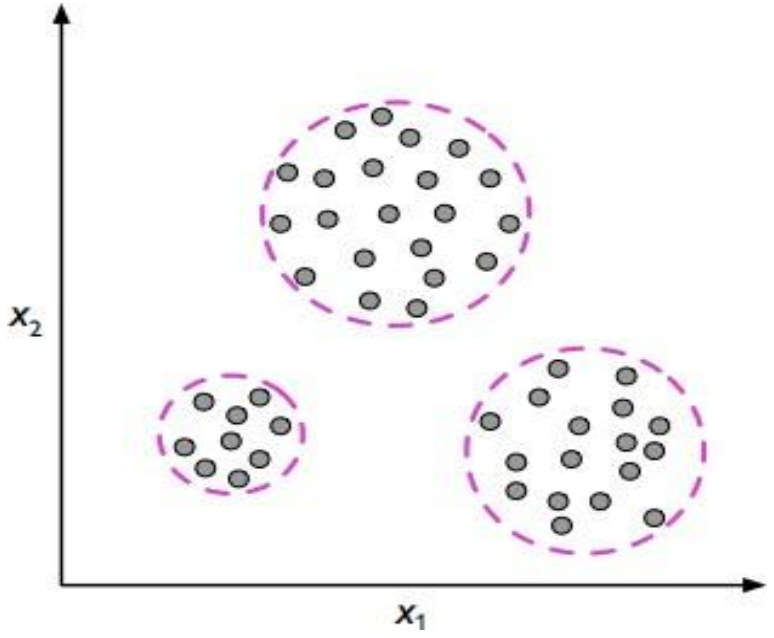
Fig 3.0 Illustration of a linear regression Model with one feature variable (x1) and The target variable y. The dashed-line indicates the functional form of the linear regression model.



# 2. Unsupervised learning

In contrast to supervised learning, unsupervised learning is a branch of machine learning that is concerned with unlabeled data. Common tasks in unsupervised learning are clustering analysis (assigning group memberships; Figure 4) and dimensionality reduction (compressing data onto a lower-dimensional subspace or manifold).

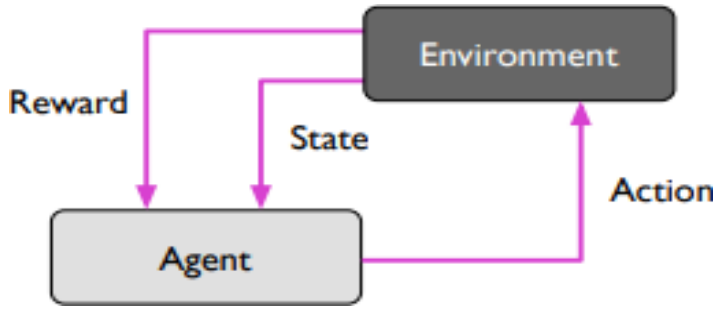
Fig 4.0: Illustration of clustering, where the dashed lines indicate potential group membership assignments of unlabeled data points.



# 3. Reinforcement learning

- Reinforcement is the process of learning from rewards while performing a series of actions. In reinforcement learning, we do not tell the learner or agent (for example, a robot), which action to take but merely assign a reward to each action and/or the overall outcome. Instead of having "correct/false" labels for each step, the learner must discover or learn a behavior that maximizes the reward for a series of actions. In that sense, it is not a supervised setting.
- RL is somewhat related to unsupervised learning; however, reinforcement learning really is its own category of machine learning. Reinforcement learning will not be covered further in this class. Typical applications of reinforcement learning involve playing games (chess, Go, Atari video games) and some form of robots, e.g., drones, warehouse robots, and more recently self-driving cars.

Fig.5.0: Illustration of reinforcement learning




4. **Semi-supervised learning**: Loosely speaking, semi-supervised learning can be described as a mix between supervised and unsupervised learning. In semi-supervised learning tasks, some training examples contain outputs, but some do not. We then use the labeled training subset to label the unlabeled portion of the training set, which we then also utilize for model training.

