Intro to ML Revision Notes

- ## Introduction to ML:

How is Machine Learning different from Classical Programming?

Classical Programming has rigid rules written by programmers and a lot of hardcoding(i.e, rules are predefined and does not change with data) is involved while ML needs data as input and it predicts based on the learnt rules it learns after training on the data.

Classical Programming: Rigid rules written by programmers, a lot of hardcoding.
Machine Learning (ML): Rules are learnt from training a system/model on the data.

When to use ML over Classical Programming ?

ML > Classical Prog. → applications which may not have easily visible patterns.

How to categorize the different types of ML models?

Based on type of Learning ML algo is categorized as:

- Supervised Learning: When output label (Y) is present in the data
- Unsupervised Learning: Output label (Y) is not in the data
- Reinforcement Learning: Used for AI in games, where there is State, Environment and Reward

Criteria: Type of Learning

1. Supervised: When output label (Y) is present in the training data
2. Unsupervised: Output label (Y) is not present in the training data
3. Reinforcement: Agent takes an action in an “environment” for maximizing “reward” and changes its “state”

Criteria: Type of Task

1. Classification: Input data needs to be classified into categories
2. Regression: Input data needs to be mapped to a real valued output
3. Clustering: Grouping of similar items together as one
4. Recommendation: To recommend items → more likable to the datapoint
5. Forecasting: Data pattern understanding → predicts future values

- **Linear Regression:**

How does the data look for Linear Regression ?

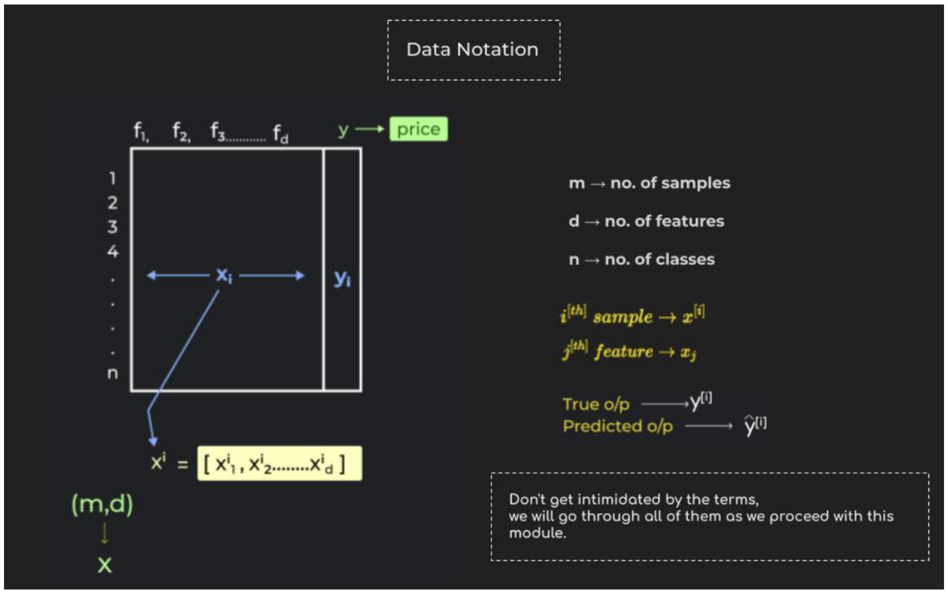
Regression Problem → Data: n samples with $[f_1, f_2, ..., f_d] \in R^{(n \times d)}$ where

- sample X_i consists the features
- That has a target label $y_i \in R$.

How does the training data look for the Regression problem?

Regression → Supervised task, target y_i is numerical

1. Input sample = $X_i \in R^{nXd}$, x_i = $[x_{i1}, ..., x_{id}]^T$
2. Output sample = $y_i \in R$
3. Training example = (X_i, y_i)
4. Training Dataset = $\{(X_i, y_i),\ i = 1, 2,..\ n\}$,


Data Notation
f_1, f_2, f_3.......... f_d
y → price
1
2
3
4
n
x_i
y_i
x^i = $[x^i_1, x^i_2 .... x^i_d]$
(m,d)
X
m → no. of samples
d → no. of features
n → no. of classes
$i^{[th]}$ sample → $x^{[i]}$
$j^{[th]}$ feature → x_j
True o/p → $y^{[i]}$
Predicted o/p → $\hat{y}^{[i]}$

Don't get intimidated by the terms, we will go through all of them as we proceed with this module.

What is the goal of the ML model ?

Ans: To find $f: X \rightarrow y$ such that $f(x_i) \approx y_i$

How to define function f ?

Algebraic Intuition → find $\hat{y} = f(x_i)$, we can say for Linear Regression:

$$f(x_{i1}, x_{i2}, \ldots, x_{id}) = w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3} + \ldots + w_d x_{id} + w_0$$

$$\hat{y}_i = f(x_i) = \sum_{j=1}^{D} w_j x_{ij} + w_0$$

Now, $w^T = [w_1, w_2 \ldots w_d]$ and $x_i = [x_{i1}, x_{i2} \ldots x_{id}]$, then:

$$\hat{y}_i = f(x_i) = w^T x_i + w_0$$

How does the ML model find the function f ?

Ans: By updating weights of the model on the training dataset

How does the ML model update weights ?
Ans: By finding the gradients of the weights with respect to the loss function and by subtracting that gradient from the weights.

Is $f(x_i)$ in Linear Regression analogous to $y = mx + c$?
Ans: Yes, it is.

