SACRAMENTO STATE

CSC 177- Data Analytics and Mining

Project 1: Data Preprocessing Project

Team Challengers (23):

1. Srujay Reddy Vangoor
2. Vaibhav Jain
3. Bashar Allwza
4. Varun Bailapudi
5. Uddayankith Chodagam

Comprehensive Set of Techniques for Preprocessing

There are a total 6 types of techniques for preprocessing and they are listed below.

- Data Cleaning
- Dimensionality Reduction
- Feature Engineering
- Sampling Data
- Data Transformation
- Imbalanced Data

Steps for Data Preprocessing

The data which we used in our project has numerical values so before training and testing we done some preprocessing on it.

- Check for null values
- Check for data type
- Analysing Data with Line and pair plots
- Creating heat map to check the correlation
- Dropping Lights and Data because of low correlation
- Checking for duplicates

Steps for Data Preprocessing (Contd.)

- Box plot to check for outliers
- Calculate Z-Score
- Removed rows where z-score is >3 and <-3
- Sorting the data frame
- Splitting into input and output feature
- Performing min-max scaling

Check for outliers

An Outlier is a data-item/object that deviates significantly from the rest of the (so-called normal)objects.

They can be caused by measurement or execution errors. The analysis for outlier detection is referred toas outlier mining. There are many ways to detect the outliers, and the removal process is the data frame same as removing a data item from the panda's data frame.

Outliers can be detected using visualization, implementing mathematical formulas on the dataset, or using the statistical approach.

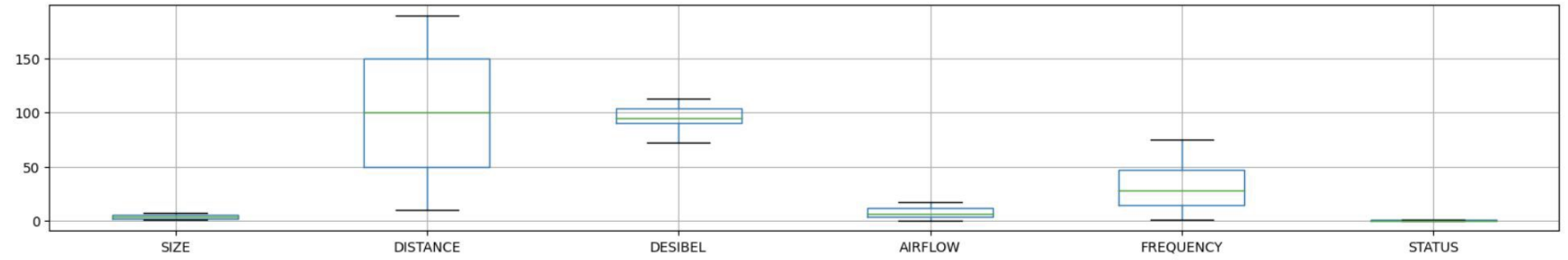
Check for outliers (Contd.)

Check for outliers

```
%matplotlib inline

data.boxplot(figsize=(20,3))
```

```
<AxesSubplot: >
```



HeatMap

Heatmaps visualize the data in 2-D colored maps making use of color variations like hue, saturation, or luminance. Heatmaps describe relationships between variables in form of colors instead of numbers.

We can create a basic heatmap using the sns.heatmap() function:

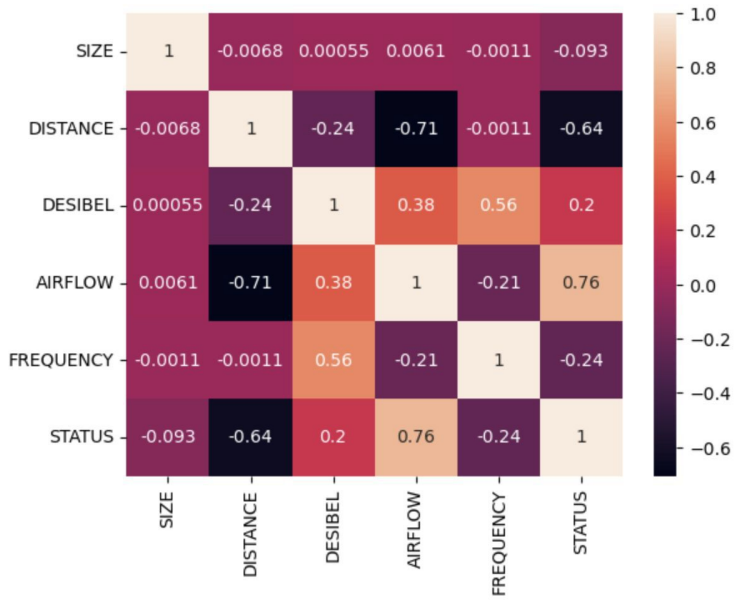
sns.heatmap(df)

HeatMap (Contd.)