# Introduction to Supervised machine learning

In supervised learning, we are given a labeled training dataset from which a machine learning algorithm can learn a model. The learned (or trained) model can be used to predict labels of unlabeled data points. These unlabeled data points could be either test data points or unlabeled data that is already collected or will be collected in the future. For example, given a corpus of spam and non-spam email, a supervised learning task would be to learn a model that predicts to which class (spam or non-spam) new emails belong. Of course, this all underlies the assumption that the training dataset and unlabeled data points for prediction have been sampled from the same probability distribution – after all, we cannot expect the model to make reliably predictions on fundamentally different data. Figures 6 and 7 provide a simplified and more detailed overview of a typical machine learning workflow.

- Fig 6.0:  rough overview of the supervised learning process
- Fig 7.0 :Detailed illustration of supervised learnin process

**Fig 6.0**

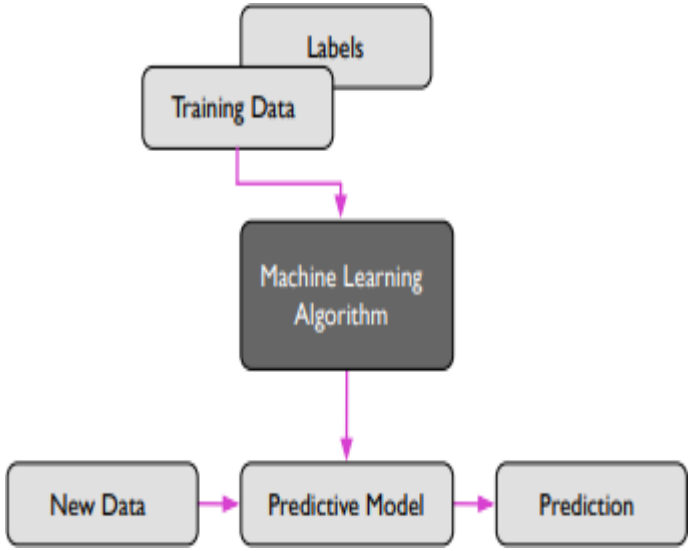
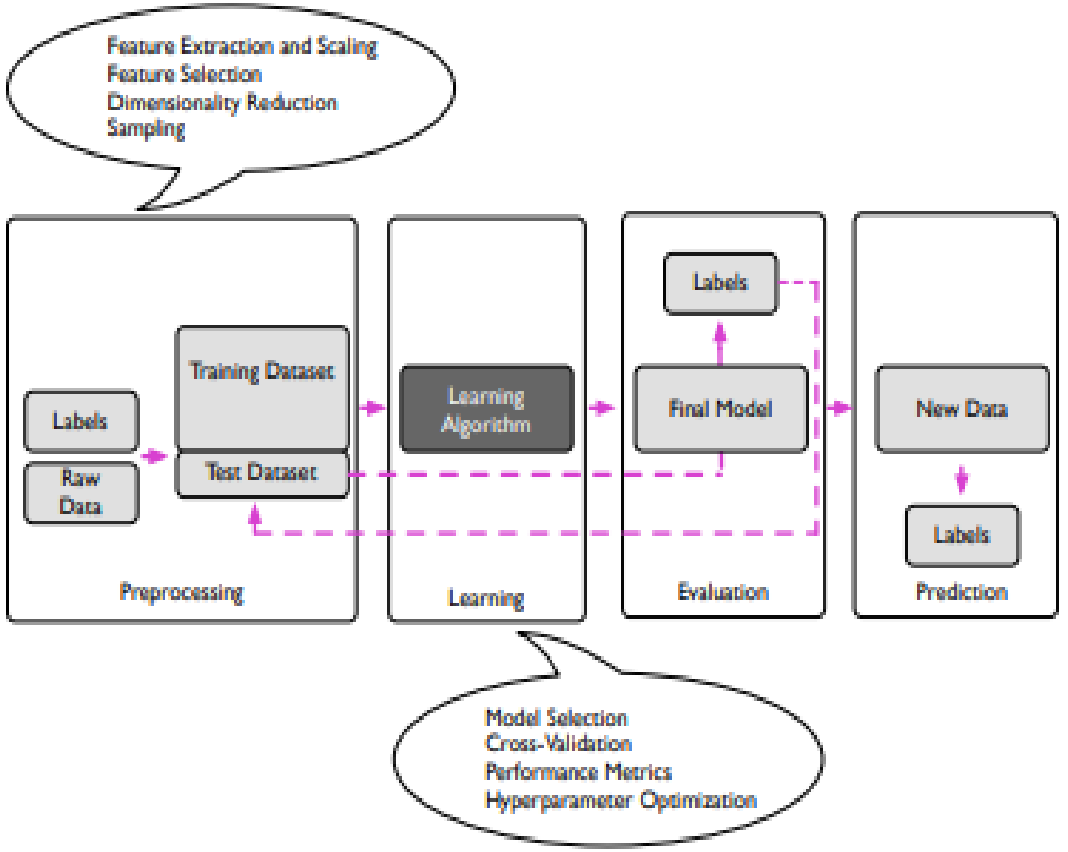