Linear Regression: finding a best D Dimensional hyperplane that fits the D-Dimensional data such that $\hat{y}_q \approx y_q$

How to find the best fit line of Linear Regression model ?
Ans: By optimizing the weights vector $W^T = [w_1, w_2, \ldots, w_d]$ wrt the loss function.

How to say Linear Regression is optimized ?
Ans: when we see the loss function is not decreasing anymore i.e. local minima

What loss function to use for linear regression optimization ?
Ans: Mean Square Error → finds the mean of the square difference between $\hat{y}$, y .

$$min_{w, w_0} \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

How to find optimal weights for Lin. Reg. ?
Ans: Gradient Descent → minimizes the Mean Squared error to reach global minima

How to find the Gradients of Mean Square Error ?
Ans: We define loss function as :

$$L(w, w_0) = \frac{1}{n}\sum_{i=1}^{n} (y_i - (w^T x_i + w_0))^2$$

On finding gradients with respect to w, loss becomes:

$$\frac{\partial L(w,w_0)}{\partial w} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial (y_i - (w^T x_i + w_0))^2}{\partial w} = \frac{2}{n}\sum_{i=1}^{n} (y_i - (w^T x_i + w_0)) \frac{\partial (y_i - (w^T x_i + w_0))}{\partial w}$$

As we know $\frac{d(uv+c+a)}{du} = v$, hence on simplifying, the equation becomes:

$$\frac{\partial L(w,w_0)}{\partial w} = \frac{2}{n}\sum_{i=1}^{n} (y_i - (w^T x_i + w_0)) (- x_i)$$

Similarly gradient for w_0 becomes:

$$\frac{\partial L(w,w_0)}{\partial w_0} = \frac{1}{n}\sum_{i=1}^{n} \frac{(y_i - (w^T x_i + w_0))^2}{\partial w_0} = \frac{2}{n}\sum_{i=1}^{n} (y_i - (w^T x_i + w_0)) \frac{\partial (y_i - (w^T x_i + w_0))}{\partial w_0}$$

$$\frac{\partial L(w,w_0)}{\partial w_0} = \frac{2}{n}\sum_{i=1}^{n} (y_i - (w^T x_i + w_0))(- 1)$$

Updating weights (w, w_0) with a learning rate α :

$$w = w - \alpha \times \frac{\partial l(w,w_0)}{\partial w}$$

$$w_0 = w_0 - \alpha \times \frac{\partial l(w,w_0)}{\partial w_0}$$

Why use Learning Rate ?
Ans: Learning Rate α → hyperparameter to control rate at which Gradient Descent reach global minima

What happens if a too small value of Learning Rate(α) is used ?
Ans: makes Gradient Descent reach the global minima **very slowly**

What happens if a too large value of Learning Rate(α) is used ?
Ans: may make the Gradient Descent **overshoot** the global minima

What will be the simplest model for predicting a value ?
Ans: Mean model → the mean of the entire data as its prediction.

After training the model, how to measure model performance ?

Ans: R-Squared **metric**. → measures the performance of Linear Regression over a mean model. It is Defined as:

$R^2 = 1 - \frac{SS_{res}}{SS_{total}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$**,** where $\bar{y}_i$ is the mean model.

SS_{res} - Squared sum of error of regression line
SS_{total}-(total sum of squares) Squared sum of error of mean line

What will be the best value of R^2 ?

Ans: 1, when $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = 0.$

What will be the minimum value of R^2 ?

Ans: - ∞ , when $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 >> \sum_{i=1}^{n}(y_i - \bar{y}_i)^2$

What happens to R-squared if we add a new feature ?
Ans: if the feature is a relevant, R-square ↑.

But if the Feature is not relevant,
R-square should ↓ when model performance gets worse →
- Model's task is to minimize the loss
- So, if adding a feature is reducing the performance i.e. increasing the loss
- Model can simply assign small or zero weights to new features in order to avoid decrease in performance.

Will the R-Square value increase or remain the same if we add a new feature?
Ans: Both are possible → small weights or zero weight can be assigned to new feature which will keep performance the same.
model performance
Model can start making spurious associations with new features (i.e. overfit) causing model performance to increase on the train set.

As R-Square fails to compare performance and model complexity (no of features), What other metrics to use ?

Ans: Adjusted R-Squared, defined as:

$$AdjR^2 = 1 - [\frac{(1-R^2)(n-1)}{(n-d-1)}],$$

where n is the number of samples, and d is the number of features

How does Adj R-Square compare performance and model complexity ?

Ans: if number of features (d) increases

with no significant feature:

- R^2 remains constant or slightly increased ⇒ (n-d-1) ↓ ⇒ Adj-R-squared ↓

with significant feature:

- R^2 ↑ significantly ⇒ $[\frac{(1-R^2)(n-1)}{(n-d-1)}]$ ↓ ⇒ Adj-R-squared ↑

How to determine which features impact the model most during prediction ?
Ans: The feature with highest weight → most important feature

What does the -ve sign mean in weight of model ?
Ans: The -ve sign means → if feature value ↑ , $\hat{y}$ ↓

Why perform column standardization ?
Ans: To bring all features to the same scale.
i.e. Removes ambiguity in feature importances

For example: if there are two features $f1, f2$:
- $f1$ value range >>> $f2$ value range → Weight of $f1$ >>> Weight of $f2$ ⇒ $f1$ important feature even though $f2$ is the important one.

