Stages of Machine Learning

1.Collection of data,

2.Importing data to the Environment(Software)

3.Cleaning of data(filling missing value,removing unwanted data)

4.Data visualization/EDA

5.Preprocessing data

6.Feature Engineering(Identifying the feature (automatically) that influences the output)

7.Model creation and Evaluating the performance

8.Comparing results with other models or CrossValidating the model

9. Deployment or Importing to joblib

 # 1.Collection of data

Data can be collected from any data resources like Kaggle,GitHub,AWS,IBM,etc.,

from sklearn import datasets # to access default dataset in scikit learn

 # 2.Importing data to the Environment(Software)

a)Load CSV with Python Standard Library

df = csv.reader(datafile, delimiter=',', quoting=csv.QUOTE_NONE)

b)Load CSV File With NumPy

import numpy as np

df = np.loadtxt(datafile, delimiter=",")

c)Load CSV File With Pandas(Most prefered)

import pandas as pd

df = pd.read_csv(filename, names=names)

 # 3.Cleaning of data(filling missing value,removing unwanted data)

As our data is not in proper form we need to do this step.It may contain missing values or outliers.

To handle missing value,we can impute them with mean,median and mode.

For handling numeric missing values **mean and median** method of filling is done.For handling non-numeric missing values **mode or "most frequent"** method of filling is done.This can be done by our own code or automatically by using imputer class.

Note:df represents the dataframe on which we are working

df.shape	This gives the shape of the dataframe
df.columns	This gives the column name of df
df.dtypes	To know the data types of various columns in the dataset
df.info( )	Information about df
df.isna( ).sum()	This gives the total number of missing values in each columns
df.nunique( )	This gives the total number of unique values in each column
df.describe( )	Gives mean,min,max and other statistical information of numerical column
df.corr( )	Gives correlation between all variables

```
from sklearn.impute import SimpleImputer
```

To drop unwanted columns from the data frame

df.drop (['column names to be dropped'] , axis=1,inplace=True)

 # 4.Data visualization / EDA

We have do this step to view the plot of df in various aspect

A)Univariate plot

from matplotlib import pyplot as plt

plt.plot (kind=['bar','box','density'] , subplots=True ,layout=(3,3), figsize= (Width,height))

plt.xlabel ('Name of x axis')

plt.ylabel ('Name of y axis')

plt.title ('Name of title')

plt.show ( )

B)Multivariate plot

import seaborn as sns

sns.heatmap(df.corr( ),annot=True)

from pandas.plotting import scatter_matrix

scatter_matrix(df,figsize=(15,10))

To plot the histogram - df.hist( )

 # 5.Preprocessing the data

1)Most of the times the samples in the data are not in uniform scale.So in order to convert them in to a uniform scale we have to preprocess the data.We can select any one of the following preprocessing library to convert our data in to standard form.

from sklearn.preprocessing import Binarizer
from sklearn.preprocessing import LabelBinarizer
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardiScaler
from sklearn.preprocessing import Normalizer

To apply transform on a particular column we can prefer ColumnTransformer class

from sklearn.compose import ColumnTransformer
ct=ColumnTransformer (['name', encoding method, list of column name])

2) If we have more number of categorical column in our dataset we cannot use it in machine learning in an effective way .So we have to convert them into numerical values using some encoding techniques.

[Do fit( ) and transform( ) for the following encoding methods]

from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import LabelEncoder [For output variable]

Note:The best practice when encoding variables is to fit the encoding on the training dataset, then apply it to the train and test datasets.

 # 6.Feature Engineering(Identifying the feature (automatically) that influences the output)

Feature selection is also called variable selection or attribute selection.
It is the automatic selection of attributes in your data (such as columns in tabular data)that are most relevant to the predictive modeling problem you are working on.

1)Univariate Selection

[Statistical tests can be used to select those features that have the strongest relationship with the output variable]

from sklearn.feature_selection import SelectKBest

from sklearn.feature_selection import f_classif

test = SelectKBest(score_func=f_classif, k=4)

fit = test.fit(X, Y)

 # Here k represents the best 4 features from the input(X)

2)Recursive Feature Elimination

The Recursive Feature Elimination (or RFE) works by recursively removing attributes and building a model on those attributes that remain.

from sklearn.feature_selection import RFE

from sklearn.linear_model import LogisticRegression

model = LogisticRegression(solver='lbfgs')

rfe = RFE(model, 3) # The best 3 features from the input(X)

fit = rfe.fit(X, Y)

3.Principal Component Analysis

Principal Component Analysis (or PCA) uses linear algebra to transform the dataset into a compressed form.

from sklearn.decomposition import PCA

pca = PCA(n_components=3) # Here n_components represents the best 3 features from the input(X)

fit = pca.fit(X)

4.Feature Importance

Bagged decision trees like Random Forest and Extra Trees can be used to estimate the importance of features.

from sklearn.ensemble import ExtraTreesClassifier

model = ExtraTreesClassifier(n_estimators=10)

model.fit(X, Y)

#To Convert a collection of text documents to a matrix of token counts

from sklearn.feature_selection.text import CountVectorizer

 # 7.Model creation and Evaluating the performance

There are various models available to fit your problem(Classification /Regression).Before to select the particular model it is wise to split your data into test and train data.(Resampling)

Training Dataset: The sample of data used to fit the model.

Validation Dataset: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

Test Dataset: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

from sklearn.model_selection import train_test_split[This results in more variance. This means that differences in the training and test dataset can result in meaningful differences in the estimate of accuracy.**]**

from sklearn.model_selection import ShuffleSplit[Another variation on k-fold cross validation is to create a random split of the data like the train/test split described above, but repeat the process of splitting and evaluation of the algorithm multiple times, like cross validation.**]**

from sklearn.model_selection import KFold [Results in less variance, Each split of the data is called a fold.]

from sklearn.model_selection import StratifiedKFold

from sklearn.model_selection import TimeSeriesSplit

What Techniques to Use When

- Generally k-fold cross validation is the gold-standard for evaluating the performance of a machine learning algorithm on unseen data with k set to 3, 5, or 10.
- Using a train/test split is good for speed when using a slow algorithm and produces performance estimates with lower bias when using large datasets.
- Techniques like leave-one-out cross validation and repeated random splits can be useful intermediates when trying to balance variance in the estimated performance, model training speed and dataset size

Various Classifier and Regression Models (Frequently used)

from sklearn.linear_model import LinearRegression, LogisticRegression, HuberRegressor

from sklearn.tree import DecisionTreeClassifier, plot_tree

from sklearn.svm import SVC,SVR

from sklearn.cluster import KMeans

from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor

from sklaern.ensemble import GradinetBoostClassifier, GradientBoostRegressor

from sklearn.naive_bayes import GaussianNB, BinomialNB, MultinomialNB

from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor, Kneighbors_graph

EVALUATING PERFORMANCE

Classification metrics

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,
auc, roc_curve[For Binary Classification Problem]

Regression metrics

from sklearn.metrics import mean_absolute_error(y_true, y_pred, *),
mean_squared_error(y_true, y_pred, *),
r2_score(y_true, y_pred, *[, â€¦])

Plots

from sklearn.metrics import plot_confusion_matrix, plot_roc_curve

8.Comparing results with other models or CrossValidating the model

To get the best model by optimizing the hyper parameter

from sklearn.model_selection import GridSearchCV

from sklearn.model_selection import RandomizedSearchCV

from sklearn.model_selection import cross_val_score

from sklearn.pipeline import Pipeline #(to automate the entire process)

 # 9. Deployment or Importing to joblib

import joblib

dump the model

load the model

test the model by giving unseen data