**Summary of Machine Learning with Python**

1. **Preparing Data**

* **Why Data Pre-processing?**

After selecting the raw data for ML training, the most important task is data pre-processing. data preprocessing will convert the selected data into a form we can work with or can feed to ML algorithms. We always need to preprocess our data so that it can be as per the expectation of machine learning algorithm.

* **Data Pre-processing Techniques**

1. **Scaling**

Data rescaling makes sure that attributes are at same scale. Generally, attributes are rescaled into the range of 0 and 1. ML algorithms like gradient descent and k-Nearest Neighbors requires scaled data. We can rescale the data with the help of MinMaxScaler class of scikit-learn Python library.

1. **Normalization**

This is used to rescale each row of data to have a length of 1

**L1 Normalization**

It may be defined as the normalization technique that modifies the dataset values in a way that in each row the sum of the absolute values will always be up to 1. It is also called **Least Absolute Deviations**.

**L2 Normalization**

It may be defined as the normalization technique that modifies the dataset values in a way that in each row the sum of the squares will always be up to 1. It is also called least squares.

1. **Binarization**

The values above that threshold value will be converted to 1 and below that threshold will be converted to 0

1. **Standardization**

Another useful data preprocessing technique which is basically used to transform the data attributes with a Gaussian distribution. It differs the mean and SD (Standard Deviation) to a standard Gaussian distribution with a mean of 0 and a SD of 1. This technique is useful in ML algorithms like linear regression, logistic regression that assumes a Gaussian distribution in input dataset and produce better results with rescaled data. We can standardize the data (mean = 0 and SD =1)

1. **Data Labeling**

We discussed the importance of good fata for ML algorithms as well as some techniques to pre-process the data before sending it to ML algorithms. One more aspect in this regard is **data labeling**. It is also very important to send the data to ML algorithms having proper labeling. For example, in case of **classification problems**, lot of labels in the form of words, numbers etc. are there on the data.

What is Label Encoding?

Most of the sklearn functions expect that the data with number labels rather than word labels. Hence, we need **to convert such labels into number labels**. This process is **called label encoding**

**Example Scaling**

In this example we will rescale the data of Pima Indians Diabetes dataset which we used earlier.

from pandas import read\_csv

from numpy import set\_printoptions

from sklearn import preprocessing

path = r'D:\Machine Learning Task\datasets\pima-indians-diabetes.csv'

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

dataframe = read\_csv(path, names=names)

array = dataframe.values

data\_scaler = preprocessing.MinMaxScaler(feature\_range=(0,1))

data\_rescaled = data\_scaler.fit\_transform(array)

set\_printoptions(precision=1)

print ("\nScaled data:\n", data\_rescaled[0:10])

Output

Scaled data:

[[0.4 0.7 0.6 0.4 0. 0.5 0.2 0.5 1. ]

[0.1 0.4 0.5 0.3 0. 0.4 0.1 0.2 0. ]

[0.5 0.9 0.5 0. 0. 0.3 0.3 0.2 1. ]

[0.1 0.4 0.5 0.2 0.1 0.4 0. 0. 0. ]

[0. 0.7 0.3 0.4 0.2 0.6 0.9 0.2 1. ]

[0.3 0.6 0.6 0. 0. 0.4 0.1 0.2 0. ]

[0.2 0.4 0.4 0.3 0.1 0.5 0.1 0.1 1. ]

[0.6 0.6 0. 0. 0. 0.5 0. 0.1 0. ]

[0.1 1. 0.6 0.5 0.6 0.5 0. 0.5 1. ]

[0.5 0.6 0.8 0. 0. 0. 0.1 0.6 1. ]]

**Example L1 Normalization**

In this example we will rescale the data of Pima Indians Diabetes dataset which we used earlier.

from pandas import read\_csv

from numpy import set\_printoptions

from sklearn.preprocessing import Normalizer

path = r'D:\Machine Learning Task\datasets\pima-indians-diabetes.csv'

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

dataframe = read\_csv (path, names=names)

array = dataframe.values

Data\_normalizer = Normalizer(norm='l1').fit(array)

Data\_normalized = Data\_normalizer.transform(array)

set\_printoptions(precision=2)

print ("\nNormalized data:\n", Data\_normalized [0:3])

Output

Normalized data:

[[0.02 0.43 0.21 0.1 0. 0.1 0. 0.14 0. ]

[0. 0.36 0.28 0.12 0. 0.11 0. 0.13 0. ]