Now Column Standardization makes : $f1$ value range ≈ $f2$ value range → Weight of $f1$ < Weight of $f2$ ⇒ $f2$ becomes important feature

Assumptions of Linear Regression :

a. **Assumption of Linearity**:
linear relationship between the features x and target variable y

b. **Features are not multi-collinear:**

What is collinearity ?
Ans: 2 features (f_1, f_2), have a linear relationship between them. $f_1 = \alpha f_2$

What is Multicollinearity ?
Ans: feature f_1 has collinearity across multiple features f_2 , f_3 , f_4

$$f_1 = \alpha_1 f_2 + \alpha_2 f_3 + \alpha_3 f_4$$

Why is MultiCollinearity a problem ?

Ans: MultiCollinearity → non-reliability on the feature importance and model interpretability.

How to resolve Multi- Collinearity ?
Ans: Using Variance Inflation factor (VIF), defined as:

$$VIF\ for\ f_j\ = \frac{1}{1-R^2_j};\ where\ R^2\ is\ Rsquared$$

VIF algorithm works as :

- Calculate VIF of each features

- if VIF >= 5 → **high Multicollinearity**

    - Remove feature having the highest VIF

    - Recalculate the VIF for the remaining feature

    - Again remove the next feature having the highest VIF

    - Repeat till all VIF<5 or some number of iteration is reached

c. **Errors are normally distributed:**

Used to ensure there are no outliers present in the data

d. **Heteroskedasticity should not exist:**

Heteroskedasticity → unequal scatter of the error term → not having the same variance

Why Heteroskedasticity is a problem ?
Ans: model inaccurate or outliers in the data.

How to check Heteroskedasticity ?

Ans: Plotting a Residual plot → Errors ($y - \hat{y}$) vs prediction ($\hat{y}$)

e. **No AutoCorrelation:**

What is AutoCorrelation ?
Ans: When the current feature value depends upon its previous value

Why is AutoCorrelation a problem ?

Ans: Linear regression assumes $\hat{y}_1 = f(x)$ has to be independent of $\hat{y}_2 = f(x+1)$ → AutoCorrelation contradicts this assumption.

Is there any other way to solve Linear Regression ?

Ans: Closed Form/ Normal Equation

Why use Normal Equations ?
Ans: Finds the optimal weights without any iterating steps as done in Gradient Descent.

The optimal weights: $W = (X^T X)^{-1} X^T Y$

Where X → feature matrix: $R^{n\ (Sample\ size)\ \times d\ (d\ dimensional)}$, and Y → target vector: $R^{n\ (Sample\ size)\ \times 1}$

Why even use gradient descent ?
Ans: $(X^T X)^{-1}$ → computationally expensive operation ⇒ Not used when the number of features is high.
Dimension of mat mul: [(d x n) * (n x d) => (d x d)]

What if there is non-linearity in the data , can Linear Regression work ?
Ans: No

What modifications can be done to Linear Regression for the model to be complex enough to fit non-linear data ?

Ans: By using **Polynomial Regression** → transforms linear equation of linear Regression to Polynomial equations

How does Polynomial Regression work ?
Ans: if Linear Regression has $\hat{y} = w_1 f_1 + w_2 f_2 + \ldots.. w_d f_d + w_0$;

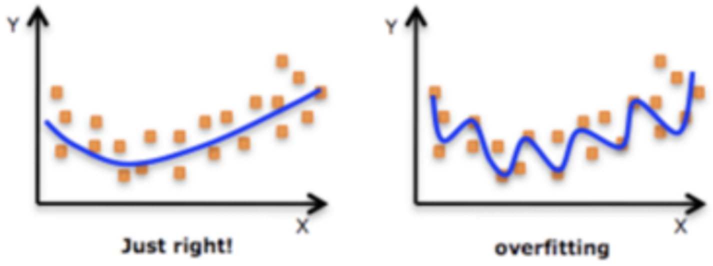
polynomial introduces terms like $f_1^{\ 2}, f_3 = f_3^{\ 4}, f_2^{\ 2} = \alpha f_2 + \alpha_0$

making the Model complex for handling non-linearity.

- **Bias-Variance,Regularization:**

If model A : covers all datapoints with high degree features
i.e. predicting hyperplane passes through all the datapoints

Model B : misses out only a handful datapoints using lower degree features,
I.e. predicting hyperplane misses a handful of datapoints



Then which model is better ?

Ans: Model B generalizes on the data → model captures the pattern of the data and not get influenced by Outliers (hence those points are missed out)

Model B , a simpler model than Model A →**Occam's Razor**

What can we say about model A ?
Ans: Model overfits the data → fitting to outliers/noises in the data

When to say model underfits the data ?
Ans: When the model is not able to predict most of the datapoints in the data → model has poor performance.

How is data split ?
Ans: Training, Validation and testing dataset.

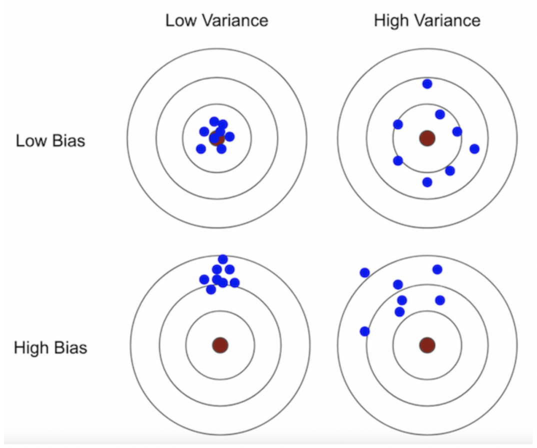
How is training and test data related to underfit and overfit model ?
Ans: an underfitted model → has a high training and test loss
- An Overfitted model → very low training loss but a high testing loss.

