Data Pre-Processing for Machine Learning Models using Python Libraries

Namrata Pandey, Pawan Kumar Patnaik, Sargam Gupta

Abstract: ***Data pre-processing is the process of transforming the raw data into useful dataset. Data pre-processing is one of the most important phase of any machine learning model because the quality and efficiency of any machine learning model directly depends upon the data-set, if we skip this step and design a model with data sets containing missing values then the model we have designed will not be that efficient and will be inconsistent model. This paper describes the methodology for pre-processing the data in seven sequence of steps using python powerful libraries which are open source machine learning libraries that support both supervised and unsupervised learning like pandas is a high level data manipulation tool, scikit learn which provides various tools for model fitting, data pre-processing, model selection and many other utilities. These steps include dealing with missing value, categorical values, importing data sets etc. This analysis helps in cleaning and transforming the datasets which future applied to any learning model and produce a efficient machine learning model.***

Keywords : Pre-processing, python libraries, machine learning..

I. INTRODUCTION

Machine Learning can be defined as a subset or application area of artificial intelligence (AI) which is acting as a hottest era in the world of automation by providing system the ability to learn by its own through its experiences without human interventions. Machine Leaning is not only limited to a single area of application but it has been an emerging technology with a wide range of application area like speech recognition, sentimental analysis, Recommendation system, Intrusion detection System and many more.

The heart of this learning process is "Data Preprocessing" as it acts as an integral part of learning (machine learning). Data pre-processing is the most important phase of any machine learning model because the ability of any model to learn efficiently and perform effectively highly and directly depends upon the quality of data [1].

Revised Manuscript Received on April 27, 2020.

***** Correspondence Author

Namrata Pandey*, Department of Computer Science and Engineering, Bhilai Institute of Technology Durg, Bhilai, Chhattisgarh, India. Email: namratapandey.26@gmail.com

Dr. Pawan Kumar Patnaik, Department of Computer Science and Engineering, Bhilai Institute of Technology Durg, Bhilai, Chhattisgarh, India. Email: pawanpatnaik37@gmail.com

Mr. Sargam Gupta, Department of Computer Science and Engineering, Bhilai Institute of Technology Durg, Bhilai, Chhattisgarh, India. Email: sargamgupta1610@gmail.com

Data preprocessing can be handled by using python powerful libraries like pandas for importing and exporting the data sets, numpy for mathematical calculations, sklearn for missing values and handling categorical variable also known as scikit learn which is one of the most powerful library of python used for data pre-processing in machine learning.

The need of data pre-processing is firstly it helps in better understanding of the data and secondly helps in handling the missing values which is highly required as if not handled can result in inefficient or we can say dump model.

II. DATA PRE-PROCESSING

Data-preprocessing can be defined as the process of "cleaning and transformation" of the data, data which can be in any format may be structured, unstructured or semi-structured and has been extracted or originated from different sources like historical data, stream-data, application-data etc.[2].

In this paper we will be discussing about Data-pre-processing for Machine Learning using Python.

The preprocessing step is applied over the KDD cup datasets using only seven features out of 41 features [3].

Seven sequence of steps need to be carried out for Data-pre-processing which are given in Table I [4].

Table-I: Data pre-processing steps

Sl.No	Steps
1	Importing Relevant libraries
2	Loading the Data-sets
3	Separating the features(variables) as independent and dependent
4	Handling the Null-values
5	Handling Categorical Variables
6	Splitting the data-sets into training and testing sets
7	Applying feature Engineering

A. Step-1 Importing Relevant Libraries

The very first step for any data pre-processing using python is importing all the relevant libraries according to your problem statements. The most commonly used or we can say basic libraries are pandas for importing and exporting the datasets, numpy for mathematical calculations and sklearn which is one of the

most powerful library used for data pre-processing [5].

We use keyword import for importing the libraries and short name for calling like importing pandas we write import pandas as pd here import is keyword used for importing any libraries and pandas is library for importing datasets and pd is short name used for calling.

