MACHINE LEARNING

What is Machine Learning?

Machine Learning is the science of getting computers to learn and act like humans do, and improve their learning over time in autonomous fashion, by feeding them data and information in the form of observations and real-world interactions.

- Making Machine Intelligent
- Machine Learning at its most basic is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world.
- Machine learning is the science of getting computers to act without being explicitly programmed.
- Machine learning is based on algorithms that can learn from data without relying on rules-based programming.
- Machine learning algorithms can figure out how to perform important tasks by generalizing from examples.

Importance of Machine Learning

Machine learning is important because of its wide range of applications and its incredible ability to adapt and provide solutions to complex problems efficiently, effectively and quickly.

Various Fields of Machine Learning

- Virtual Personal Assistant
- Facial Recognition
- Email Spam Filter
- Recommendation engine on an e-commerce website
- Online fraud detection
- Social Media Services such as “People you may know” on Facebook
- Online Customer Support i.e. Chatbot
- Search Engine Result Refining
- Predictions while commuting using Google Maps

STEP BY STEP PROCEDURE FOR MACHINE LEARNING (ML) PROGRAM

The First step to dive in Machine Learning is **Data Preprocessing.**

What is Data?

It can be any unprocessed fact, value, text, sound or picture that is not being interpreted and analyzed. Data is the most important part of all Data Analytics, Machine Learning, Artificial Intelligence.

We can get out Data from Different Field. Set of all data is Known as **Dataset**.

What is Dataset?

A **dataset** is a collection of **data**. In other words, a **dataset** corresponds to the contents of a single database table, or a single statistical **data** matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the **dataset** in question.

The Dataset Contains of Independent and Dependent variable's.

- Independent Variable: Independent variables (also referred to as Features) are the input for a process that is being analyzes.
- Dependent Variable: Dependent variables are the output of the process.

What is Data Preprocessing?

Data preprocessing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we preprocess our data before feeding it into our model.

Thing We do in Preprocessing -

- Handling Null Values
- Standardization

- Handling Categorical Variables
- One-Hot Encoding
- Multicollinearity

Steps of Data Preprocessing:

- Importing python libraries
- Importing the Dataset
- Finding the Missing data
- Categorical of Data
- Splitting Dataset into Training set and Test set
- Feature Scaling

IMPORTING PYTHON LIBRARIES

Libraries are sets of routines and functions that are written in a given language. A robust set of libraries can make it easier for developers to perform complex tasks without rewriting many lines of code.

Here is the list of Python Libraries used in data preprocessing:

- Numpy
- Pandas
- Matplotlib
- Sklearn

// Python Code

 # Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

NumPy: - It is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

Matplotlib: - Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

Pandas: - Pandas is an open-source python package built on top of Numpy developed by Wes McKinney. It is used as one of the most important data cleaning and analysis tool. It provides fast, flexible, and expressive data structures.

IMPORTING THE DATASET

A **dataset** is a collection of **data**. In other words, a **dataset** corresponds to the contents of a single database table, or a single statistical **data** matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the **dataset** in question.

The Dataset Contains of Dependent and Independent variable.

//Python Code

```
# Importing the dataset
    dataset = pd.read_csv('Data.csv')
    X=dataset.iloc[:,:-1].values
    Y=dataset.iloc[:,3].values
```

- **X= dataset.iloc[:,:-1].values** : This Line takes all the independent values from the Datset, except the dependent values.
- **: -1** :- This Means last column should be avoided
- **Y=dataset.iloc[:,3].values :-** This lines take only the last column of the dataset.

MISSING DATA

In Some Dataset, there will be some missing data in the dataset. We can't ignore/remove that data from the dataset, It could be quite dangerous because imagine that dataset contains some crucial information. So for these kind of we need figure out the solution.

