**Machine Learning**

*AI vs ML vs DL*

AI: Train machine to perform task without human intervention

ML: subset of AI to analysis and predict the data based on Algorithm

DL: subset of ML, used to mimic human brain by multi layers neural network

**Supervised:** Classification Regression

**Unsupervised:** Clustering Dimension Reduction

**Supervised:** One dependent and many independent features

* **Linear regression:** output will be continuous
* **Classification:** when the output has fixed number of outputs (Ex: Binary classification)

**Unsupervised**: No dependent variable

* **Clustering**: Similar groups are identified and clustered/Grouped (Ex: customer segmentation)
* **Dimensionality Reduction**: Convert 1000 features to 100 (PCA)

**Linear Regression**

y= mx + c m= slope c= intercept

Draw the best fit line: so that the error should be minimum (summation of difference between predicted and actual should be minimum)

Cost function: Squared error function: we should minimize the cost function by adjusting slop and intercept.

Gradient descent: the line drawn by taking different cost functions and x values.

Global Minima: gives the best fit line. Reach towards global minima. Convergence algorithm is used to find the global minima.

Learning rate (alpha): to find the global minima.

Repeat gradient descent until convergence

**R square and Adjusted R square:** check how good the model is performing.

If i got R square 91%. If i add another feature 93%, if i add another feature then R square will increase. If we add features (ex: male and female in data which gas no impact on data) R square keep on increasing. In order to correct this, we use adjusted R square.

Adjusted R square includes the features by taking R square. Adjusted R square will always be smaller than R square. Because in Adjusted R square formula the number of features are in denominator so the value of adjusted R square will always be smaller

**Ridge and Lasso Regression**

The main aim of linear regression is to minizine the cost function

Overfitting: Model performs well with training data and fail to perform well in test data (Low bias and high variance)

Underfitting: Model fails to perform In test as well as training data (High bias and High variance)

**Model 1 Model 2 Model 3**

Training 90 Training 92 Training 70

Test 80 Test 91 Test 65

Overfit Generalized Underfit

*To reduce overfitting, we use ridge and lasso*

**Ridge (L2 regularization)** add the parameters **Lamda (slope)2** to the cost function Lamda is the hyperparameter to check how steep the slope is.

**Lasso (L1 regularization)** add the parameters **Lamda |slope** **|** (mode of slope)to the cost function. mode of slope helps in feature selection. The slope value for the feature which are not important will be very low, those features will be neglected.

Lasso prevent overfitting and feature selection

Iterations is also a hyperparameter. Lamda is also a hypermeter. We should try both the regularization and use where the performance is high.

**Assumptions of linear regression**

* Feature following normal or gaussian distribution then model will fit well
* Standard scaling using Z score mean =0 sd=1 (wherever gradient descent involved standardization is good)
* Works with respect to linearity of data.
* Multicollinearity

To optimise the model standardization is good

**Logistic regression (Classification)**

Logistic regression well with binary classification we can also do multiclass classification

Sigmoid activation function is used to squash the liner regression line

*Performance matrix*

Confusion matrix:

|  |  |  |  |
| --- | --- | --- | --- |
| Predicted | Actual | | |
|  | 1 | 0 |
| 1 | TP | FP |
| 0 | FN | TN |

Accuracy = TP +TN / TP +FP + FN +TN

Precsion = TP / TP+ FP(out of all predicted positive values how many of them, are positive )

Recall = TP/ TP +FN (out of all positives how many of them as predicted as positive)

F1 score =

**Navis Bayes** (works good for classifications)

Works based on bayes theorem

**KNN (Both classification and regression)**

By selecting the neighbour points based on distance (Euclidian or Manhattan)

Euclidian:

Manhattan:

K is the hyperparameters

KNN is very bad at outliers and imbalanced data

**Decision Trees**

Root node: first node

Leaf node:

Pure split: Either yes or no only one outcome or pure leaf node

Impure split: both the outcome yes and no

Entropy:

Gini impurity:

Information gain:

**SVM**

The data is divided by hyperplane

**Ensemble techniques**

**Bagging**: We split the dataset by row sampling with replacement and each data is given to given to different model (KNN, decision tress etc…,) and each model will give the output, All the output are obtained and majority voting will be considered.

Bootstrap aggregation: aggregation of the outputs

**Random forest classifier Random forest regressor**

**Boosting:** sequential combination of model**,** data flows from one model to another, these models are called as weak learners, combine all the weak learners and get the output**.**

**Adaboost, Gradient boost, XGBoost.**

**Random Forest**

Decision tress leads to overfitting so we need to convert high variance to low variance

All the models are decision tress in random forest. Every decision tress will create low bias and high variance if we combine the tress, we will get low variance.

Decision tress does not require standardization but KNN requires

Decision tress not impacted by outliers but KNN is impacted.

**Boosting**

1. **Adaboost:**

Decision trees in one depth is called stump

1. **XGBoost**

**K-Means clustering**

In custom ensemble techniques we can use k means clustering

K =centroid we will try with different k values and identify the exact number of centroids based on the distance between centroid and points.