Generate Heatmap

```
import seaborn as sn
corr = data.corr(numeric_only = True)
sn.heatmap(corr, annot = True)
```

```
<AxesSubplot: >
```



Perform Standardizing

To standardize the data , Z-Score is a very popular method in pandas that is used to standardize the data. Z-Score will tell us how many standard deviations away a value is from the mean. When we standardize the data the data will be changed into a specific form where the graph of its frequency will form a bell curve.

Perform Standardizing (Contd.)

Perform Standardizing

```
Z = (data-data.mean(numeric_only=True))/data.std(numeric_only=True)
```

```
print('Number of rows before discarding outliers = %d' % (Z.shape[0]))

Z2 = Z.loc[((Z > -3).sum(axis=1)==9) & ((Z <= 3).sum(axis=1)==9),:]
print('Number of rows after discarding missing values = %d' % (Z2.shape[0]))
```

```
Number of rows before discarding outliers = 17379
Number of rows after discarding missing values = 0
```

```
Z['FUEL'] = data['FUEL']
```

Shuffle the dataset and sort dataframe

Because our data is often sorted in a particular way (say, for example, by date or by geographical area), we want to make sure that our data is representative. Because of this, we will want to shuffle our Pandas dataframe prior to taking on any modelling. Shuffling data has the effect of reducing variance and making sure that models remain general and overfit less.

We will be using the sample() method of the pandas module to randomly shuffle DataFrame rows in Pandas. In order to sort the dataframe in pandas, function sort_values() is used. Pandas sort_values() can sort the data frame in ascending or descending order.

Shuffle the dataset and sort dataframe (Contd.)

Shuffle the dataset

```
data = data.sample(frac=1).reset_index(drop=True)
```

```
data.head()
```

	SIZE	FUEL	DISTANCE	DESIBEL	AIRFLOW	FREQUENCY	STATUS
0	6	lpg	40.0	107.0	14.5	46.0	1
1	7	lpg	50.0	105.0	12.5	21.0	1
2	2	kerosene	10.0	108.0	15.4	22.0	1
3	5	thinner	180.0	91.0	2.5	19.0	0
4	7	lpg	170.0	85.0	3.3	5.0	0

Sorting dataframe

```
data = data.sort_values(by='DISTANCE',ascending=True)
```

```
data
```

	SIZE	FUEL	DISTANCE	DESIBEL	AIRFLOW	FREQUENCY	STATUS
1860	4	thinner	10.0	90.0	11.9	12.0	1
5584	4	thinner	10.0	106.0	15.4	40.0	1
10038	5	thinner	10.0	106.0	15.4	40.0	1
11583	4	kerosene	10.0	108.0	15.2	19.0	1
14425	3	kerosene	10.0	96.0	0.0	72.0	0
...	...	...	...	...	...	...	...
8636	4	kerosene	190.0	103.0	3.2	42.0	0
10919	2	gasoline	190.0	91.0	2.5	20.0	0
4956	1	gasoline	190.0	90.0	3.6	15.0	0
13701	2	gasoline	190.0	97.0	2.6	28.0	0
3395	4	kerosene	190.0	94.0	2.9	14.0	0

17379 rows × 7 columns

Label encoding for categorical data

1. Create an instance of LabelEncoder() and store it in labelencoder variable/object.

2. Apply fit and transform which does the trick to assign numerical value to categorical value and the same is stored in new column called “encoded_FUEL”.

3. Note that we have added a new column called “encoded_FUEL” which contains numerical value associated to categorical value and still the column called State is present in the dataframe. This column needs to be removed before we feed the final preprocess data.

Label encoding for categorical data (Contd.)

Label encoding for categorical data

```
from sklearn import preprocessing

le = preprocessing.LabelEncoder()
data['encoded_FUEL'] = le.fit_transform(data['FUEL'])
```

```
data
```

	SIZE	FUEL	DISTANCE	DESIBEL	AIRFLOW	STATUS	encoded_FUEL
4112	7	lpg	60.0	108.0	12.6	1	2
1529	4	kerosene	80.0	96.0	2.0	0	1
9211	2	kerosene	130.0	103.0	3.7	0	1
4928	2	kerosene	140.0	90.0	4.5	0	1
9002	6	lpg	170.0	93.0	4.2	0	2
...	...	...	...	...	...	...	...
7154	6	lpg	110.0	106.0	2.7	0	2
16541	2	kerosene	10.0	88.0	8.8	1	1
13094	2	gasoline	30.0	110.0	15.2	1	0
9525	3	kerosene	180.0	91.0	3.2	0	1
16135	5	thinner	100.0	87.0	8.5	0	3