**Fig 7.0**



# Classes of supervised machine learning algorithms

Listed below are some common classes of machine learning algorithms:

- Generalized linear models (e.g., logistic regression)
- Support vector machines (e.g., linear SVM, RBF-kernel SVM)
- Artificial neural networks (e.g., multi-layer perceptrons)
- Tree- or rule-based models (e.g., decision trees)
- Graphical models (e.g., Bayesian networks)
- Ensembles (e.g., Random Forest)
- Instance-based learners (e.g., K-nearest neighbors)

# Components of Machine Learning Algorithms

1. **Representation.** The first component is the "representation," i.e., which hypotheses we can represent given a certain algorithm class. Optimization.
2. The second component is the **optimization** metric that we use to fit the model.
3. **Evaluation.** The evaluation component is the step where we evaluate the performance of the model after model fitting.

To extend this list slightly, these are the following 5 steps that we want to think about when approaching a machine learning application:

1. Define the problem to be solved.
2. 2. Collect (labeled) data.
3. 3. Choose an algorithm class.
4. 4. Choose an optimization metric for learning the model.
5. 5. Choose a metric for evaluating the model.

# Model Evaluation metrics

- While data preparation and training a machine learning model is a key step in the machine learning pipeline, it's equally important to measure the performance of this trained model. How well the model generalizes on the unseen data is what defines adaptive vs non-adaptive machine learning models.
- By using different metrics for performance evaluation, we should be in a position to improve the overall predictive power of our model before we roll it out for production on unseen data.
- Without doing a proper evaluation of the ML model using different metrics, and depending only on accuracy, it can lead to a problem when the respective model is deployed on unseen data and can result in poor predictions.
- This happens because, in cases like these, our models don't learn but instead memorize; hence, they cannot generalize well on unseen data.

# Model evaluation metrics contd'

Let us now define the evaluation metrics for evaluating the performance of a machine learning model, which is an integral component of any data science project. It aims to estimate the generalization accuracy of a model on the future (unseen/out-of-sample) data.

These metrics are grouped into different categories based on the ML model/application they are mostly used for, and cover the popular metrics used in the following problems:

- *Classification Metrics (accuracy, precision, recall, F1-score, ROC, AUC, …)*
- *Regression Metrics (MSE, MAE)*
- *Ranking Metrics (MRR, DCG, NDCG)*
- *Statistical Metrics (Correlation)*
- *Computer Vision Metrics (PSNR, SSIM, IoU)*
- *NLP Metrics (Perplexity, BLEU score)*
- *Deep Learning Related Metrics (Inception score, Frechet Inception distance)*

There is no need to mention that there are various other metrics used in some applications (FDR, FOR, hit@k,etc. **It** is also worth mentioning that **metric is different from loss function**. Loss functions are functions that show a measure of the model performance and are used to train a machine learning model (using some kind of optimization), and are usually differentiable in model's parameters. On the other hand, metrics are used to monitor and measure the performance of a model (during training, and test), and do not need to be differentiable. However if for some tasks the performance metric is differentiable, it can be used both as a loss function and a metric, such as MSE.

