CSE 4020 Machine Learning
Lab Assessment - 1

Sujay Kumar M 20BDS0294
Computer Science Engineering with Specialization with DataScience
sujaykumarreddy.m2020@vitstudent.ac.in
https://github.com/sujaykumarmag/CSE4020

March 1, 2023

1 Making a DataSet

I made a Student Dataset with 10 Columns and 22 rows

	Roll No	Name	Age	DOB	CGPA	Courses	Graduation Year	Placements	M.Tech/MS	Startup
0	19BCI0876	Akhil	19	23-12-2003	8.45	8.0	2023	Yes	Yes	No
1	20BCE0076	Ram	20	3-10-2002	6.75	7.0	2024	Yes	No	Yes
2	20BDS0957	Rishab	21	2-12-2001	7.16	6.0	2024	Yes	No	No
3	20BDS0294	Sujay	20	02-12-2002	8.02	9.0	2024	No	No	Yes
4	20BCI0805	Atul	19	12-07-2003	9.14	12.0	2024	Yes	Yes	No
5	20BKT0012	Nivas	20	3-1-2002	9.54	6.0	2024	No	No	Yes
6	20BCT0121	Harshil	19	2-06-2003	8.90	5.0	2024	Yes	No	No
7	20BCI0234	Robert	20	23-1-2002	5.56	9.0	2024	No	Yes	Yes
8	20BCE0294	Richard	21	13-09-2001	6.98	8.0	2024	Yes	Yes	Yes
9	20BCE2265	Nicolas	21	17-08-2001	7.23	13.0	2024	No	No	No
10	20BCE2095	Bernard	22	27-10-2000	7.56	6.0	2024	Yes	Yes	No
11	20BCE1067	Steve	20	19-11-2002	6.90	8.0	2024	No	Yes	No
12	20BDS0398	Sanjana	20	12-05-2002	9.30	10.0	2024	Yes	No	Yes
13	20BCT0081	Misha	19	20-09-2003	8.30	12.0	2024	No	No	Yes
14	20BCI0405	Maya	20	19-10-2002	8.75	7.0	2024	No	No	NO
15	20BCI0417	Priya	19	23-07-2003	6.90	7.0	2024	NaN	NaN	NaN
16	19BDS0412	Pragun	21	13-12-2001	NaN	NaN	2023	Yes	Yes	Yes
17	19MIC0020	Telavu	21	12-08-2001	9.78	6.0	2023	Yes	No	No
18	20BCE2075	Karishma	22	10-08-2000	9.67	10.0	2024	NaN	NaN	NaN
19	20BCE1099	Lavanya	20	23-09-2002	8.50	9.0	2024	Yes	No	Yes
20	20BCE2222	Preetha	20	13-11-2002	8.23	9.0	2024	NaN	NaN	NaN
21	20BDS0165	Navya	20	13-11-2002	7.98	10.0	2024	No	Yes	Yes

2 Data Manipulation Techniques

2.1 Insertion

```
# Insert a column
data.insert(10,"Govt Exams","Yes")

# Insert a Row
df={
    "Roll No":["20BCE0049"],
    "Name":["Samridh"],
    "Age" :[20],
    "DOB":["10-12-2002"],
    "CGPA":[9.95],
    "Courses":[10],
    "Graduation Year":[2024],
    "Placements":["Yes"],
    "M.Tech/MS":["Yes"],
    "Startup":["No"],
    "Govt Exams":["Yes"]
}
new_row = pd.DataFrame(df)
new_row.to_csv('Untiled.csv', mode='a', index=False, header=False)
data = data.append(new_row)

# Insert a Cell
data.iloc[0,1]="Sahil"
```

	Roll No	Name	Age	DOB	CGPA	Courses	Graduation Year	Placements	M.Tech/MS	Startup	Govt Exams
0	19BCI0876	Sahil	19	23-12-2003	8.45	8.0	2023	Yes	Yes	No	Yes
1	20BCE0076	Ram	20	3-10-2002	6.75	7.0	2024	Yes	No	Yes	Yes
2	20BDS0957	Rishab	21	2-12-2001	7.16	6.0	2024	Yes	No	No	Yes
3	20BDS0294	Sujay	20	02-12-2002	8.02	9.0	2024	No	No	Yes	Yes
4	20BCI0805	Atul	19	12-07-2003	9.14	12.0	2024	Yes	Yes	No	Yes