What is a suitable model ?
Ans: a tradeoff between both such tha model has a low training and testing loss → perfectly fit model

We can understand Underfit and overfit using Bias and Variance.

What do we mean by Bias and Variance ?

Ans: Understanding bias- variance with a target shooting example



Observe
- High Bias → have a wrong aim
- High Variance → an unsteady aim.

How is underfit related to Bias and Variance ?
Ans: Now in Underfitting → predictions are consistent but are wrong→ for different training sets → **High Bias and low Variance**

How is overfit related to Bias and Variance ?
Ans: Now in Overfitting → predictions vary too much and are wrong → for different training sets → **Low Bias and High Variance.**

How to control Underfit Overfit tradeoff to find the perfect model ?

Ans: Regularization in the loss function → adds a term $\sum_{j=1}^{d} w_j^2$ → making weight small → for insignificant features

How does Regularization make weights small for insignificant features ?
Ans: With optimization algorithm, → minimizes the values of w_j

$$TotalLoss = \min_{w_j} Lossfunction + \lambda \sum_{j=1}^{d} w_j^2$$

How to control Regularization ?
Ans: By using regularization parameter λ:

since too much regularization → makes the model underfit the data

Too little regularization → makes the model overfit.

Thus ⇒ λ becomes hyperparameter → on tuning gives the overfit-underfit tradeoff

Is squaring of weights the only way for Regularization ?

Ans: No, Regularization is majorly of three types:

A. **L1 / Lasso Regularization :** Uses the term $\sum_{j=1}^{d} |w_j|$ → has $\frac{d|w_j|}{w_j} = 0$, when $w_j = 0$ → making the **weight vector sparse.**

B. **L2/ Ridge/ Tikhonov regularization** : Uses the term $\sum_{j=1}^{d} w_j^2$ → have close to 0 values → for insignificant features.

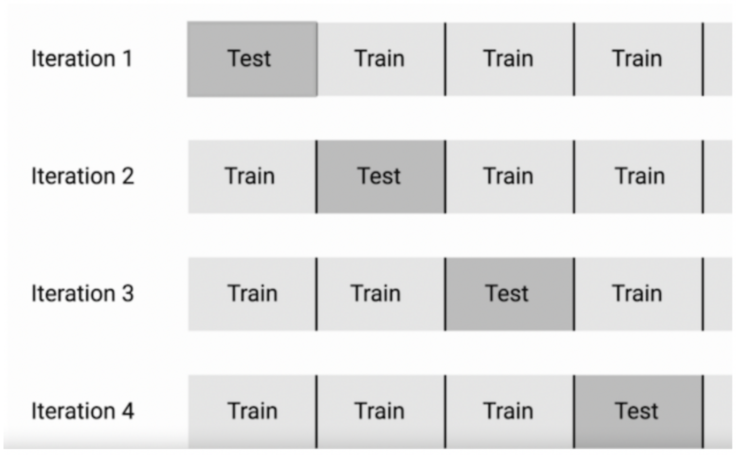
C. **ElasticNet Regularization:** Combination of both L1 and L2 Regularization → with λ_1 and λ_2 as regularization parameters respectively.

$$TotalLoss = \min_{w_j} Lossfunction + \lambda_1 \sum_{j=1}^{d} w_j^2 + \lambda_2 \sum_{j=1}^{d} |w_j|$$

Why split data into Validation dataset ?
Ans: hyperparameter tuning → done only on Validation data → test data solely used for evaluating the model on unseen data.

What are the steps for a model building ?
Ans: The steps are :
- Train model → with some regularization parameter λ → on training data
- Measure the model performance → with different value of hyperparameters → on the Validation dataset
- Pick the hyperparameters of the best performing model
- Measure the performance of the Best performing model on Test data.

If data is too small to have a validation dataset, what to do then ?
Ans: use **k-Fold CV algorithm** since**:**
- splits data into k smaller sets
- for each iteration, the model trained on k-1 folds
- validated on 1 fold
- performance is avg over all the iterations.



Note: Though k-Fold is a computationally expensive algorithm, it is useful when dataset is small.

- **Logistic Regression**

Why do we need Logistic Regression ?
- Useful for binary classification

What are the assumptions of Logistic regression ?
- Data should be linearly separable

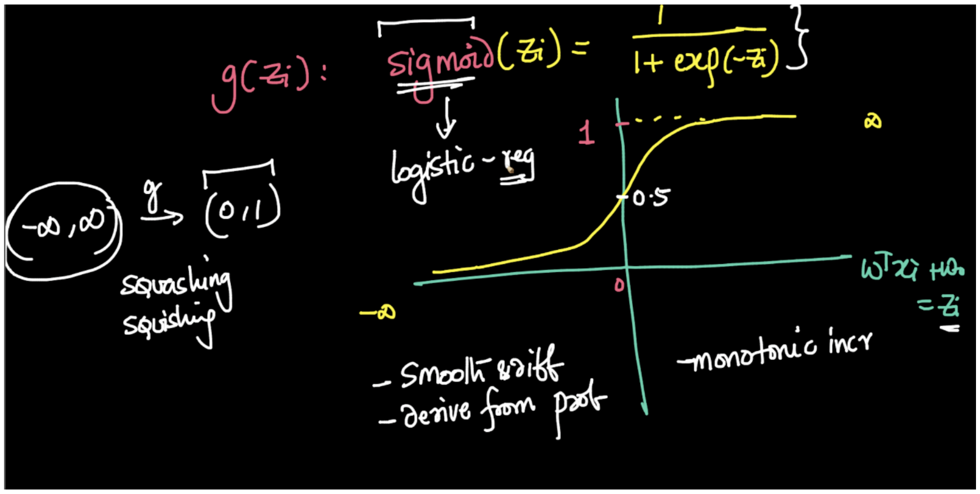
What is the goal of this algorithm ?
- To find a hyperplane π which accurately separates the data