## Confusion matrix

This is not a metric but important to know.

- A confusion matrix is a matrix representation of the prediction results of any binary testing that is often used to describe the performance of the classification model (or "classifier") on a set of test data for which the true values are known.
- The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

# Confusion matrix contd'

- One of the key concept in classification performance is **confusion matrix** (AKA error matrix), which is a tabular visualization of the model predictions versus the ground-truth labels. Each row of confusion matrix represents the instances in a predicted class and each column represents the instances in an actual class.
- Let's go through this with an example. Let's assume we are building a binary classification to classify cat images from non-cat images. And let's assume our test set has 1100 images (1000 non-cat images, and 100 cat images), with the below confusion matrix.

- **Confusion matrix representation:**

| | | Actual Class | |
|---|---|---|---|
| | | Cat | Non-Cat |
| Predicted Class | Cat | 90 | 60 |
| | Non-Cat | 10 | 940 |

A simple confusion matrix

- **Out of 100 cat images** the model has predicted 90 of them correctly and has mis-classified 10 of them. If we refer to the "cat" class as positive and the non-cat class as negative class, then 90 samples predicted as cat are considered as as **true-positive**, and the 10 samples predicted as non-cat are **false negative**.
- **Out of 1000 non-cat images**, the model has classified 940 of them correctly, and mis-classified 60 of them. The 940 correctly classified samples are referred as **true-negative**, and those 60 are referred as **false-positive**.
- As we can see diagonal elements of this matrix denote the correct prediction for different classes, while the off-diagonal elements denote the samples which are mis-classified.

# Confusion matrix contd'

Each prediction can be one of the four outcomes, based on how it matches up to the actual value:

**True Positive (TP):** Predicted True and True in reality.

**True Negative (TN):** Predicted False and False in reality.

**False Positive (FP):** Predicted True and False in reality.

**False Negative (FN):** Predicted False and True in reality.

- Now let us understand this concept using hypothesis testing.
- A **Hypothesis** is speculation or theory based on insufficient evidence that lends itself to further testing and experimentation. With further testing, a hypothesis can usually be proven true or false.
- A **Null Hypothesis** is a hypothesis that says there is no statistical significance between the two variables in the hypothesis. It is the hypothesis that the researcher is trying to disprove.

- We would always reject the null hypothesis when it is false, and we would accept the null hypothesis when it is indeed true.
- Even though hypothesis tests are meant to be reliable, **there are two types of errors that can occur.**
- These errors are known as **Type 1 and Type II errors.**
- For example, when examining the effectiveness of a drug, the null hypothesis would be that the drug does not affect a disease.
- **Type I Error:-** equivalent to False Positives(FP).
- The first kind of error that is possible involves the rejection of a null hypothesis that is true.
- Let's go back to the example of a drug being used to treat a disease. If we reject the null hypothesis in this situation, then we claim that the drug does have some effect on a disease. But if the null hypothesis is true, then, in reality, the drug does not combat the disease at all. The drug is falsely claimed to have a positive effect on a disease.

- **Type II Error:-** equivalent to False Negatives(FN).
- The other kind of error that occurs when we accept a false null hypothesis. This sort of error is called a type II error and is also referred to as an error of the second kind.
- If we think back again to the scenario in which we are testing a drug, what would a type II error look like? A type II error would occur if we accepted that the drug has no effect on disease, but in reality, it did.

There are two types of metrics under supervised machine learning:

i. Classification related metrics

ii. Regression related metrics

# CLASSIFICATION RELATED METRICS

- Classification is one of the most widely used problems in machine learning with various industrial applications, from face recognition, Youtube video categorization, content moderation, medical diagnosis, to text classification, hate speech detection on Twitter.
- Models such as support vector machine (SVM), logistic regression, decision trees, random forest, XGboost, convolutional neural network[1], recurrent neural network are some of the most popular classification models.