2.2 Delection

```
# Delete a Column
data = data.drop("Govt Exams",axis=1)

# Delete a Row
data = data.drop(0,axis=0)

# Delete a Cell
data.iloc[1,2]="NaN"
```

	Roll No	Name	Age	DOB	CGPA	Courses	Graduation Year	Placements	M.Tech/MS	Startup
1	20BCE0076	Ram	20	3-10-2002	6.75	7.0	2024	Yes	No	Yes
2	20BDS0957	Rishab	NaN	2-12-2001	7.16	6.0	2024	Yes	No	No
3	20BDS0294	Sujay	20	02-12-2002	8.02	9.0	2024	No	No	Yes
4	20BCI0805	Atul	19	12-07-2003	9.14	12.0	2024	Yes	Yes	No
5	20BKT0012	Nivas	20	3-1-2002	9.54	6.0	2024	No	No	Yes

2.3 Updation

I wanted-ly left out most of the attributes to be NaN because to impute data by using Mean, Median Mode

```
# Update a Column
# arr = ["Yes","No","Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","Yes"]
data[["Govt Exams"]]="No"

# Update a Row
data.iloc[0]=pd.DataFrame({
    "Roll No":"19BCI0876",
    "Name":"Sahil Khan"
})

# Update a Cell
data.iloc[0,2]=20
```

	Roll No	Name	Age	DOB	CGPA	Courses	Graduation Year	Placements	M.Tech/MS	Startup	Govt Exams
1	19BCI0876	Sahil Khan	20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	20BDS0957	Rishab	NaN	2-12-2001	7.16	6.0	2024.0	Yes	No	No	No
3	20BDS0294	Sujay	20	02-12-2002	8.02	9.0	2024.0	No	No	Yes	No
4	20BCI0805	Atul	19	12-07-2003	9.14	12.0	2024.0	Yes	Yes	No	No
5	20BKT0012	Nivas	20	3-1-2002	9.54	6.0	2024.0	No	No	Yes	No

3 Data Pre-Processing

3.1 Statistical Techniques

```
data.dtypes

data.var()

data.isnull().sum()
```

```
Roll No             0
Name                0
Age                 0
DOB                 1
CGPA                2
Courses             2
Graduation Year     1
Placements          4
M.Tech/MS           4
Startup             4
Govt Exams          1
dtype: int64
```

3.2 Replace missing values by mean, median, and mode operations.

```
from sklearn.impute import SimpleImputer
imputer_mode = SimpleImputer(missing_values=np.nan,strategy='most_frequent');
imputer_mean = SimpleImputer(missing_values=np.nan,strategy='mean');
imputer_median = SimpleImputer(missing_values=np.nan,strategy='median');

# FOR MODE
mode_apply = data[["DOB","Placements","M.Tech/MS","Startup","Govt Exams"]]
imputer_mode.fit(mode_apply)
data[["DOB","Placements","M.Tech/MS","Startup","Govt Exams"]] = imputer_mode.transform(
    mode_apply)

# FOR MEDIAN
median_apply = data[["CGPA","Courses"]]
imputer_median.fit(median_apply)
data[["CGPA","Courses"]] = imputer_median.transform(median_apply)

# FOR MEAN
mean_apply = data[["Graduation Year"]]
imputer_mean.fit(mean_apply)
data[["Graduation Year"]] = imputer_mean.transform(mean_apply)
```

```
Roll No             0
Name                0
Age                 0
DOB                 0
CGPA                0
Courses             0
Graduation Year     0
Placements          0
M.Tech/MS           0
Startup             0
Govt Exams          0
dtype: int64
```

