Multiclass Classification-

The problem of solving instances into one of three or more classes.

o/p-> Failure Type

1)
```
import numpy as np

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Numpy-to perform wide variety of mathematical operations on arrays

Pandas-For data manipulation and analysis

Matplotlib- creating static, animated and interactive visualization in python

Seaborn- making statistical graphics in python

2)
```
from google.colab import files

uploaded = files.upload()
```

import and upload files

3)
```
df=pd.read_csv('predictive_maintenance.csv')

df.head()
```

A CSV is a comma-separated values file, which allows data to be saved in a tabular format.

df.head()-1st five rows

4)
```
df.shape
```

enables us to obtain the shape of a DataFrame.

5)
```
df.info()
```

prints information about the DataFrame.

6)
```
df.isna().sum()
```

df.isna().sum() returns the number of missing values in each column.

7)
```
df.describe()

a.only on numerical column
```

b.descriptive statistics

c.shows scale or range of each column

8) df['Failure Type'].value_counts()

a.pandas function

b.it returns object containing counts of unique values

c.NO FALIURE upto90%

d.there are high chance of wrong o/p bcoz we taught machine something wrong

e.there ois uniformly distributed output.

f.we detected class imbalance problem here

EDA-1)explore null values

2)data types of each column

3)find missing values

Now,we have to do column by column analysis

Fistly, we did column by column ananlysis of o/p column to find how many classes are present in that

Now, we will find is there any dependency in categorical and numerical column?

For categorical column, to find dependency there is one function

i.e crosstab

9) pd.crosstab(index = df['Type'],columns = df['Failure Type'],dropna = True,normalize = 'columns',margins = True)

a) joint probability (liklihood of two independent event happening at same time)

b)we are finding relation between this 2 categorical column

c) The dropna() method removes the rows that contains NULL values.

d) normalize = 'columns'=== to scale numeric data from different columns down to an equivalent scale.

e) margins = True All columns and rows will be added with partial group aggregates across the categories on the rows and columns.

10) plt.figure(figsize=(15,10))

```
sns.heatmap(df.corr(),annot = True,cmap = "RdYlBu")
plt.draw()
```

```
a) df.corr()=to find relation between 2 numerical columns
b) we put df.corr() fun in seaborn libraris heatmap function
c) annnot=true --→ it shows values in the box
d) cmap→ color
e) more dsrker then more relation between 2 column
```

11) df.hist(figsize=(16, 16),color = 'green',edgecolor = 'white',bins = 10)

WE can evaluate spread and max-min of the column

A bin is a single range of continuous values used to group values in a chart.

12) sns.set(style ="darkgrid")

```
sns.regplot(x = df["Air temperature [K]"],y = df["Process temperature [K]"],fit_reg = True,marker = "*")
```

as we studied that there is positive relation between AT and PT so we draw graph using seaborn's regplot function and even we fitted regression line using fit_reg=true

13) sns.set(style ="darkgrid")

```
sns.regplot(x = df["Rotational speed [rpm]"],y = df["Torque [Nm]"],fit_reg = True,marker = "*")
```

as there is negative relation between this two

hence graph in descending order

14) `sns.lmplot(x="Rotational speed [rpm]",y ="Torque [Nm]",data = df,fit_reg= False ,hue = 'Failure Type',legend = True,palette = "Set1" )`

`lmplot` **=You can draw 2 numerical column graph according to some categorical column**

here we have categorical column= faiure type

here No failure spread is max and even we see its range

from above we can see that if rotational speed is between 1250 to 1500 then there is high chances of power failure

this is how we are fetching some insights from given data

until this , we have explored data and after this modelling starts…..

15) `df["Type"].replace({"H":0,"L":1,"M":2}, inplace=True)`

Inplace=true ➔ data frame has to make changes permanent

16)

```
col_name=df.columns.to_list()
col_name
```

returns column name in list

17)

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=40, test_size=0.33)
```

First of all, we convert input data for training and testing. 70% data for training and 30% for testing. And for this we use Random forest classifier.

The random forest classifier is a set of decision trees from a randomly selected subset of the training set.

The sklearn. ensemble module includes two averaging algorithms based on randomized decision trees: the RandomForest algorithm and the Extra-Trees method.

`test_size=0.33`➔ this means 77% of observations from your complete data will be used to train/fit the model and rest 33% will be used to test the model.

18)

```
from imblearn.combine import SMOTETomek
```

```
smote = SMOTETomek(random_state=42)
X1_res, y1_res = smote.fit_resample(X_train,y_train)
```

Imblearn techniques are the methods by which we can generate a data set that has an equal ratio of classes.

Synthetic Minority Oversampling Technique (SMOTE) is a statistical technique for increasing the number of cases in your dataset in a balanced way.

SMOTETomek is somewhere upsampling and downsampling

19)
```
from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.preprocessing import (OneHotEncoder, PowerTransformer,
StandardScaler)
from sklearn.impute import SimpleImputer
```

column transformer- for preprocessing

make_pipeline= you can do scaling, fill missing values

i.e whatever we have done before modelling can do here in one function

The sklearn. preprocessing package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.

StandardScaler removes the mean and scales each feature/variable to unit variance

One-hot encoding can be used to transform one or more categorical features into numerical dummy features useful for training machine learning model

A power transform will make the probability distribution of a variable more Gaussian.

The imputer is an estimator used to fill the missing values in datasets.

simpleImputer is a class in the sklearn. impute module that can be used to replace missing values in a dataset, using a variety of input strategies

20)
```
from sklearn.ensemble import RandomForestClassifier

from sklearn.multiclass import OutputCodeClassifier
rfc = OutputCodeClassifier(RandomForestClassifier(), code_size=6,
random_state=40)
```

`OutputCodeClassifier=>`Output-code based strategies consist in representing each class with a binary code

`code_size=6..> needs 6 decision trees`

21) `model_rfc = pipeline.fit(X1_res, y1_res)`

```
y_pred_rfc = model_rfc.predict(X_test)
```

The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters.

21) `from sklearn.metrics import confusion_matrix`

```
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
cm = confusion_matrix(y_test, y_pred_rfc)
print(classification_report(y_test, y_pred_rfc))
```

`confusion_matrix=` It is a table that is used in classification problems to assess where errors in the model were made

22). `from sklearn.metrics import accuracy_score`

```
print("Accuracy Score: %.4f" %(accuracy_score(y_test,y_pred_rfc)))
```

Accuracy score is evaluated