## 1.Classification accuracy

Overall, how often is the classifier correct?

- Accuracy = (TP+TN)/total
- When our classes are roughly equal in size, we can use accuracy**,** which will give us correctly classified values.

- Accuracy is a common evaluation metric for classification problems. It'sthe number of correct predictions made as a ratio of all predictions made.
- **Misclassification Rate(Error Rate):** Overall, how often is it wrong. Since accuracy is the percent we correctly classified (success rate), it follows that our error rate (the percentage we got wrong) can be calculated as follows:
- Misclassification Rate = (FP+FN)/total
- Classification accuracy is perhaps the simplest metrics one can imagine, and is defined as the **number of correct predictions divided by the total number of predictions,** multiplied by 100**.** So in the above example, out of 1100 samples 1030 are predicted correctly, resulting in a classification accuracy of:
- **Classification accuracy**= (90+940)/(1000+100)= 1030/1100= 93.6%

# 2. Precision

There are many cases in which classification accuracy is not a good indicator of your model performance. One of these scenarios is when your class distribution is imbalanced (one class is more frequent than others). In this case, even if you predict all samples as the most frequent class you would get a high accuracy rate, which does not make sense at all (because your model is not learning anything, and is just predicting everything as the top class). For example in our cat vs non-cat classification above, if the model predicts all samples as non-cat, it would result in a 1000/1100= 90.9%.

# Precision contd’

- Therefore we need to look at class specific performance metrics too. Precision is one of such metrics, which is defined as:
- **Precision= True_Positive/ (True_Positive+ False_Positive)**
- The precision of Cat and Non-Cat class in above example can be calculated as:
- **Precision_cat**= #samples correctly predicted cat/#samples predicted as cat = **90/(90+60) = 60%**
- **Precision_NonCat= 940/950= 98.9%**
- As we can see the model has much higher precision in predicting non-cat samples, versus cats. This is not surprising, as model has seen more examples of non-cat images during training, making it better in classifying that class.

# 3. Recall

- Recall is another important metric, which is defined as the fraction of samples from a class which are correctly predicted by the model. More formally:
- **Recall= True_Positive/ (True_Positive+ False_Negative)**
- Therefore, for our example above, the recall rate of cat and non-cat classes can be found as:
- **Recall_cat= 90/100= 90%**
- **Recall_NonCat= 940/1000= 94%**

# 4. F1-score

- Depending on application, you may want to give higher priority to recall or precision. But there are many applications in which both recall and precision are important. Therefore, it is natural to think of a way to combine these two into a single metric. **One popular metric which combines precision and recall is called F1-score**, which is the harmonic mean of precision and recall defined as:
- **F1-score= 2*Precision*Recall/(Precision+Recall)**
- So for our classification example with the confusion matrix in Figure 1, the F1-score can be calculated as:
- **F1_cat= 2*0.6*0.9/(0.6+0.9)= 72%**
- The generalized version of F-score is defined as below. As we can see F1-score is special case of F_$\mathcal{B}$ when $\mathcal{B}$= 1.