How to perform prediction through logistic regression ?

- Given labels -> $y_i \epsilon \{0, 1\}$
- Compute a linear function of x -> $z_i = w^t x_i + w_0 \epsilon \{-\infty, \infty\}$
- Compute Sigmoid(z$_i$) -> $\sigma(z_i) = \frac{1}{1+e^{-z_i}}$
- Predicted label -> $\hat{y}_i = 1 \; if \; \sigma(z_i) > threshold \; else \; 0$

What are the properties of Sigmoid function ?

- Range -> (0, 1)
- Smooth and differentiable at all points



What is the derivative of a logistic regression model ?

$$\sigma'(z) = \sigma(z) \times [1 - \sigma(z)]$$

What is the significance of the sigmoid function ?

- $\sigma(z_i)$ is the probability of x$_i$ belonging to class 1

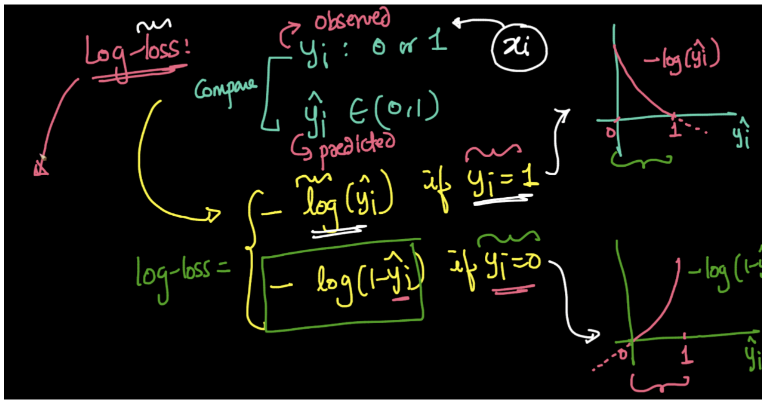
Which loss function is used for Logistic Regression ?

- **Log loss ->** Combination of:
  - $-\log(\hat{y}_i)$ when $y_i = 1$
  - $-\log(1 - \hat{y}_i)$ when $y_i = 0$.

$$Log\ Loss = -y_i \log(\hat{y}_i) - (1 - y_i)\log(1 - \hat{y}_i)$$

Total loss becomes:

$$TotalLoss = Log\ Loss + \lambda \sum_{j=1}^{d} w_j^2$$



How does log loss help train logistic regression model ?

- $-\log(\hat{y}_i)$ -> High when $\hat{y}_i = 0$ and very low when $\hat{y}_i = 1$.
- $-\log(1 - \hat{y}_i)$ ->High when $\hat{y}_i = 1$ and very low when $\hat{y}_i = 0$.

Thus both of these components help penalize the model most when doing wrong predictions.

Also if $y_i \epsilon \{-1, 1\}$, then $Log\ Loss = \sum_{i=1}^{n} log(1 + e^{-y_i z_i})$

But why can't we use Mean Square Error as in Linear regression ?

- Non-convex curve : contains a lot of local minimas
- Difficult for Gradient Descent to reach global minima

What does the derivative of Log Loss look like ?

$$\frac{\partial Log\ Loss}{\partial w} = \frac{\partial(-y\log(\hat{y}) - (1-y)\log(1-\hat{y}))}{\partial w}, \hat{y} = \sigma(z) \ and\ z = w^t x + w_0$$

Intermediate steps : reference blog

On solving:

$$\frac{\partial Log\ Loss}{\partial w} =- y\frac{\partial z}{\partial w} + y \times \hat{y}\frac{\partial z}{\partial w} + \hat{y}\frac{\partial z}{\partial w} - y \times \hat{y}\frac{\partial z}{\partial w}$$

$$\frac{\partial Log\ Loss}{\partial w} =- y\frac{\partial z}{\partial w} + \hat{y}\frac{\partial z}{\partial w}$$

Also since $\frac{\partial z}{\partial w} = x$; $\frac{\partial Log\ Loss}{\partial w} = (\hat{y} - y)\, x$

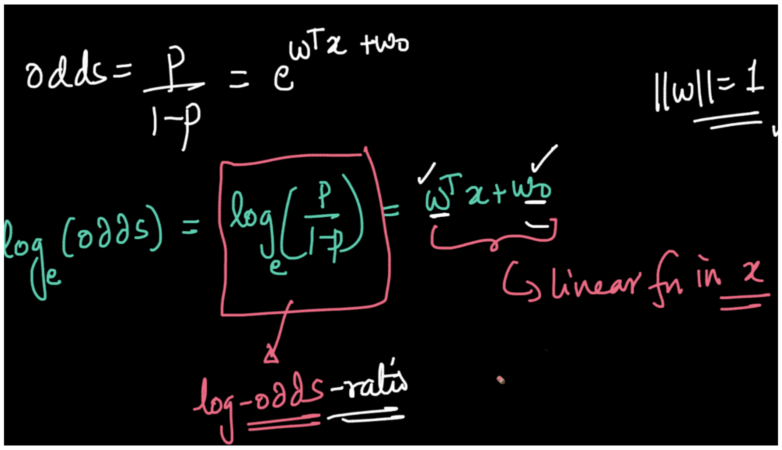
What if we want to predict odds of $y_i = 1$ vs $y_i = 0$?