3.3 Encoding Techniques

```
# ORDINAL TO CATEGORICAL
# One-hot Encoding (As there's no perfect way to do ione-hot encoding in this dataset)
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder(handle_unknown='ignore')
# enc.fit(X)
data.shape


# BINARY ENCODING
data["Placements"] = data["Placements"].apply(lambda row : 1 if row=='Yes' else 0)
data["Govt Exams"] = data["Govt Exams"].apply(lambda row : 1 if row=='Yes' else 0)
data["Startup"] = data["Startup"].apply(lambda row : 1 if row=='Yes' else 0)
data["M.Tech/MS"]= data["M.Tech/MS"].apply(lambda row : 1 if row == 'Yes' else 0)

# TEXT TO NUMERIC
data["Graduation Year"] = data["Graduation Year"].apply(pd.to_numeric)
data["Placements"] = data["Placements"].apply(pd.to_numeric)
data["Startup"] = data["Startup"].apply(pd.to_numeric)
data["M.Tech/MS"] = data["M.Tech/MS"].apply(pd.to_numeric)
data["Govt Exams"] = data["Govt Exams"].apply(pd.to_numeric)
data["Courses"] = data["Courses"].apply(pd.to_numeric)
data["Age"] = data["Age"].apply(pd.to_numeric)
```

	Roll No	Name	Age	DOB	CGPA	Courses	Graduation Year	Placements	M.Tech/MS	Startup	Govt Exams
1	19BCI0876	Sahil Khan	20	13-11-2002	8.23	9.0	2023.9	1	0	1	0
3	20BDS0294	Sujay	20	02-12-2002	8.02	9.0	2024.0	0	0	1	0
4	20BCI0805	Atul	19	12-07-2003	9.14	12.0	2024.0	1	1	0	0
5	20BKT0012	Nivas	20	3-1-2002	9.54	6.0	2024.0	0	0	1	0
6	20BCT0121	Harshil	19	2-06-2003	8.90	5.0	2024.0	1	0	0	0

4 Normalization Techniques

```
# TRAIN-TEST SPLIT
X = data.drop(["Roll No","Name","DOB",'M.Tech/MS','Startup', 'Govt Exams'],axis=1)
y = data[["CGPA"]]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y , test_size=0.33, random_state
    =42)

# MIN-MAX SCALAR
from sklearn.preprocessing import MinMaxScaler
norm = MinMaxScaler().fit(X_train)
X_train_norm = norm.transform(X_train)
X_test_norm = norm.transform(X_test)

# STANDARD SCALAR
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_X = sc_X.fit_transform(X_train)
```

```
array([[ 0.33333333,  0.4801444 ,  0.5       ,  0.9       ,  1.        ],
       [ 0.33333333,  0.57761733,  0.5       ,  1.        ,  1.        ],
       [ 0.66666667,  1.03971119,  0.125     ,  0.        ,  1.        ],
       [ 0.33333333,  0.40433213,  0.5       ,  1.        ,  0.        ],
       [ 1.        ,  0.23826715,  0.125     ,  1.        ,  1.        ],
       [ 0.33333333, -0.48375451,  0.5       ,  1.        ,  0.        ],
       [ 0.        ,  0.50541516,  0.875     ,  1.        ,  0.        ]])
```

```
array([[-0.17817416,  1.27123032, -1.26491106,  0.28867513, -1.26491106],
       [-0.17817416, -0.05938138,  0.10540926,  0.28867513,  0.79056942],
       [ 2.13808994,  1.40327575,  0.5621827 ,  0.28867513,  0.79056942],
       [-1.33630621, -1.41030776, -0.80813762,  0.28867513,  0.79056942],
       [-1.33630621,  0.86493667,  1.47572957,  0.28867513,  0.79056942],
       [-0.17817416, -1.41030776, -0.35136418,  0.28867513, -1.26491106],
       [-0.17817416, -0.31331491,  0.5621827 ,  0.28867513, -1.26491106],
       [-1.33630621,  0.62116048, -1.7216845 ,  0.28867513,  0.79056942],
       [-0.17817416,  0.46880036, -0.80813762,  0.28867513, -1.26491106],
       [ 0.97995789, -1.0751155 ,  1.93250301,  0.28867513, -1.26491106],
       [-0.17817416,  1.02745413,  0.5621827 ,  0.28867513,  0.79056942],
       [ 0.97995789, -0.05938138,  0.10540926, -3.46410162,  0.79056942],
       [ 0.97995789, -1.32904903, -0.35136418,  0.28867513,  0.79056942]])
```