# F1-score contd'

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

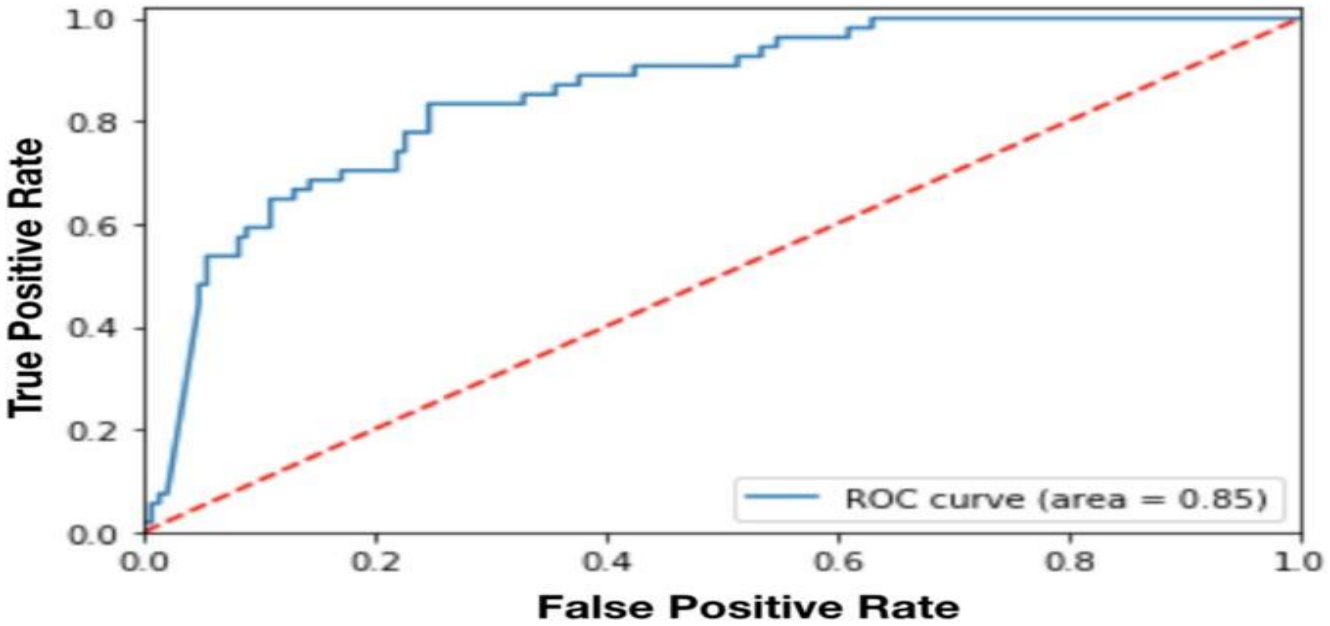
It is good to mention that there is always a trade-off between precision and recall of a model, if you want to make the precision too high, you would end up seeing a drop in the recall rate, and vice versa.

# 5. Sensitivity and specifity

- Sensitivity and specificity are two other popular metrics mostly used in medical and biology related fields, and are defined as:
- Sensitivity= Recall= TP/(TP+FN)
- Specificity= True Negative Rate= TN/(TN+FP)

# 6. ROC Curve

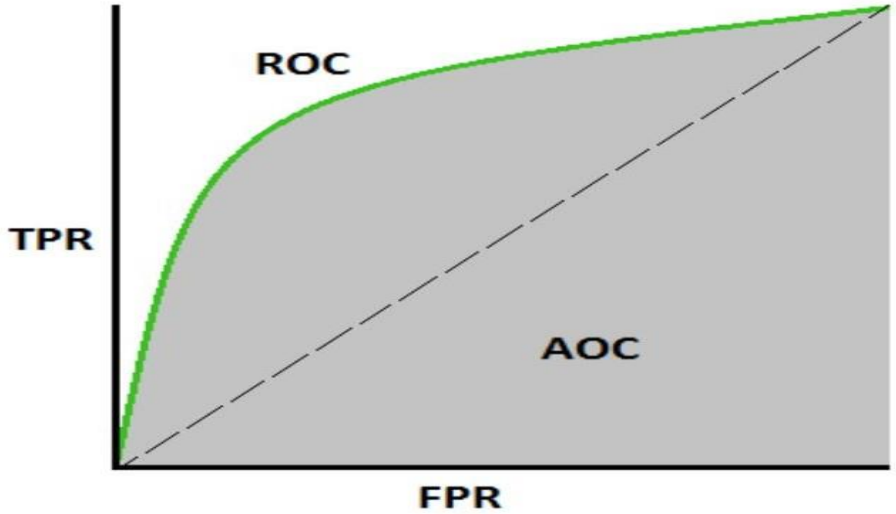
- The **receiver operating characteristic curve** is plot which shows the performance of a binary classifier as function of its cut-off threshold. It essentially shows the true positive rate (TPR) against the false positive rate (FPR) for various threshold values. Let's explain more.
- Many of the classification models are probabilistic, i.e. they predict the probability of a sample being a cat. They then compare that output probability with some cut-off threshold and if it is larger than the threshold they predict its label as cat, otherwise as non-cat. As an example your model may predict the below probabilities for 4 sample images: **[0.45, 0.6, 0.7, 0.3]**. Then depending on the threshold values below, you will get different labels:
- cut-off= 0.5: predicted-labels= [0,1,1,0] (default threshold)
  cut-off= 0.2: predicted-labels= [1,1,1,1]
  cut-off= 0.8: predicted-labels= [0,0,0,0]
- As you can see by varying the threshold values, we will get completely different labels. And as you can imagine each of these scenarios would result in a different precision and recall (as well as TPR, FPR) rates.
- ROC curve essentially finds out the TPR and FPR for various threshold values and plots TPR against the FPR. A sample ROC curve is shown in Figure below:
-




