**Unit 4 → Data Pre-processing**

# **Machine Learning Model :**

1. Data Collection
2. Data Preparation : data pre-processing
   1. Check for gaps (missing values, blanks, nulls).
   2. Handling the gaps :
      1. Dropping
      2. Imputing (Replacing with meaningful values )
         1. fiilna(), interpolate(),
         2. Build a regression model and predict the missing values.
   3. Feature Scaling : If attribute ranges vary largely, bring them to similar ranges.
   4. Identifying extreme values & treating them(outliers).
   5. Convert strings to numeric values ;
      1. Categorical to Integer : Data encoding
      2. String to Integer : Word embedding
   6. Dimensionality reduction
3. Building the model – Training the model.
4. Evaluate the performance of the model.
5. Tune the model till satisfied.
6. Use the model for prediction.

### 

### **Handling the missing values using a dataset and applying Linear Regression :**

###### [Handling Missing Values Problem](https://colab.research.google.com/drive/1BmSISYClDdMALBglaN3ly2TebZ3IVVpA?usp=sharing)

### **Handling the missing values using a dataset and applying Simple & KNN Impute :**

[Simple Impute & KNN Impute](https://colab.research.google.com/drive/1vuO2CaUbgBixC6gl0ktdtqZlFSayorrB?usp=sharing)

**Data Normalization / Feature Scaling :**

1. When feature ranges vary largely, they need to be brought down to similar ranges.
2. To improve the model performance through predicting more accurate values.

Methods for

1. Minmax Scaling
2. Standard scaling / Z Score Transformation
3. Robust Scaler
4. Log Transformation

To perform scaling, you write your own methods or make use of Sklearn libraries. This process comes under the pre-processing step of model building.

1. **Minmax Scaling :**
2. Feature values are brought to the range from 0-1.
3. **(value-min) / (max-min)** → Apply on each column.

**User defined Function & SKLearn Library:**

[Data Normalization](https://colab.research.google.com/drive/1w8ivBUW4nOz9zOH069kGIYbdhE6H_v8B?usp=sharing)

**2. Standard scaling:**

1. Transforms each feature to have mean of 0 & Standard deviation of 1.
2. X - Mean / Std.Deviation
3. With standard scalers, the ranges are fixed to a particular range unlike minmax scaling.

**User defined Function & SKLearn Library :**

[Data Normalization](https://colab.research.google.com/drive/1w8ivBUW4nOz9zOH069kGIYbdhE6H_v8B?usp=sharing)

**3. Robust scaling:**

1. First outliers are removed & then scaling is done.
2. So the data is robust to outliers.

**Implementation :**

[Data Normalization](https://colab.research.google.com/drive/1w8ivBUW4nOz9zOH069kGIYbdhE6H_v8B?usp=sharing)

**4. Log Transformation:**

1. The larger values are scaled down to smaller values.
2. The skewness of the data is taken care of.
3. It cannot be applied if data contains negative values.

**Implementation :**

[Data Normalization](https://colab.research.google.com/drive/1w8ivBUW4nOz9zOH069kGIYbdhE6H_v8B?usp=sharing)

**Inverse Transformation :**

After scaling down, you can get back the original data using Inverse Transformation method on Normalized method.

**Implementation :**

[Data Normalization](https://colab.research.google.com/drive/1w8ivBUW4nOz9zOH069kGIYbdhE6H_v8B?usp=sharing)

**Data Transformation :**

Handling Categorical Data

1. **Data Encoding** : Convert categorical to numeric.
2. **Word Embedding** : Convert text to numeric.

**Methods in Pandas :** df.replace()

1. **Pandas Replace method** :

Numeric values are assigned to categorical values in alphabetical order.

1. **Get dummies** : pd.get\_dummies()

* It returns a dataframe with no. of columns == no. of unique values.
* Df contains 0 & 1.
* It increases the dimensionality of the dataset, as there are more no. of unique values.
* Most of the values are 0’s thus creating a **Sparse Datasets**.