To decide the k value, we will use elbow method. K value vs wcss (within cluster sum of squares) in a graph

**Hierarchical clustering**

**DBSCAN (density based spatial clustering of application with noise)**

It helps when there are outliers in data. Outliers are considered as noise and not considered in clusters.

**Deep Learning**

Deep learning is becoming popular because of enormous data generated nowadays. In order to get better product, the generated data need to be utilized properly and better decisions.

**Perceptron**

Input layer: no of features

Hidden layer: number of neurons (there may be many hidden layers)

Output layer: output

We assign weights to each layer: multiply wights with each feature, weight helps the neurons at what level they should be activated.

If Weights ate initialized as zero or when weights are zero then when we multiply weight with input the sum becomes zero, so to avoid this we add bias.

**Activation function**: tells us weather the neuron should be activated or not (ex: sigmoid and ReLu)

**Loss function**: difference between actual and predicted, we should minimze loss function

**Optimizers**: helps to update the weights (ex: Gradient descent)

**Forward propagation**: assign weights randomly

**Back Propagation**: to update the weights

Sigmoid activation function used for binary classification.

To reduce the loss function we need to update the weights

**Back propagation**: Weight updating, in order to reduce loss function.

Weight updating formula (Wnew) = Wold – learning rate (derivative of loss/ derivative of Wold)

Learning rate should be a small number: so that we slowly converge to the global minima (0.01 or 0.001)

**Chain rule of derivative**: helps to update of weights

**Vanishing gradient problem**: weights are not updating during the derivative, the Wnew and Wold will be same so we use different activation functions.

**Different type of activation function**

At each neuron we apply activation function

1. **Sigmoid activation function:**

Smooth gradient prevent jump in output values, but output is not centred to zero

Prone to vanishing gradient problem

1. **Tanh:**

The value -1 to +1 and derivate value 0 to 1

And its s zero centric

1. **ReLu (commonly used)**

Rectified linear unit

0 to 1

1. **Leaky ReLu:**

Will have small negative values and solves dead neuron problem

1. **ELU** Exponential Liner Units
2. **Softmax**
3. **PReLu**

**Which activation function we should use.**

Sigmoid and Tanh Functions cause vanishing gradient problem

**Binary Classification**: In hidden layer always use ReLu and in Output layer use sigmoid

**Multiclass Classification**: In hidden layer always use ReLu and in Output layer use Softmax.

**Regression:** In hidden layer always use ReLu and in Output layer use liner activation function

**Loss Function:**

In loss function only one data point is provided whereas in cost function batch of data points is provided

1. **Regression:**

**MSE (Mean squared error)**: not robust to outliers

**MAE (Mean absolute error):** robust to outliers

**Huber loss**: Combination of MSE and MAE

1. **Classification:**

**Cross entropy: a**) Binary cross entropy

b) categorical cross entropy for multiclass classification

**OPTIMIZERS**

Helps to adjusting weight in back propagation to minimize the loss function

1. Gradient descent: to find the global minima
2. SGD (Stochastic gradient descent): each record as one iteration
3. Mini batch SGD: takes records in batches
4. SGD with momentum
5. Adagrad (Adaptive gradient descent)
6. RMSPROP
7. Adam

**Epoch:** one forward and one backward propagation is called as one epoch

**Iteration**: no of records passed to each epoch: one record passed at a time and it comes one iteration.

**Batch size:** if I have 1000 records, I set a batch size of 100 records for each epoch, so that the number of interactions can be reduced.

**Adam optimizer**: momentum + RMSPROP (Adaptive learning rate) Commonly used optimizer.

The algorithms which include distance based and gradient descent we need feature scaling (dealing with categorical values)

**ANN in TensorFlow**

*fit\_transform* applied only to train dataset and not test data, to avoid data leakage and overfitting.

Import sequential: the entire neural network is called sequential (Block of neurons)

Import dense: we will create hidden output and input layers

Import Dropout: to drop the certain neuron in each layer to avoid overfitting. If 30% means 30% of neurons will get deactivated.

**Early stopping**: stop the epochs at the level which accuracy will not improve.

**CNN:**

**What are images?**

Black and white (Single channel) or RBG

The image is divided into pixels and each pixel have particular value from 0 to 255

0 Black 255 white

RGB three channels and each channel have value of 0 to 255 for each pixel.

**Convolution:**

The values of the pixels are brought between 0 to 1 (min- max scaling). The image is passed through the filter/kernel (Horizontal edge filter/Vertical edge filter)

Stride: when the filter jumps from one position to other, 1= one step jump

n = 6 size of image filter = 3 (3\*3)

Output image = n - f + 1 4

The image size is decreasing, it should not decrease, to avoid this padding is done (adding another layer on the top of the image)

To prevent the information loss, we apply padding

After padding = n + 2p – f + 1 / S p= padding and S =stride

Based on the input image we will update the filters. For each and every value of the output we apply the ReLu activation function.

**Max Pooling**

To extract more clear information the highest value obtained in each stride will be used and updated (Min pooling, Average Pooling, Other pooling)

Convolution and max pooling used horizontally we can stack it any number of times.

**Flaten:** like ANN dense layer, the input from max pooling is flattened/elongated.

After flatten fully created layer is created.