As we can see from this example, the lower the cut-off threshold on positive class, the more samples predicted as positive class, i.e. higher true positive rate (recall) and also higher false positive rate (corresponding to the right side of this curve). Therefore, there is a trade-off between how high the recall could be versus how much we want to bound the error (FPR). ROC curve is a popular curve to look at overall model performance and pick a good cut-off threshold for the model.

# 7.Area Under Curve(AUC)

- The grey area denotes the AUC




- On high-level, the higher the AUC of a model the better it is. But sometimes threshold independent measure is not what you want, e.g. you may care about your model recall and require that to be higher than 99% (while it has a reasonable precision or FPR). In that case, you may want to tune your model threshold such that it meets your minimum requirement on those metrics (and you may not care even if you model AUC is not too high).

# REGRESSION RELATED METRICS

- Regression models are another family of machine learning and statistical models, which are used to predict a continuous target values[3]. They have a wide range of applications, from house price prediction, E-commerce pricing systems, weather forecasting, stock market prediction, to image super resolution, feature learning via auto-encoders, and image compression.
- Models such as linear regression, random forest, XGboost, convolutional neural network, recurrent neural network are some of the most popular regression models.
- Metrics used to evaluate these models should be able to work on a set of continuous values (with infinite cardinality), and are therefore slightly different from classification metrics.

# 1. MSE (Mean Squared Error)

- "Mean squared error" is perhaps the most popular metric used for regression problems. It essentially finds the average squared error between the predicted and actual values.
- Let's assume we have a regression model which predicts the price of houses in Seattle area (show them with $\hat{y}_i$), and let's say for each house we also have the actual price the house was sold for (denoted with $y_i$). Then the MSE can be calculated as:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- Sometimes people use RMSE to have a metric with scale as the target values, which is essentially the square root of MSE.
- Looking at house pricing prediction, RMSE essentially shows what is the average deviation in your model predicted house prices from the target values (the prices the houses are sold for).

# 2.MAE(Mean Absolute Error)

- Mean absolute error (or mean absolute deviation) is another metric which finds the average absolute distance between the predicted and target values. MAE is define as below:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

- MAE is known to be more robust to the outliers than MSE. The main reason being that in MSE by squaring the errors, the outliers (which usually have higher errors than other samples) get more attention and dominance in the final error and impacting the model parameters.
- It is also worth mentioning that there is a nice maximum likelihood (MLE) interpretation behind MSE and MAE metrics. If we assume a linear dependence between features and targets, then MSE and MAE correspond to the MLE on the model parameters by assuming Gaussian and Laplace priors on the model errors respectively.

# 3. Inlier Ration Metric

- There is also another metric for evaluating regression models, called inlier ratio, which is essentially the percentage of data points which are predicted with an error less than a margin. This metric is mainly used in RANSAC model and its extensions (a family of robust estimation models).