17379 rows × 7 columns

Splitting dataframe

```
#We split the data into test and train set with trainset having 75% data and test set having 25% data
```

```
# Split into train/test
x_train, x_test, y_train, y_test = train_test_split(data[["SIZE","FUEL","DISTANCE","DESIBEL","AIRFLOW"]], data["STAT
```

```
x_train, x_test, y_train, y_test
```

```
(       SIZE      FUEL  DISTANCE  DESIBEL  AIRFLOW
 6600      4  kerosene     180.0     91.0      2.9
 4217      2   thinner      40.0     96.0      0.0
 11500     3   thinner     110.0    104.0      8.7
 15926     4   thinner     120.0     90.0      4.2
 14153     5  kerosene      20.0    110.0     15.0
 ...     ...       ...       ...      ...      ...
 2465      4   thinner      50.0     90.0     11.0
 14276     5  gasoline     180.0     93.0      3.4
 16814     2  gasoline      80.0    106.0     12.3
 9748      1   thinner      60.0    108.0     14.1
 14358     3  gasoline      70.0    103.0      9.5

 [13034 rows x 5 columns],
       SIZE      FUEL  DISTANCE  DESIBEL  AIRFLOW
 6262      5  gasoline     190.0    104.0      0.0
 5117      2   thinner      10.0     90.0     11.0
 7706      5  gasoline     190.0     92.0      2.3
 7598      6       lpg      10.0     93.0     15.4
 9908      2   thinner      10.0    110.0     15.2
 ...     ...       ...       ...      ...      ...
 8754      2  gasoline     110.0     96.0      0.0
 7137      3  gasoline      80.0    103.0      8.2
 12838     2   thinner      10.0     96.0      0.0
 7482      4   thinner      60.0     90.0      9.6
 16658     1  kerosene      90.0     75.0      0.0
```

Means and Standard Deviations on data

The mean is the sum of all the entries divided by the number of entries. Standard Deviation is a measure of spread in Statistics. It is used to quantify the measure of spread, variation of a set of data values. A low measure of Standard Deviation indicates that the data are less spread out, whereas a high value of Standard Deviation shows that the data in a set are spread apart from their mean average values.

Means and Standard Deviations on data (Contd.)

```
# Mean and Standard Deviation for train data
```

```
SIZE          3.420209
DISTANCE    100.101274
DESIBEL      96.419268
AIRFLOW       6.987825
dtype: float64
```

```
x_train.mean(numeric_only=True)
```

```
SIZE          3.420209
DISTANCE    100.101274
DESIBEL      96.419268
AIRFLOW       6.987825
dtype: float64
```

```
x_train.std(numeric_only=True)
```

```
SIZE          1.752277
DISTANCE     54.498309
DESIBEL       8.150154
AIRFLOW       4.722938
dtype: float64
```

```
# Mean and Standard Deviation for test data
```

```
x_test.mean(numeric_only=True)
```

```
SIZE          3.421404
DISTANCE    100.764097
DESIBEL      96.245740
AIRFLOW       6.855339
dtype: float64
```

```
x_test.std(numeric_only=True)
```

```
SIZE          1.735729
DISTANCE     55.253607
DESIBEL       8.169883
AIRFLOW       4.734388
dtype: float64
```

Discretization

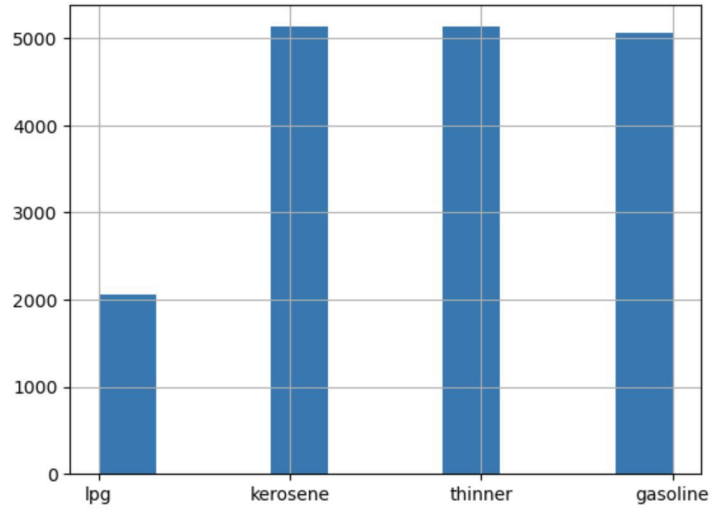
Discretization is the process of converting continuous data into discrete buckets by grouping it. Discretization is also known for easy maintainability of the data. Training a model with discrete data becomes faster and more effective than when attempting the same with continuous data. Here the value counts gives how many unique data elements are present and count of each of these unique elements.

Discretization (Contd.)

Discretization

```
data['FUEL'].hist(bins=10)
data['FUEL'].value_counts(sort=False)
```

```
lpg         2052
kerosene    5130
thinner     5130
gasoline    5067
Name: FUEL, dtype: int64
```



Principal Component Analysis

Principal Component Analysis is basically a statistical procedure to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables.

Each of the principal components is chosen in such a way that it would describe most of them still available variance and all these principal components are orthogonal to each other. In all principal components, first principal component has a maximum variance.

Human insights on preprocessed data

- The data can be used for Energy Prediction of Household Appliances.
- The data was pre-processed keeping in mind that the result data can be directly given to the model for prediction.
- As such the correlation was determined with heat-maps and made sure that the data will not negatively affect the predictions that are going to be done my the Neural Network model.
- Min Max scaling was also done so as to decrease the prediction deviation.

Thank you!!