[0.03 0.59 0.21 0. 0. 0.07 0. 0.1 0. ]]

**Example L2 Normalization**

In this example we will rescale the data of Pima Indians Diabetes dataset which we used earlier.

from pandas import read\_csv

from numpy import set\_printoptions

from sklearn.preprocessing import Normalizer

path = r'D:\Machine Learning Task\datasets\pima-indians-diabetes.csv'

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

dataframe = read\_csv (path, names=names)

array = dataframe.values

Data\_normalizer = Normalizer(norm='l2').fit(array)

Data\_normalized = Data\_normalizer.transform(array)

set\_printoptions(precision=2)

print ("\nNormalized data:\n", Data\_normalized [0:3])

Output

Normalized data:

[[0.03 0.83 0.4 0.2 0. 0.19 0. 0.28 0.01]

[0.01 0.72 0.56 0.24 0. 0.22 0. 0.26 0. ]

[0.04 0.92 0.32 0. 0. 0.12 0. 0.16 0.01]]

**Example Binarization**

In this example, we will rescale the data of Pima Indians Diabetes dataset which we used earlier

from pandas import read\_csv

from sklearn.preprocessing import Binarizer

path = r'D:\Machine Learning Task\datasets\pima-indians-diabetes.csv'

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

dataframe = read\_csv(path, names=names)

array = dataframe.values

binarizer = Binarizer(threshold=0.5).fit(array)

Data\_binarized = binarizer.transform(array)

print ("\nBinary data:\n", Data\_binarized [0:5])

Output

Binary data:

[[1. 1. 1. 1. 0. 1. 1. 1. 1.]

[1. 1. 1. 1. 0. 1. 0. 1. 0.]

[1. 1. 1. 0. 0. 1. 1. 1. 1.]

[1. 1. 1. 1. 1. 1. 0. 1. 0.]

[0. 1. 1. 1. 1. 1. 1. 1. 1.]]

**Example Standarization**

from sklearn.preprocessing import StandardScaler

from pandas import read\_csv

from numpy import set\_printoptions

path = r'D:\Machine Learning Task\datasets\pima-indians-diabetes.csv'

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

dataframe = read\_csv(path, names=names)

array = dataframe.values

data\_scaler = StandardScaler().fit(array)

data\_rescaled = data\_scaler.transform(array)

set\_printoptions(precision=2)

print ("\nRescaled data:\n", data\_rescaled [0:5])

**Output**

Rescaled data:

[[ 0.64 0.85 0.15 0.91 -0.69 0.2 0.47 1.43 1.37]

[-0.84 -1.12 -0.16 0.53 -0.69 -0.68 -0.37 -0.19 -0.73]

[ 1.23 1.94 -0.26 -1.29 -0.69 -1.1 0.6 -0.11 1.37]

[-0.84 -1. -0.16 0.15 0.12 -0.49 -0.92 -1.04 -0.73]

[-1.14 0.5 -1.5 0.91 0.77 1.41 5.48 -0.02 1.37]]

**Example Data Labeling**

import numpy as np

from sklearn import preprocessing

input\_labels = ['red','black','red','green','black','yellow','white']

encoder = preprocessing.LabelEncoder()

encoder.fit(input\_labels)

test\_labels = ['green','red','black']

encoded\_values = encoder.transform(test\_labels)

print("\nLabels =", test\_labels)

print("Encoded values =", list(encoded\_values))

encoded\_values = [3,0,4,1]

decoded\_list = encoder.inverse\_transform(encoded\_values)

print("\nEncoded values =", encoded\_values)

print("\nDecoded labels =", list(decoded\_list))

**Output**

Labels = ['green', 'red', 'black']

Encoded values = [1, 2, 0]

Encoded values = [3, 0, 4, 1]

Decoded labels = ['white', 'black', 'yellow', 'green']

1. **Feature Selection**

* **Importance of Feature Selection**

The following are some of the benefits of automatic feature selection before modeling the data

1. Performing feature selection before data modeling will **reduce the overfitting**.
2. Performing feature selection before data modeling will **increases the accuracy of ML model.**
3. Performing feature selection before data modeling will **reduce the training time**

* **Feature selection techniques**

1. **Univariate Selection**

Used to select those features that have **the strongest relationship** with **the output variable**

1. **Recursive Feature Elimination**

feature selection technique **removes the attributes recursively** and builds the model with remaining attributes

1. **Principal Component Analysis (PCA)**

**PCA**, generally called **data reduction technique**, is very useful feature selection technique as it uses linear algebra to transform the dataset into a compressed form