- **Log - odds:** Shows how the model is similar to a linear model which is predicting log-odds of $y_i = 1 \; vs \; y_i = 0$, defined as **:**

$$log_e(odds) = \log[\frac{p}{1-p}] \; ; \; p = \frac{1}{1+e^{-z_i}} = \frac{e^{z_i}}{e^{z_i}+1} \; and \; 1 - p = \frac{1}{e^{z_i}+1}$$

On substituting the value of p and 1-p , and solving it we get

$$log_e(odds) = \log_e[e^{z_i}] = z_i = w^t x_i + w_0$$



$odds = \frac{p}{1-p} = e^{w^Tx + w_0}$
$\|w\| = 1$
$\log_e(odds) = \boxed{\log_e\left(\frac{p}{1-p}\right)} = \underbrace{w^T x + w_0}$
linear fn in x
log-odds-ratio

Note: Hence the name Regression in Logistic Regression.

Is there any way to use Logistic Regression for multiclass classification ?

- **One vs Rest Method:**
  - For $y_i = \{1, 2, 3......, K\}$, generate k-binary logistic Regression models
  - Final prediction -> Argmax of all of the predictions made by each models

---

- **Classification Metrics**

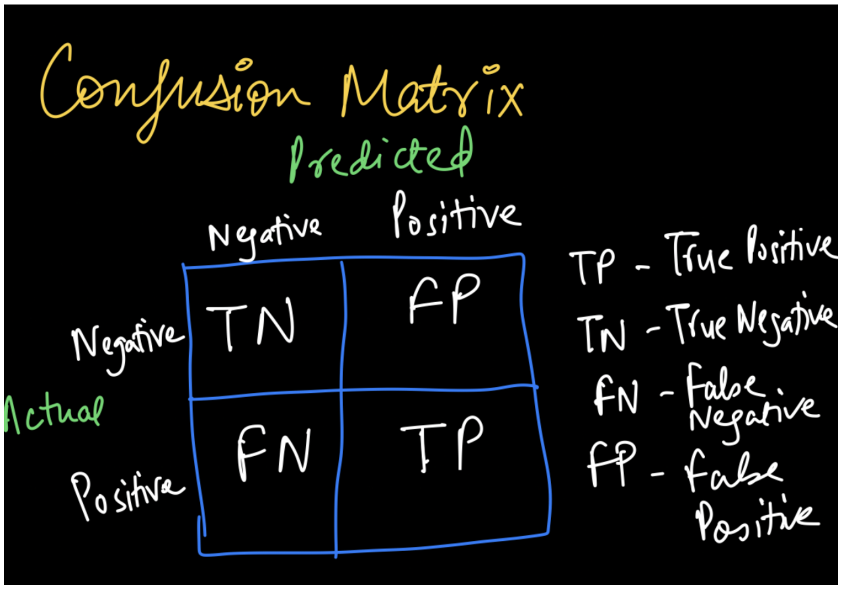
Issue with Accuracy ?
Ans: Accuracy metric fails when data is imbalanced.
- Consider there are 90 data points of class 1 and 10 data points of class 0.
- If the model predicts every datapoint as class 1.

⇒ The accuracy of the model is at 90% , which is completely wrong.

What other metric to use ?
Ans: Confusion matrix

When a model predicts, there can be 4 scenarios:

A. **True Positive (TP):** Model predicts True, is Actually True
B. **False Positive (FP):** Model predicts True, is Actually False
    - Aka Type 1 Error
C. **True Negative (TN):** Model predicts False, is Actually False
D. **False Negative (FN):** Model predicts False, is Actually True
    - Aka Type 2 Error



Note:

- For a dumb model that predicts everything as negative, FP = TP = 0
- For an Ideal model that has no incorrect classification, FP = FN = 0

How to use CM to determine if a model with high accuracy is actually good?

Ans. High accuracy can be deceiving. It is only a good model if:

- Both TP and TN should be high
- Both FP and FN should be low

Given a CM, how can we calculate actual positives?

Ans. TP + FN = P (total actual positives)

⇒ Similarly, FP + TN = N (total actual negatives)

How to calculate accuracy from the confusion matrix ?

Ans: Accuracy: $Accuracy = \frac{Correct\ Predictions}{Total\ Number\ of\ Predictions} = \frac{TP+TN}{TP+TN+FN+FP}$

Which metric to use, when we cannot afford to have any false positives?

Ans: Precision: It tells us out of all points predicted to be positive, how many are actually positive.

$Precision = \frac{TP}{TP+FP}$

For ex

- Misclassification of a spam email as not spam is somewhat acceptable i.e FN
- However, classifying an important mail as spam can lead to major loss i.e. FP

⇒ i.e reducing FP is more critical.

What is the range of precision values?

Ans. Between 0 to 1.