Data Pre-processing For Machine Learning Using Python Labraries

1. Importing libraries

```
In [1]: # importing relevant libraries
        %matplotlib inline
        import matplotlib
        import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        import sklearn
        import imblearn
        import seaborn as sns
        import sys
```

Now for rectifying the version of our libraries we make use of following sets of code as illustrated below.

```
In [2]: print("pandas : {0}".format(pd.__version__))
        print("numpy : {0}".format(np.__version__))
        print("matplotlib : {0}".format(matplotlib.__version__))
        print("seaborn : {0}".format(sns.__version__))
        print("sklearn : {0}".format(sklearn.__version__))
        print("imblearn : {0}".format(imblearn.__version__))

        pandas : 0.25.1
        numpy : 1.16.5
        matplotlib : 3.1.1
        seaborn : 0.9.0
        sklearn : 0.21.3
        imblearn : 0.5.0
```

B. Step-2 Loading the Datasets

The next step after importing the libraries is loading the datasets. Using python we can read different data format files like image files, coma separated values (csv files), Excel files, Text files, HTML, Hierarchical Data format, audio files, video files and many more.

Below code illustrates the reading of Excel datasets, text datasets and csv datasets which are mainly used file format for machine learning [6].

Syntax for loading the datasets using python Variable_name = pd.read_file format ("file location/path of the file"). where, Variable name is the name of variable in which data will get stored. Pd means using pandas libraries for reading the data sets and file format is the format of your data in which format you have saved your datasets.

File location is the location of your file where you have located or saved your file. Next, we make use of head () function to display few rows of data or we can simply use the print () function for displaying the loaded datasets.

2. Loading the Datasets

```
In [3]: # Loading the data-sets using the pandas libraries
        ## Reading Excel Datasets
        Xlsxdata = pd.read_excel("C:/Users/hp/Desktop/pre-processing/Train.xlsx")
        ## Reading Csv Datasets
        Csvdata = pd.read_csv("C:/Users/hp/Desktop/pre-processing/train.csv")
        ## Reading Text Datasets
        Textdata = pd.read_table("C:/Users/hp/Desktop/pre-processing/KDDTrain.txt")
```

```
Xlsxdata.head(5)
```

	duration	protocol_type	service	flag	src_bytes	land	attack
0	0	tcp	private	0.0	0.0	0.0	neptune
1	2	tcp	ftp_data	12983.0	0.0	0.0	normal
2	0	icmp	eco_i	NaN	0.0	0.0	saint
3	1	tcp	telnet	NaN	15.0	0.0	mscan
4	0	tcp	http	267.0	14515.0	0.0	normal

Next, we make use of describe() function to describe our data

```
# Descriptive Statistics
Xlsxdata.describe()
```

	duration	flag	src_bytes	land
count	22543.000000	2.254100e+04	2.254200e+04	22542.000000
mean	218.868784	1.039683e+04	2.056194e+03	0.008429
std	1407.207069	4.728179e+05	2.122023e+04	0.142605
min	0.000000	0.000000e+00	0.000000e+00	0.000000
25%	0.000000	0.000000e+00	0.000000e+00	0.000000
50%	0.000000	5.400000e+01	4.600000e+01	0.000000
75%	0.000000	2.870000e+02	6.010000e+02	0.000000
max	57715.000000	6.282565e+07	1.345927e+06	3.000000

C. Step-3 separating the features (variables) as independent and dependent

While designing any machine learning model we need to split the variable as independent variable and dependent variable. Independent variables are that quantity that is being manipulated by the researcher and dependent variables represent a quantity whose value depends on those manipulations.

Independent variables are often designated by X and dependent variables as Y. Mathematically denoted as Y = f(X) (Y is the function of X) means Y depends on or determined by X. Below code illustrate splitting of variables as dependent and independent variables , where X and Y denotes independent and dependent variable, Xlsxdata is variable name in which we have stored our dataset, iloc[row limit: column limit].values is the function used. Row limit tell about the row to be included and the column limit tell about the column to be included if [:] only this is mentioned then it means entire row to be included and if [:-1] means including all the columns excluding the last one.

