**The Machine Learning process**

Step-1: Data pre-processing

* Import the data
* Clean the data
* Split into training and test sets

Step-2: Modelling

* Build the model
* Train the model
* Make predictions

Step-3: Evaluation

* Calculate performance metrics
* Make a verdict

**The data pre-processing tools**

**Step-1: Importing the main libraries**

1. **NumPy**: It is used for numerical computing and support of arrays, matrices, and mathematical functions to operate on these arrays efficiently.
2. **Pandas**: It is designed for data manipulation and analysis. It provides data structures like series and data frames (2d labelled data structure) along with tools for reading and writing data, handling missing data, reshaping, merging, and more.
3. **Scikit-learn:** It is a popular machine-learning library in Python.

**Step-2: Importing the dataset**

When importing a dataset it needs to be assigned to a variable. Pandas’ library is used for importing. Moreover, the delimiter needs to be checked before importing the file. Other

than Commas, the type of delimiter must be mentioned when importing.

**data\_frame = pd.read\_csv(‘file\_name.csv’, delimiter=” ,”)**

Types of Variables:

Generally, the data frame is divided into two types. They are:

**Independent variables or features**: These variables are used to check whether there is a relation with the dependent variable or target.

Usually, these variables are placed at starting of a table.

Dependent variable or target: This variable is placed in the last column of the table.

To divide the dataset into two variables, we can use different types of methods:

iloc[ row-start : row-end, column-start : column-end]

drop[ “column-name”, axis = 1]

dataset\_name[“column-name”]

**step-3: Taking care of the missing data**

Generally, when a dataset is imported, it may contain missing numerical values and this may affect our results. So, the best way to replace the null values in a variable is to use its mean.

Median can be used, but it can be easily affected by outliers.

* The first step is to import SimpleImputer class

From sklearn.impute import SimpleImputer

* Create an object using SimpleImputer class

imputer = SimpleImputer(missing\_values=np.nan, strategy = ‘mean’)

* Apply fit method to get mean and median values

imputer.fit(x. iloc[ row-start : row-end, column-start : column-end])

* Assign the transform method to the variable you choose. Remember the must be numerical.

x.iloc[ row-start : row-end, column-start : column-end] = imputer.transform(iloc[ rowstart : row-end, column-start : column-end]

**step-4: Encoding categorical data**

So, in machine learning, categorical data should be encoded so that in future models it can be used.

**Encoding an independent categorical variable:**

For encoding an independent categorical, one-hot encoding is preferred.

One-hot encoding is a technique used in machine learning and data processing to convert categorical data into a binary format.

This encoding is particularly useful when dealing with categorical variables in machine learning algorithms that require numerical input, such as neural networks. It helps to ensure that the model can interpret and process categorical data effectively.

From sklearn.compose import ColumTransformer

From sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[(‘encoder’, OneHotEncoder(), [0])], remainder=’passthrough’)

X = np.array(ct.fit\_transform(x))

* After importing ColumnTransformer and OneHotEncoder classes, an object is created using these classes.
* “passthrough” is used for skipping the unwanted columns to be un coded.

**Encoding the dependent variable:**

For dependent variables, LabelEncoder is used

From sklearn.preprocessing import LabelEncoder

Variable\_name = LabelEncoder()

Y = variable\_name.fit\_transform(y)

**Step 5: Splitting the dataset into the training set and test set**

The dataset is divided into training and test datasets. The ratio is 8:2. The model from the training dataset is used to evaluate the test dataset.

**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 = 1)**

**Step 6: Feature Scaling**

**Feature Scaling:** It is a data preprocessing technique used for transforming the values of features or variables in a dataset to a similar scale. It is applied across columns, not rows.

Two types of feature scaling are:

* Normalization: X-Xmin / Xmax – Xmin

Range – 0 to 1

Used when there is a normal distribution

* Standardization: X – Mean / standard deviation

Range – (-3 to 3)

Standard feature scaling is generally used

* Mostly, standardization is preferred

From sklearn.preprocessing import StandardScaler

Variable\_name = StandardScaler()

x\_train[:, :] = sc.fit\_transform(x\_train[:, :])

x\_test[:, :] = sc.transform(x\_test[:, :])