Which metric to use, when we cannot afford to have any false negatives?

Recall / Sensitivity / Hit Rate: It tells us out of all the actually positive points, how many of them are predicted to be positive

$Recall = \frac{TP}{TP+FN}$

For ex

- Classifying a healthy person as cancerous and carry out further testing is somewhat acceptable
- However, classifying a person with cancer as healthy can be life-death situation.

⇒ Here, reducing FN is more critical.

Note:

- **True Positive Rate (TPR):** $TPR = \frac{TP}{TP+FN}$; is same as Recall

What is the range of recall values?

Ans. Between 0 to 1.

What other metrics to look for ?

- True Negative Rate (TNR): $TNR = \frac{TN}{FP+TN}$

- TNR tells out of all the actual negative points, how many have been predicted as False .

- Also called as **Specificity / Selectivity**

- False Positive Rate (FPR):$FPR = \frac{FP}{FP+TN}$

- Intuitively, it tells, out of all data points which are actually negative, how many are misclassified as positive

- False Negative Rate (FNR):$FNR = \frac{FN}{FN+TP}$

- Intuitively, it tells, out of all data points which are actually positive, how many are misclassified as negative

Which metrics are used in the medical domain?
They are used to measure how good a test is at correctly identifying the presence or absence of a disease.
In medical terms,

- **Sensitivity** : proportion of people with the disease who test positive for it
  - ⇒ Test is good to be used as screening test
  - ⇒ There is low chance of missing out a person with disease (low FN)
- **Specificity** : proportion of people without the disease who test negative for it
  - ⇒ It means that test is good for confirmatory test
  - ⇒ There will be low FP

What is F1 score? When is it used?
When both Precision and Recall is equally important measure for the model evaluation, then we use F1-Score:

$$F1\ Score = \frac{2\times(Precision \times Recall)}{Precision + Recall}$$

For ex, in a fintech company, giving out loans,

- Loss to business if they give loan to people who are unable to repay it (FP)
- Also a loss if they miss out on good people who will be able to repay (FN)

⇒ Here, we need to focus on both FP and FN.

Note:

- F1-score is just the Harmonic mean of Precision and Recall.
- F1-score Is also Useful when data is imbalanced.
- Range - [0, 1]

Why do we take harmonic mean in F1 score instead of arithmetic mean?
Harmonic Mean penalizes the reduction in Precision and Recall more than Arithmetic Mean.

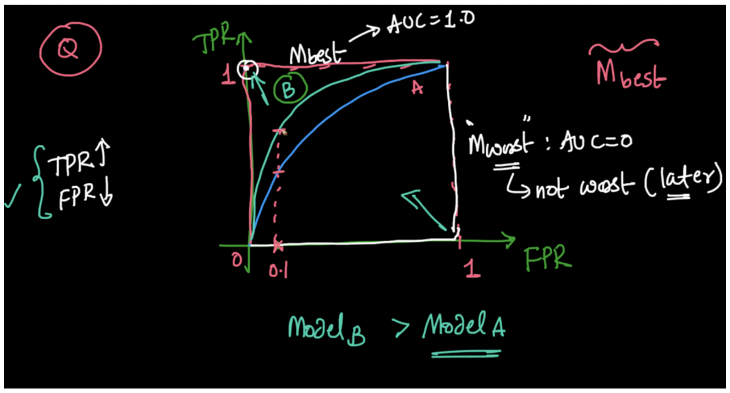
Can we adjust F1 score to give more preference to precision or recall?
A beta parameter can be added to make F-1 Score have more attention on either Precision score or Recall score.

$$Fbeta = \frac{(1+beta^2) \times Precision \times Recall}{beta^2 \times Precision + Recall}$$

- Beta = 2 if Recall more important than Precision.
- Beta = 0.5 when Precision is more important.

What is AU-ROC? Where is it used?
- AU-ROC : Area Under Receiver Operating Characteristic Curve
- Used to find the best model by plotting TPR and FPR by sorting value of $\hat{y}_i$ and keeping them as threshold for final prediction (y_{pred}).

How do we determine the better model using AU-ROC?
After plotting whichever curve has the most area covered tends to be the better model.
For ex:
- Here, AUC of model B > model A
- Hence, model B is better



Note:
- Unlike precision, recall or F1-score, AU-ROC does not work well for highly imbalanced data.

What is the fundamental difference between AUC and the other metrics?
When we calculate Precision, Recall or F1 score

- We calculate it for a certain threshold on $\hat{y}_i$
- This threshold is 0.5, by default

On the other hand, for AU-ROC

- we are calculating it using all possible thresholds

What will be the AUC of a random model?

The ROC curve will be diagonal. ⇒ Hence AUC will be 0.5

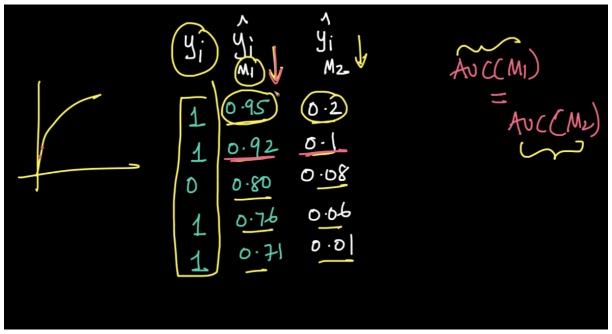
What to do if a model's AUC < 0.5 ?

A simple fix is to invert your predictions.