Next to see features included as dependent and independent variables we can simply use X and Y and then press shift plus enter.

```
# Separating Dependent and Independent variables
X=Xlsxdata.iloc[:,:-1].values
Y=Xlsxdata.iloc[:,-1].values
```

```
X
```

```
array([[0, 'tcp', 'private', 0.0, 0.0, 0.0],
       [2, 'tcp', 'ftp_data', 12983.0, 0.0, 0.0],
       [0, 'icmp', 'eco_i', nan, 0.0, 0.0],
       ...,
       [0, 'tcp', 'http', 54540.0, 8314.0, 0.0],
       [0, 'udp', 'domain_u', 42.0, 42.0, 0.0],
       [0, 'tcp', 'sunrpc', 0.0, 0.0, nan]], dtype=object)
```

```
Y
```

```
array(['neptune', 'normal', 'saint', ..., 'back', 'normal', 'mscan'],
      dtype=object)
```

D. Step-4 Handling the Null-values

The next and most crucial step in data pre-processing is to handle the missing values or null values in python we say NaN values. Simply we can handle the null values by: Dropping the entire row containing the null values, which sometimes tends to result unpredictable result and is not the best way to handle missing values. By using python libraries and implementing strategies like mean, median or mode, depends upon the problem statements. Most common strategy used "mean"

is illustrated below using sklearn python libraries.

3. Handling Missing Values

```
# Replacing missing value with average value (mean)
# Using scikit-Learn Libraires
# imputer is an istance/object of this class use to call this class
#takes only numeric value
```

```
Train.head(5)
```

	duration	protocol_type	service	flag	src_bytes	Land	attack
0	0	tcp	private	0.0	0.0	0.0	neptune
1	2	tcp	ftp_data	12983.0	0.0	0.0	normal
2	0	icmp	eco_i	NaN	0.0	0.0	saint
3	1	tcp	telnet	NaN	15.0	0.0	mscan
4	0	tcp	http	267.0	14515.0	0.0	normal

As shown above the dataset contain missing values as NaN, which is replaced by the mean strategy.

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy="mean")
imputer.fit(X[:200, 3:6])
X[:200 , 3:6]=imputer.transform(X[:200, 3:6])
```

Here we can see that the NaN value is replaced by mean value like flag value of row 2 which is null denoted as NaN is replaced by 6062.090909090909(which is flag mean value).

```
In [42]: print(X)
[[0 'tcp' 'private' 0.0 0.0 0.0]
 [2 'tcp' 'ftp_data' 12983.0 0.0 0.0]
 [0 'icmp' 'eco_i' 6062.090909090909 0.0 0.0]
 ...
 [0 'tcp' 'http' 54540.0 8314.0 0.0]
 [0 'udp' 'domain_u' 42.0 42.0 0.0]
 [0 'tcp' 'sunrpc' 0.0 0.0 0.00842871085085618]]
```

E. Step-5 Handling Categorical Variables

In this phase we handle the categorical variables using python module LabelEncoder from python library sklearn pre-processing. As dealing with missing values requires data to be stored in numeric form so its required to handle the categorical values by converting the string into numeric.

Handling Categorical Variable

```
# handling categorical variable
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
X[:, 1] = labelencoder_X.fit_transform(X[:,1])
```

```
X
```

```
array([[0, 1, 45, 0.0, 0.0, 0.0],
       [2, 1, 19, 12983.0, 0.0, 0.0],
       [0, 0, 13, 10396.832882303359, 0.0, 0.0],
       ...,
       [0, 1, 22, 54540.0, 8314.0, 0.0],
       [0, 2, 11, 42.0, 42.0, 0.0],
       [0, 1, 52, 0.0, 0.0, 0.00842871085085618]], dtype=object)
```

```
# creating dummy matrix for categorical dataset
from sklearn.preprocessing import OneHotEncoder
onehotencoder = OneHotEncoder(categorical_features = [0])
X = onehotencoder.fit_transform(X).toarray()
np.set_printoptions(suppress=True)
X
```

```
array([[    1.        ,     0.        ,     0.        , ...,
            0.        ,     0.        ,     0.        ],
       [    0.        ,     0.        ,     1.        , ...,
        12983.        ,     0.        ,     0.        ],
       [    1.        ,     0.        ,     0.        , ...,
        10396.8328823 ,     0.        ,     0.        ],
       ...,
       [    1.        ,     0.        ,     0.        , ...,
        54540.        ,  8314.        ,     0.        ],
       [    1.        ,     0.        ,     0.        , ...,
           42.        ,    42.        ,     0.        ],
       [    1.        ,     0.        ,     0.        , ...,
            0.        ,     0.        ,     0.00842871]])
```

