GROUP C
PDS Assignment No.- 15

Page :
Date :

Assignment No. 15

AIM →

using the sample dataset

(i) Handle the null values if any by removing them or perform imputation.

(ii) Import the necessary package and perform the train and test split on the dataset.

Theory →

Imputation →

— Imputation is a technique used for replacing the missing data with some substitute value to retain most of the data/ information of the dataset.

— These techniques are used because removing the data from the datasets every time is not feasible and can lead to a reduction in size of the dataset to a large extend, which not only raises concern for biasing the dataset but also leads to incorrect analysis.

Libraries used →

Numpy → Numpy is a python library used for working with arrays.

Matplotlib → Matplotlib is a plotting library for the python programming language, its numerical mathematical extention Numpy.

Pandas → pandas is a python package providing fast, flexible data structures designed to make working with relational data.

Camlin

rudraksh.karpe.cs@ghrcem.raisoni.net

PART-1

Program code:

```
import pandas as pd

import numpy as np

df=pd.DataFrame(np.random.randn(5,3),index=['a', 'c', 'e', 'f', 'h'],

columns=['One','Two','Three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])


print(df)

# To check missing value is available or not print (df.isnull())

# replace missing values using different method

#1.Replace NaN with a Scalar Value

print ("NaN replaced with '0':")

print(df.fillna(0))


#2.Replace NaN with Fill NA Backward -bfill/backfill

print(df.fillna(method='bfill'))


#3.Replace NaN with Fill NA Forward -pad/fill
```

```
print(df.fillna(method='pad'))
```

#3.Replace NaN if index having all NaN with drop

```
c=df.dropna()
print(c)
```

Output:

```
SCOB86_Rudraskh Karpe

        One       Two     Three
a -2.282657 -1.186591 -0.688045
b       NaN       NaN       NaN
c  0.663604  0.399326  0.633861
d       NaN       NaN       NaN
e  0.929333  0.768018 -1.058835
f  0.612848  1.066516  0.199086
g       NaN       NaN       NaN
h -1.883670  0.273025  0.236867
NaN replaced with '0':
        One       Two     Three
a -2.282657 -1.186591 -0.688045
b  0.000000  0.000000  0.000000
c  0.663604  0.399326  0.633861
d  0.000000  0.000000  0.000000
e  0.929333  0.768018 -1.058835
f  0.612848  1.066516  0.199086
g  0.000000  0.000000  0.000000
h -1.883670  0.273025  0.236867
        One       Two     Three
a -2.282657 -1.186591 -0.688045
b  0.663604  0.399326  0.633861
c  0.663604  0.399326  0.633861
d  0.929333  0.768018 -1.058835
e  0.929333  0.768018 -1.058835
f  0.612848  1.066516  0.199086
g -1.883670  0.273025  0.236867
h -1.883670  0.273025  0.236867
        One       Two     Three
a -2.282657 -1.186591 -0.688045
b -2.282657 -1.186591 -0.688045
c  0.663604  0.399326  0.633861
d  0.663604  0.399326  0.633861
e  0.929333  0.768018 -1.058835
f  0.612848  1.066516  0.199086
g  0.612848  1.066516  0.199086
h -1.883670  0.273025  0.236867
        One       Two     Three
a -2.282657 -1.186591 -0.688045
c  0.663604  0.399326  0.633861
e  0.929333  0.768018 -1.058835
f  0.612848  1.066516  0.199086
h -1.883670  0.273025  0.236867
```

PART-2

Program code:

```
#Import the necessary package and perform the train and test split on the dataset.

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

print("SCOB86_Rudraskh Karpe\n")

dataset = pd.read_csv("C:\\Users\\rudra\\OneDrive\\Desktop\\NIFTY 500.csv")


X = dataset.iloc[:, [2, 3]].values


y = dataset.iloc[:, 4].values


# Splitting Data into Training & Testing



from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.25, random_state = 0)


print(X_train)

print(X_test)

print( y_train)

print(y_test)
```

Output:

```
SCOB86_Rudraskh Karpe

[[4549.9  4504.95]
 [9898.55 9845.55]
 [1781.25 1759.4 ]
 ...
 [2570.35 2536.85]
 [4320.9  4288.45]
 [4790.7  4622.  ]]
[[ 622.25  616.2 ]
 [9458.45 9326.1 ]
 [9392.75 9282.1 ]
 ...
 [ 704.7   698.35]
 [6644.8  6590.8 ]
 [9367.05 9297.75]]
[4538.75 9892.3  1775.   ... 2559.65 4314.8  4639.1 ]
[ 620.6  9344.15 9382.05 ...  698.4  6595.85 9360.45]
```