To Find out the missing value, the most common method is using the Statistical Function - **Mean.**

List of the Statistical function:

- **MEAN**
- **MEDIAN**
- **MODE**

//Python Code

#Getting the Missing Values/Datas in Dependent varaibale

```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(missing_values=np.nan, strategy='mean')

imputer=imputer.fit(X[:,1:3])

X[:,1:3]=imputer.transform(X[:,1:3])
```

Sklearn :- Sklearn is also known as Scikit-learn. Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib

The above code fill the missing values in the dataset by using **Strategy= Mean.**

For Example:

47+58+68+95+89 = 357

357/5 = 71.4

Mean Value = 71.4

CATEGORICAL DATA

The machine learning model is completely based on the mathematical equation. So in the dataset, if any Column/Row contains words example: - “Sun”, “France”, “Germany” etc... It can be anything a name, country name, direction any word.

We need to encode these words. Using the python libraries encoders.

//Python Code

#Categorical Data

```
from sklearn.preprocessing import LabelEncoder , OneHotEncoder

from sklearn.compose import ColumnTransformer

labelencoder_X=LabelEncoder()

X[:,0]= labelencoder_X.fit_transform(X[:,0])

onehotencoder = ColumnTransformer([( "Country", OneHotEncoder(), [0])], remainder = 'passthrough')

X = np.array(onehotencoder.fit_transform(X), dtype = np.float64)

X = X[:, 0:]

labelencoder_Y=LabelEncoder()

Y= labelencoder_Y.fit_transform(Y)
```

LabelEncoder: - LabelEncoder encode labels with a value between 0 and n_classes-1 where n is the number of distinct labels

For Example: SUN ----- 0

France --- 1

OneHotEncoder: - OneHotEncoder is the library function is used to encode the data into Binary Form of data.

For Example: 0 ---------- 1 0 0

1 ---------- 0 1 0

2 ---------- 0 0 1

SPLITTING DATASET INTO TRAINING SET AND TEST SET

Why Training Set and Test Set: -

Well Focus on the Machine learning, it itself is about a machine that is going to learn something.

It's your algorithm model that is going to learn from your data to make prediction.

So the data's dataset is split to Training data and Test data.

For example: - Just think you have 50 data in dataset. So you program like 40 data to training set and balance 10 for test set. So the machine will learn by training that 40 data and get tested by that 10 data. We can figure out the output values of both.

//Python Code

#Splitting Dataset into Training Set and Test Set

```
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=0)
```

X_train:- X_train is the training part of the matrix of feature.

X_test: - X_test is the test part of the matrix of feature.

Y_train: - Y_train is the training part of the dependent variable vector associated to X_train.

Y_test: - Y_test is the test part of the dependent variable vector associated to X_test

FEATURE SCALING

Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization. For example, many classifiers calculate the distance between two points by the Euclidean distance.

If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance.

Euclidean Distance Between p_1 and p_2= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Standardisation	Normalisation
$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$	$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$

//Python Code

#Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc_X=StandardScaler()
X_train=sc_X.fit_transform(X_train)
X_test=sc_X.transform(X_test)
```

DATA PREPROCESSING TEMPLATE

//Full Program of Data PreProcessing

 # Data Preprocessing Template

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Data.csv')
X=dataset.iloc[:,:-1].values
Y=dataset.iloc[:,3].values

#Getting the Missing Values/Datas in Dependent varaibale
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer=imputer.fit(X[:,1:3])
X[:,1:3]=imputer.transform(X[:,1:3])

#Categorical Data
from sklearn.preprocessing import LabelEncoder ,OneHotEncoder
from sklearn.compose import ColumnTransformer
labelencoder_X=LabelEncoder()
X[:,0]= labelencoder_X.fit_transform(X[:,0])
onehotencoder = ColumnTransformer([( "Country",
OneHotEncoder(), [0])], remainder = 'passthrough')
```

```
X = np.array(onehotencoder.fit_transform(X), dtype = np.float64)
X = X[:, 0:]
labelencoder_Y=LabelEncoder()
Y= labelencoder_Y.fit_transform(Y)

#Splitting Dataset into Training Set and Test Set
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=0)

#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X=StandardScaler()
X_train=sc_X.fit_transform(X_train)
X_test=sc_X.transform(X_test)
```