1. **Feature Importance**

feature importance technique is used **to choose the importance features**

**Example Univariate Selection**

Pima Indians Diabetes dataset to select 4 of the attributes having best features with the help of chi-square statistical test.

from pandas import read\_csv

from numpy import set\_printoptions

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import chi2

path = r'D:\Machine Learning Task\datasets\pima-indians-diabetes.csv'

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

dataframe = read\_csv(path, names=names)

array = dataframe.values

X = array[:,0:8]

Y = array[:,8]

test = SelectKBest(score\_func=chi2, k=4)

fit = test.fit(X,Y)

set\_printoptions(precision=2)

print(fit.scores\_)

featured\_data = fit.transform(X)

print ("\nFeatured data:\n", featured\_data[0:4])

**Output:**

Featured data:

[[148. 0. 33.6 50. ]

[ 85. 0. 26.6 31. ]

[183. 0. 23.3 32. ]

[ 89. 94. 28.1 21. ]]

**Example Recursive Feature Elimination**

Select the best 3 attributes having the best features from Pima Indians Diabetes dataset

Output

Number of Features: 3

Selected Features: [ True False False False False True True False]

Feature Ranking: [1 2 4 6 5 1 1 3]

from pandas import read\_csv

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LogisticRegression

path = r'D:\Machine Learning Task\datasets\pima-indians-diabetes.csv'

names = ['preg', 'plas', 'pres', 'skin',

         'test', 'mass', 'pedi', 'age', 'class']

dataframe = read\_csv(path, names=names)

array = dataframe.values

X = array[:, 0:8]

Y = array[:, 8]

model = LogisticRegression(solver='lbfgs', max\_iter=150)

rfe = RFE(model, 3)

fit = rfe.fit(X, Y)

print("Number of Features: ",rfe.n\_features\_)

print("Selected Features: ",rfe.support\_)

print("Feature Ranking: ",rfe.ranking\_)

**Example of Principal Component Analysis (PCA)**

Select best 3 Principal components from Pima Indians Diabetes dataset.

from pandas import read\_csv

from sklearn.decomposition import PCA

path = r'D:\Machine Learning Task\datasets\pima-indians-diabetes.csv'

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

dataframe = read\_csv(path, names=names)

array = dataframe.values

X = array[:,0:8]

Y = array[:,8]

pca = PCA(n\_components=3)

fit = pca.fit(X)

print("Explained Variance:",fit.explained\_variance\_ratio\_)

print(fit.components\_)

**Output**

Explained Variance: [0.88854663 0.06159078 0.02579012]

[[-2.02176587e-03 9.78115765e-02 1.60930503e-02 6.07566861e-02

9.93110844e-01 1.40108085e-02 5.37167919e-04 -3.56474430e-03]

[-2.26488861e-02 -9.72210040e-01 -1.41909330e-01 5.78614699e-02

9.46266913e-02 -4.69729766e-02 -8.16804621e-04 -1.40168181e-01]

[-2.24649003e-02 1.43428710e-01 -9.22467192e-01 -3.07013055e-01

2.09773019e-02 -1.32444542e-01 -6.39983017e-04 -1.25454310e-01]]

**Example of Feature Importance**

In this example, we will use ExtraTreeClassifier to select features from Pima Indians Diabetes dataset.

from pandas import read\_csv

from sklearn.ensemble import ExtraTreesClassifier

path = r'D:\Machine Learning Task\datasets\pima-indians-diabetes.csv'

names = ['preg', 'plas', 'pres', 'skin',

         'test', 'mass', 'pedi', 'age', 'class']

dataframe = read\_csv(path, names=names)

array = dataframe.values

X = array[:, 0:8]

Y = array[:, 8]

model = ExtraTreesClassifier()

model.fit(X, Y)

print(model.feature\_importances\_)

Output:

[0.10015719 0.24417902 0.09168858 0.07403065 0.07324341 0.14316549

0.11164413 0.16189154]

**Machine Learning Algorithm Classification**

* **Introduction to Classification**

Classification may be defined as the process of predicting class or category from observed values or given data points.

Binary Classification : Spam or No spam

Type of Learners in classification

* Lazy Learners

such kind of learners **waits for the testing data to be appeared after storing the training data**. Classification is done only after getting the testing data. They spend **less time** on **training** but **more time on predicting**

* Eager Learners

eager learners construct classification **model without waiting for the testing data to be appeared after storing the training data**. They spend **more time** on training but **less time on predicting**.