[Handling Categorical Data](https://colab.research.google.com/drive/1OQ8nX8bn3kgMiomsR0JwTf2j_JhTbCpI?usp=sharing)

**SKLearn Methods for Data Encoding:**

1. **Label Encoding :**

Numeric values are assigned in alphabetical order.

1. **One-Hot Encoding :**

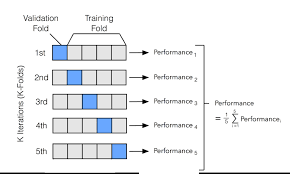
[SKLearn Methods](https://colab.research.google.com/drive/1OQ8nX8bn3kgMiomsR0JwTf2j_JhTbCpI?usp=sharing)

* In KNN, k-value is chosen by the user. Choose the value of k that gives best model performance.
* Similarly, in the Decision Tree, depth is chosen by the user.
* In K-Means also, the best value of k must be chosen.
* K, Depth, Kernel in SVM etc… are called **Hyperparameters.** You have to choose the best hyperparameters values.
* **How to choose the best hyperparameter values?**
* This is called **Hyperparameter Tuning**.

**Model Development :**

1. Data Collection
2. Data Pre-processing
3. Build the model - This includes training phase
4. Test the Model.
5. If not satisfactory, improve the model performance in ;
   1. Hyper Parameter Tuning
   2. Use K-Fold Cross Validation :

**K-Fold Cross Validation :**



1. Use Set-A for testing and others for training - take accuracy score.
2. Use set-B for testing and remaining for training set & repeat for all the other sets.
3. Final Accuracy is the average of all the accuracy scores for individual sets.

The Model performance improves as the different data is used for training.

**K Fold Cross Validation & Stratified Cross Validation using Logistic Regression :**

[K Fold & Stratified Cross Validation Implementation](https://colab.research.google.com/drive/1qXwOhvtgu4UOXVxXAwHUci-GSMO03ENL?usp=sharing)

* Divide the dataset into K subsets.
* 1 subset is a test set and the remaining (K-1) subsets are used for training.
* The Avg. test accuracy for K-subsets is the final test accuracy.
* Preferred when the dataset size is small.
* The Variance problem of the model is taken care of.

**Dataset** is divided into **Train** + **Test** + **Validation**.

Train set is used for training the model to find the model parameters.

Test set is used for model evaluation.

Validation set is used for hyper-parameter tuning.

**Hyperparameter Tuning:**

1. **Manual Tuning :**

User chooses hyperparameter values by trial and error.

[Manual Hyper Parameter Tuning Using SVM](https://colab.research.google.com/drive/1Jjg3qgCVdL0iE3NgPEFvx3MfPFTaafSh?usp=sharing)

We used SVM. The Hyperparameters used are-

1. Linear
2. Polynomial with varying degrees
3. RBF

Finding out the best combination is tedious, so go for Automatic parameter tuning.

1. **Automatic Tuning :**
   1. Grid Search
   2. Randomized Search

Code Implementation :

[Grid & Randomized Auto Tuning](https://colab.research.google.com/drive/1u2iXrvT55YPolFWPJuu_Z5MBSAeRBVTO?usp=sharing)

**Ensemble Methods :**

Build multiple models and combine the predictions as a final prediction. Ensembling means combining.

Ensemble methods can be categorized into 2 types :

1. **Sequential method** :
   1. Multiple models are built one after the other.
   2. Next model is built based on the predictions of the previous one.
   3. Eg : Adaboost Algorithm.
2. **Parallel method** :
   1. Multiple models are built simultaneously.
   2. Eg : Random Forest Algorithm.

**How to perform the combining of the predictions ?**

* **In case of Regression :**

1. **Averaging** : Prediction of all N models is averaged to get the final prediction.
2. **Weighted Average** :

* **In case of Classification :**

1. **Hard Majority Voting** :

For a given datapoint,

Model 1 : ClassA

Model 2 : ClassB

Model 3 : ClassA

Final prediction for the datapoint is **ClassA**.