⇒ After inverting, you will get area of 1-(actual area value)

Does AUC depend on the actual values of $\hat{y}_i$?

No. AU-ROC depends on the how the ordering of the $\hat{y}_i$ is done and not on the actual values of $\hat{y}_i$.



For Example, say we have two models M1 and M2

- Actual y labels: [1, 1, 0, 1, 1]
- $\hat{y}_i$ for M1: $[0.95. 0.92,\ 0.80,\ 0.76,\ 0.71]$
- And $\hat{y}_i$ for M2: $[0.2,\ 0.1,\ 0.08,\ 0.06,\ 0.01]$

⇒ Since both have the same ordering of how $\hat{y}_i$ are arranged , hence AUC(M1) = AUC(M2).

AU-ROC is highly sensitive to imbalanced data, what metric can we use there?
We can use Area under the Precision-Recall cure (AU-PRC).

This is a very good metric for imbalanced data.

How is PRC plotted?

- Precision on y axis
- Recall on x axis
- Similar to ROC curve, we'll take each $\hat{y}_i$ as threshold

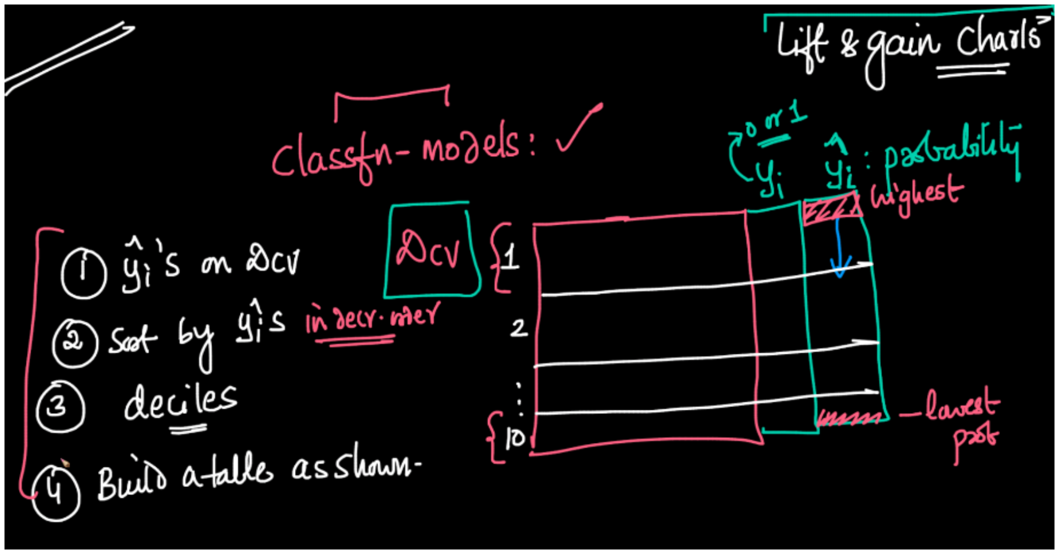
Then we take the area under the PRC curve to get AU-PRC.

Which metric can we use to know how a model would do business wise?

- Business people need to know how our model's differences would make in the business term compared to random targeting.
- ⇒ Lift and Gain charts are used.
- ⇒ Help us graphically understand the benefit of using that model (in layman terms)

How to make lift and gain charts?

- **Step 1:** Obtain the predictions $\hat{y}_i$ on cross-validation data Dcv
- **Step 2:** Sort the data predicted probability in desc order We will have highest probability at the top, lowest at the bottom.
- **Step 3:** Break the sorted cross validation data into 10 groups (or deciles). We get the data for each decile.
- **Step 4:** Using these deciles, we built a table as follows
  - ⇒ 1st decile will have datapoints with the highest pred. probability
  - ⇒ 10th decile will have datapoints with lowest pred. probability



Calculation of Gain:

- Dividing the cumulative responses by total number of responses

- i.e. cumulative number of positives by total number of positives.

Calculation of lift:

- Dividing total % of positive point up to that decile (gain) for a smart model by total % of positive points if we had a random model
- in other terms, it is ratio of gain of our model to a gain of random model

⇒ After calculating the lift and gain values, we plot them for each decile.

What does gain mean?

Gain for ith decile tells us "what percentage of positive points are in i th or smaller decile"
For example:

- If gain is 97.87 % for the 8th decile
  ⇒ It means we cover 97% of positive datapoints, till the 8th decile.

What does lift mean?

- It means Cumulative percentage of positive points till i th decile divided by cumulative percentage of positive points by random model
- It is intuitively telling how much better a model is compared to a random model.

How to use Accuracy metric even when data is imbalanced ?

Ans: G-Mean: When data is imbalanced, Geometric-Mean(G-Mean) measures model performance on both the majority and minority classes.

$$GMean = \sqrt{Specificity \times Sensitivity}$$

Which metric to use when? (Cheat Sheet)

- If we want probabilities of classes: **Log loss**
- If classes are balanced: **Accuracy**
- IF classes are imbalanced:
  - If FP is more critical: **Precision.**
  - If FN is more critical: **Recall.**
  - **F1 score** is a balance between precision and recall.
  - If our concern is both classes (TN and TP)**: ROC_AUC**
- If severe imbalance**: PR AUC**

How are performance metrics different from loss functions?

- Loss functions are usually differentiable in the model's parameters.
- Performance metrics don't need to be differentiable.
- A metric that is differentiable can be used as a loss function also. For ex: MSE