

MACHINE LEARNING

LAB DIGITAL ASSIGNMENT 1

PARTH MAHADIK

19BCE0443

1. Demonstrate possible missing value analysis approaches using any real-world data.

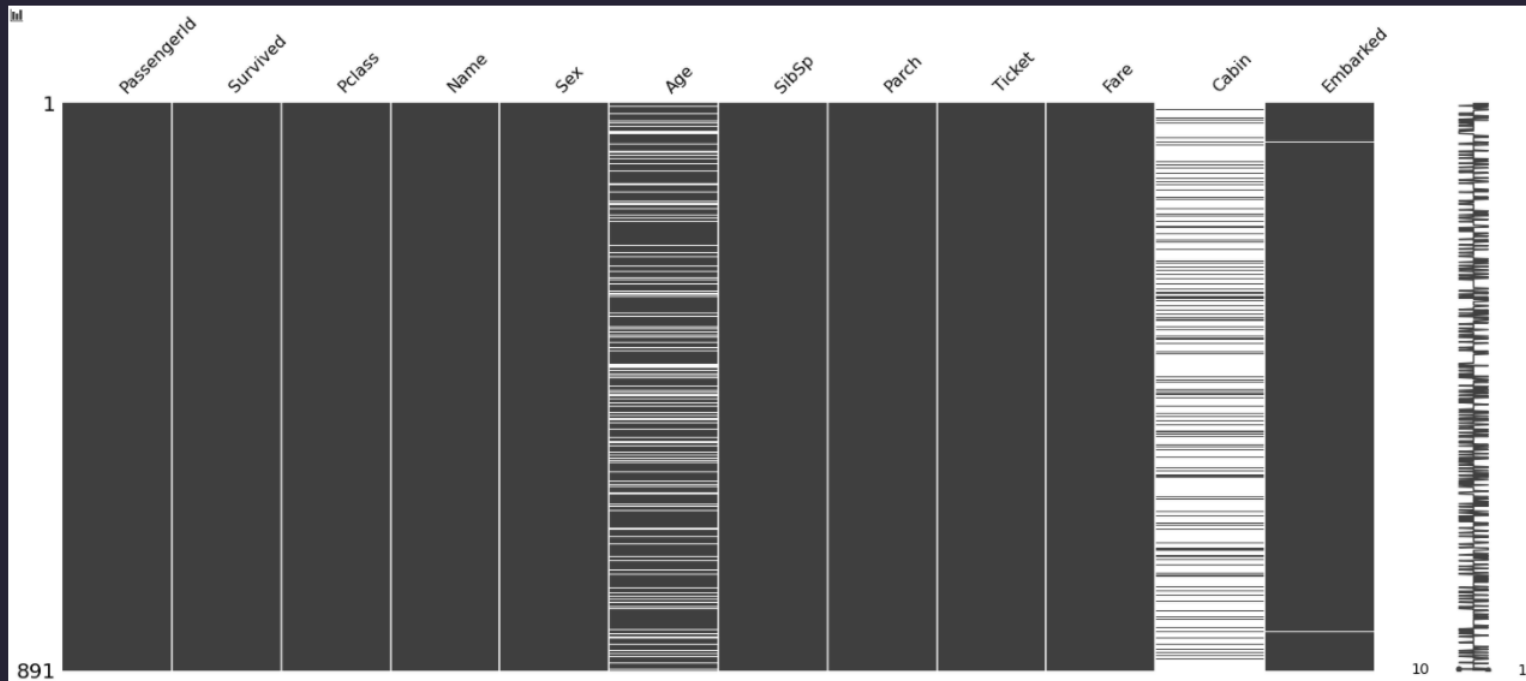
[6]

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import missingno as msno
```

[8]

```
data = pd.read_csv("E:\\VIT\\Semester 4\\Machine Learning\\LAB\\titanic\\train.csv")
msno.matrix(data)
```

<AxesSubplot:>



10

11

Removing Missing values (`data.dropna(inplace=True)`)

```
[9] In [9]: ML
print(data.isnull().sum())
print(data.shape)

PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age           177
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin         687
Embarked       2
dtype: int64
(891, 12)

[10] In [10]: ML
data.dropna(inplace=True)

[11] In [11]: ML
print(data.isnull().sum())
print(data.shape)

PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
(183, 12)
```

REPLACING WITH MEAN (`data["Age"] = data["Age"].replace(np.NaN, data["Age"].mean())`)

```
[17] In [17]: ML
data["Age"][0:20]

0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
5     NaN
6    54.0
7     2.0
8    27.0
9    14.0
10    4.0
11    58.0
12    20.0
13    39.0
14    14.0
15    55.0
16     2.0
17    NaN
18    31.0
19    NaN
Name: Age, dtype: float64

[18] In [18]: ML
data["Age"] = data["Age"].replace(np.NaN, data["Age"].mean())
data["Age"][0:20]