1. **Soft Majority Voting** :

Each model gives only the probability of being a particular class.

For a given datapoint,

A B

Model 1 : (0.7, 0.3)

Model 2 : (0.4, 0.6)

Model 3 : (0.2, 0.8)

Probability of ClassA =  = 0.43

Probability of ClassB = = 0.57

Final prediction is **ClassB**.

**More Advanced Ensemble methods are,**

1. **Bagging :**

* Divide the Train dataset into K-subsets.
* On each subset, apply the same classification Algo. & then combine the results.
* Subsets can be formed across Rows or Columns of a dataset.
  + Randomly select subset of data points/ features/ both.
* Eg : **Random Forest** : Use Decision Tree Algo. to form Random Forest.

**Code Implementation :**

[Implementation of Ensemble Method - Bagging](https://colab.research.google.com/drive/1rV6-Sapb2fUsyM-o9ABUWL5Mm_RPKvIK?usp=sharing)

Random Forest :

* It is a bagging method where the base estimator is a Decision tree.
* You can again train the model only on important features to get the better model.
* **Code Implementation :**

[Random Forest - A Bagging Ensemble Method](https://colab.research.google.com/drive/1jwOCk8YWraYxsUYmckK4dZ7amgAC-l5G?usp=sharing)

1. **Boosting :**

* Boosting is a sequential ensembling method.
* Create a sequence of models where, Kth model is an improvement of (K-1) Model.
* Therefore the final model is iteratively built.

Based on how the model is improved, you can categorize the boosting methods into :

* 1. **Adaptive Boosting / Adaboost Algorithm :** 
     + First create the initial model by giving equal importance to all the data points.
     + Then look into the misclassifications to those data points to correct them.
     + Continue the process iteratively.
  2. **Gradient Boosting:**
     + Trains on the errors to minimize over the previous model.
     + Combines Gradient Descent Algo. & Boosting Method.
  3. **Extreme Gradient Boost / XG Boost :**
     + It is designed to improve on the competitional speed and also scale using multiple processing units.
     + So, the training process is faster and scales it for bigger data.

**Code Implementation :**

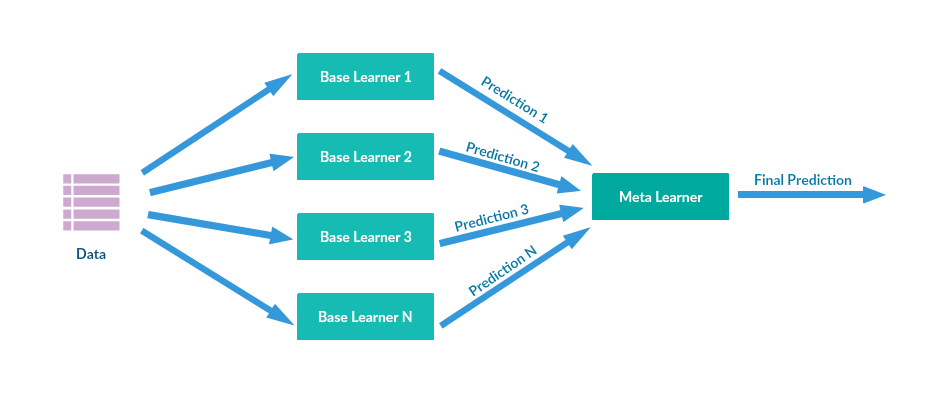
[AdaBoost & GradientBoost Implementation](https://colab.research.google.com/drive/1t7HGaJOQHkE7hdMJENFDlOWdsc1HDXAY?usp=sharing)

* **Merits & Demerits of Boosting Ensemble Method:**
* Reduce the Bias problem.
* Computational Efficiency.
* Possibility of Overfitting.
* Slower than Bagging method.

1. **Stacking :**

**Code Implementation :**

[Stacking Method Implementation](https://colab.research.google.com/drive/1t7HGaJOQHkE7hdMJENFDlOWdsc1HDXAY?usp=sharing)



**End of Chapter(4)**