```
# handle Y matrix for categorical data
labelencoder_Y = LabelEncoder()
Y = labelencoder_Y.fit_transform(Y)
Y
```

```
array([14, 16, 24, ...,  1, 16, 11])
```

For assigning appropriate value we make use of dummy matrix (there is no priority-based value is assigned all values are treated equally and the value assigned depends upon the occurrence of that particular variable).

F. Step-6 Splitting the data-sets into training and testing sets

Before dealing with the step first let us understand that what is train data and testing datasets.

Training Data: Testing data sets are data which are used to train the model.

Testing Data: Testing data sets are data which are used for testing or we can say that used for validating our model.

Slitting the data sets into training and testing sets means splitting the entire dataset into two ratio one for training and second for testing as shown in below Figure 1[2].

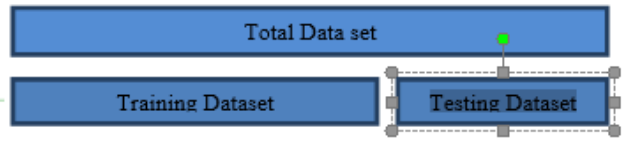

Figure 1. splitting dataset into training and testing sets

Here we need to pay attention while splitting the sets is that the ratio of training sets should be equal or greater than the test sets.

Splitting the data-set into training and testing dataset

```
# prepare test and training data 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)
```

```
X_train
```

```
array([[  1.,   0.,   0., ...,   0.,  44.,   0.],
       [  1.,   0.,   0., ..., 210., 1909.,   0.],
       [  1.,   0.,   0., ..., 157., 1964.,   0.],
       ...,
       [  1.,   0.,   0., ...,   0.,  44.,   0.],
       [  1.,   0.,   0., ...,   0.,   0.,   0.],
       [  1.,   0.,   0., ...,  30.,  217.,   0.]])
```

```
X_test
```

```
array([[  1.,   0.,   0., ...,  20.,   0.,   0.],
       [  1.,   0.,   0., ...,   0.,   0.,   0.],
       [  1.,   0.,   0., ...,   0.,   0.,   0.],
       ...,
       [  1.,   0.,   0., ...,   0.,   0.,   0.],
       [  1.,   0.,   0., ...,   0.,   0.,   0.],
       [  1.,   0.,   0., ..., 315., 347.,   0.]])
```

G. Step-7 Applying feature Scaling

The last step is applying the feature scaling which is an optional step but highly recommended as after applying this step all the variables values are contains a consistent value which is very helpful in piping the model as well as training the model through any algorithm.

Feature scaling

```
# Feature scaling used for scaling the dataset
# Normalization and standard deviation
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit_transform(X_train[0:22544,0:6])
scaler.transform(X_test[0:22544,0:6])
```