0    22.000000
1    38.000000
2    26.000000
3    35.000000
4    35.000000
5    29.699118
6    54.000000
7     2.000000
8    27.000000
9    14.000000
10    4.000000
11    58.000000
12    20.000000
13    39.000000
14    14.000000
15    55.000000
16     2.000000
17    29.699118
18    31.000000
19    29.699118
Name: Age, dtype: float64
```

REPLACING WITH MEDIAN (`data["Age"] = data["Age"].replace(np.NaN, data["Age"].median())`)

```
[20] data["Age"][0:20]

0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
5     NaN
6    54.0
7     2.0
8    27.0
9    14.0
10     4.0
11    58.0
12    20.0
13    39.0
14    14.0
15    55.0
16     2.0
17     NaN
18    31.0
19     NaN
Name: Age, dtype: float64
```

```
[21] data["Age"] = data["Age"].replace(np.NaN, data["Age"].median())
data["Age"][0:20]

0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
5    28.0
6    54.0
7     2.0
8    27.0
9    14.0
10     4.0
11    58.0
12    20.0
13    39.0
14    14.0
15    55.0
16     2.0
17    28.0
18    31.0
19    28.0
Name: Age, dtype: float64
```

REPLACING WITH MOST OCCURRING VALUE (`data["Cabin"]` `data["Cabin"].fillna('U')`)

```
[22] data.isnull().sum()

PassengerId      0
Survived          0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin           687
Embarked         2
dtype: int64

[24] data["Cabin"] = data["Cabin"].fillna('U')

[25] data.isnull().sum()

PassengerId      0
Survived          0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin            0
Embarked         2
dtype: int64
```

2. Suppose you have height and weight data for a group of people. For example: Heights are in feet, like 6.5, and weight is in grams, like 80000. In many machine learning situations, you want to normalize the data — scale the data so that the values in different columns have roughly the same magnitude so that large values (like the weight) don't overwhelm smaller values (like the heights). Create a raw data of minimum 40 records of height and weight in above mentioned format and use MinMax Normalization to normalize the weights in the range from (-10.0 to 10) as well as use Z-score to normalize the weights.

Min-Max Scalar

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
data = pd.read_csv("E:\\VIT\\Semester 4\\Machine Learning\\LAB\\book1.csv")
data.head()
arr = data.values
X = arr[:,0:1]
Y = arr[:,1:2]
scaler = MinMaxScaler(feature_range=(-10, 10))
rescaledY = scaler.fit_transform(Y)
```

```
[1] In ML
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

[2] In ML
data = pd.read_csv("E:\\VIT\\Semester 4\\Machine Learning\\LAB\\book1.csv")

[3] In ML
data.head()

   Height  Weight
0     5.8   80000
1     6.3   65000
2     5.7   65900
3     5.2   78000
4     5.5   78200

[4] In ML
arr = data.values

[5] In ML
X = arr[:,0:1]
Y = arr[:,1:2]

[6] In ML
scaler = MinMaxScaler(feature_range=(-10, 10))
rescaledY = scaler.fit_transform(Y)
```

```
[7] In ML
np.set_printoptions(precision=3)

[9] In ML
rescaledY[0:10]

array([[ -3.333],
       [-10.   ],
       [-9.6   ],
       [-4.222],
       [-4.133],
       [-5.467],
       [-5.067],
       [-3.778],
       [-2.884],
       [-4.133]])
```

Z-Score

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
data = pd.read_csv("E:\\VIT\\Semester 4\\Machine Learning\\LAB\\book1.csv")
data.head()
arr = data.values
X = arr[:,0:1]
Y = arr[:,1:2]
scaler = StandardScaler().fit(Y)
rescaledY = scaler.transform(Y)
np.set_printoptions(precision=3)
rescaledY[0:10]
```

```
[1] > In [1]: import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler

[2] > In [2]: data = pd.read_csv("E:\\VIT\\Semester 4\\Machine Learning\\LAB\\book1.csv")

[3] > In [3]: data.head()

   Height  Weight
0     5.8   80000
1     6.3   65000
2     5.7   65900
3     5.2   78000
4     5.5   78200

[5] > In [5]: arr = data.values

[6] > In [6]: X = arr[:,0:1]
Y = arr[:,1:2]

[7] > In [7]: scaler = StandardScaler().fit(Y)
rescaledY = scaler.transform(Y)

[8] > In [8]: np.set_printoptions(precision=3)
```

```
[8] > In [8]: np.set_printoptions(precision=3)

[9] > In [9]: rescaledY[0:10]

array([[ -0.375],
       [-1.661],
       [-1.584],
       [-0.547],
       [-0.53 ],
       [-0.787],
       [-0.71 ],
       [-0.461],
       [-0.289],
       [-0.53 ]])
```