```
array([[ 0.43152433, -0.1641179 , -0.06758503, -0.07277173, -0.15758737,
        -0.08320761],
       [ 0.43152433, -0.1641179 , -0.06758503, -0.07277173, -0.15758737,
        -0.08320761],
       [ 0.43152433, -0.1641179 , -0.06758503, -0.07277173, -0.15758737,
        -0.08320761],
       ...,
       [ 0.43152433, -0.1641179 , -0.06758503, -0.07277173, -0.15758737,
        -0.08320761],
       [ 0.43152433, -0.1641179 , -0.06758503, -0.07277173, -0.15758737,
        -0.08320761],
       [ 0.43152433, -0.1641179 , -0.06758503, -0.07277173, -0.15758737,
        -0.08320761]])
```

III. RESULT AND DISCUSSION

The data set used for designing any machine learning model are extracted from different source like historical data extracted from data warehouse, streaming data sets etc and are of different types like structured data, unstructured data and semi-structured data. Moreover, the file formats are also different like text, csv, html, video file and many more formats.

These datasets which are obtained from different source of different type having different file formats contain missing values, categorical value and need to be transformed before applied for any model. The process of transformation and cleaning the data sets is known as data pre-processing phase which is one of the most crucial steps of any machine learning model.

Data pre-processing helps in dealing with null value also known as NaN in python, splitting the data sets as training and testing, separating the variables like independent and dependent and many more utilities.Python powerful libraries are easily used to handle the raw data and helps in the process of the data pre-processing. After the implementation of above described procedure the obtained results are tabulated which is given in Table II.

.

Table-II Illustrates the difference in data sets before and after applying data preprocessing.

S.NO	Data before Pre-processing	Data after pre-processing	Remark
1.	Data sets containing missing or Null values, in python missing values are represented using NaN.	Null values are replaced by mean, median or mode strategy or we can also drop the entire row containing missing value which is not a good practice all the time.	Null values are removed. Thus, datasets with null or missing value result into ambiguous model.
2.	Data sets containing Categorical value.	Categorical variables are handled using python library LabelEncoder.	Helps in processing categorical values.
3.	No Feature Scaling resulting to different approximation values.	Features are scaled so that data sets can be used for implementing any model, for feature scaling using python library sklearn pre-processing StandardScalar.	After applying feature scaling all the variable values are having same approximation after decimal.
4.	No Variable separation.	Variables are separated as independent and dependent variable.	Variables are separated as independent and dependent which is required for carrying out any mathematical or research work.
5.	No Splitting of datasets.	Splitting the datasets as training and testing sets using python library sklearn model_selection train_test_split.	Data set are splitted as different sets of data are required for training and testing the model. The ration of training data sets should be grater or equal to testing data sets.

IV. CONCLUSION

In this paper we have carried out seven sequence steps of data preprocessing. The processed data after complete analysis shows that the data is noise free, null value or missing values are replaced by any one of strategy like mean, median and mode. We have also handled the categorical variables, separation of independent variable and dependent variable. Moreover, dataset is splited as training and testing datasets.

Thus, it can be observed that after applying the defined seven sequence of steps obtained dataset can easily be applied to any model or any learning algorithm like SVM, logistic regression etc. Moreover, it was observed that the model obtained from dataset after applying data preprocessing steps was more efficient and less prone to ambiguity.

REFERENCES

1. Agarwal V. Research on Data Preprocessing and Categorization Technique for Smartphone Review Analysis. International Journal of Computer Applications. 2015 Dec;975:8887.
2. Iliou T, Anagnostopoulos CN, Nerantzaki M, Anastassopoulos G. A novel machine learning data preprocessing method for enhancing classification algorithms performance. InProceedings of the 16th International Conference on Engineering Applications of Neural Networks (INNS) 2015 Sep 25, pp. 1-5.
3. Shyu ML, Chen SC, Sarinnapakorn K, Chang L. A novel anomaly detection scheme based on principal component classifier. MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL AND COMPUTER ENGINEERING; 2003 Jan.
4. Smolinska A, Hauschild AC, Fijten RR, Dallinga JW, Baumbach J, Van Schooten FJ. Current breathomics—a review on data pre-processing techniques and machine learning in metabolomics breath analysis. Journal of breath research. 2014 Apr 8;8(2):027105.
5. Hornik K, Buchta C, Zeileis A. Open-source machine learning: R meets Weka. Computational Statistics. 2009 May 1;24(2):225-32.
6. Munková D, Munk M, Vozár M. Data pre-processing evaluation for text mining: transaction/sequence model. Procedia Computer Science. 2013 Jan 1;18:1198-207.

AUTHORS PROFILE

Namrata Pandey is a final year postgraduate student pursuing M.Tech in Data Science in Bhilai Institute of Technology, Chhattisgarh, India. Her research interests are Machine Learning and Big Data for Cyber and Information Security and Cyber Crime Analysis.

Dr. Pawan Kumar Patnaik is a Associate Professor in Department of Computer Science and Engineering at Bhilai Institute of Technology, Durg. He obtained his Ph.D. and M. Tech degree from Chhattisgarh Swami Vivekananda Technical University, Bhilai. His broad research interests are in the areas of Theoretical Computer Science and algorithm. He has close to 14 years of experience in the Theory of Computation, algorithm and Compiler Design. He is having 35 research publications in several International/National journals and conferences. He visited Czech Republic for the academic research interaction.

Sargam Gupta is Assistant Professor in Department of computer Science and Engineering at Bhilai Institute of Technology. He obtained his M.Tech.(Computer Networks), B.E. (Computer Science & Engineering) from Chhattisgarh swami Vivekananda Technical University, Bhilai. His broad research interests are in the area of Computer Networks and Artificial intelligence.