CSB352: Data Mining LAB

Instructor : [Dr. Chandra Prakash]

- For more information visit the class website.
- DATE: 01-Feb-2021

LAB 5: Data Pre-Processing

Feature Engineering in ML

1. Data Visualization
2. Data Pre-processing
    - Missing Values
    - Handling Outliers
    - Label Encoding
    - Log Transform
        - handle skewed data and after transformation, the distribution becomes more approximate to normal.
        - decreases the effect of the outliers
    - Binning
        - Binning can be applied on both categorical and numerical data.
        - Model more robust and prevent overfitting
    - Scaling/ Normalization
        - Need of Normalization
        - Type of Normalization
            - Min-Max Normalization
            - Standard Normalization / Standardization
            - Z-score Normalization
    - Feature Split / Joining

Need of Pre-Processing

title

PART 4.1: Tabular data Pre-processing

- Structured data - Tabular data
- Semistructured data - Json data
- Unstructured data - Text file

Downlaod the DATA

```
from datetime import datetime

try:
    from google.colab import drive
    %tensorflow_version 2.x
    COLAB = True
    print("Assignment 4")
    print("Note: using Google CoLab")
except:
    print("Assignment 4")
    print("Note: not using Google CoLab")
    COLAB = False

# Print your name and Roll No.
print('Name :Rohit Byas')
print('Roll No. : 181210043')
# Print the curent time

now = datetime.now()

current_time = now.strftime("%H:%M:%S")
print("Current Time =", current_time)
```

```
Assignment 4
Note: using Google CoLab
Name :Rohit Byas
Roll No. : 181210043
Current Time = 08:56:34
```

Data reading and setup

```
import numpy as np
import pandas as pd


# Read the Dataset

data_actuals=pd.read_csv('data/actuals.csv')
data_schedules=pd.read_csv('data/schedules.csv')
data_drones=pd.read_csv('data/drones.csv')

data_services=pd.read_csv('data/services.csv')
data_stations=pd.read_csv('data/stations.csv')


# Show the top 5 rows of actuals.csv

data_actuals.head()
```

	actual_arrival_datetime	actual_departure_datetime	actual_fuel_consumption_gm_per_
0	2050-07-01 05:06	2050-07-01 06:00	5
1	2050-07-01 06:19	2050-07-01 07:00	16
2	2050-07-01 07:09	2050-07-01 08:00	7
3	2050-07-01 08:21	2050-07-01 09:00	16
4	2050-07-01 09:09	2050-07-01 10:00	6

```
# Show the top 5 rows of schedules.csv
data_schedules.head()
```

	forecast_windspeed_kts	scheduled_arrival_datetime	scheduled_departure_datetime	s
0	4	2050-07-01 05:06	2050-07-01 06:00	
1	7	2050-07-01 06:19	2050-07-01 07:00	
2	6	2050-07-01 07:09	2050-07-01 08:00	
3	8	2050-07-01 08:21	2050-07-01 09:00	
4	13	2050-07-01 09:09	2050-07-01 10:00	

```
# Show the top 5 rows of drones.csv

data_drones.head()
```

	capacity_persons	drone_id	fuel_consumption_gm_per_mile	manufacturer	max_altitud
0	114	971-LGB	1.87	Sharp Ltd	8
1	438	ZTG 357	5.10	Freeman-Garrison	7
2	493	I55 1AC	0.79	Perez Inc	4
3	154	XUW-	2.84	Sharp Ltd	9

Merge DATA

```
# HOW  to Merge df_schedules and df_drones data
# Check the merge command help

## WRITE YOUR CODE
final_data = data_schedules.merge(data_drones,on='service_id')
final_data.head()
```

	forecast_windspeed_kts	scheduled_arrival_datetime	scheduled_departure_datetime
0	4	2050-07-01 05:06	2050-07-01 06:00
1	7	2050-07-01 06:19	2050-07-01 07:00
2	6	2050-07-01 07:09	2050-07-01 08:00
3	8	2050-07-01 08:21	2050-07-01 09:00
4	13	2050-07-01 09:09	2050-07-01 10:00

Drop a column /Row

```
# Check the drop command help and write the syntex
```

```
clm_to_drop = ['scheduled_arrival_datetime','scheduled_departure_datetime','service_id','stat
```

```
final_data
```

	forecast_windspeed_kts	scheduled_arrival_datetime	scheduled_departure_datetim
0	4	2050-07-01 05:06	2050-07-01 06:0
1	7	2050-07-01 06:19	2050-07-01 07:0
2	6	2050-07-01 07:09	2050-07-01 08:0
3	8	2050-07-01 08:21	2050-07-01 09:0
4	13	2050-07-01 09:09	2050-07-01 10:0
...	...	...	.
12489	1	2050-07-30 07:54	2050-07-30 12:0
12490	1	2050-07-30 19:36	2050-07-30 22:0
12491	0	2050-07-31 04:54	2050-07-31 07:0
12493	6	2050-08-01 06:36	2050-08-01 09:0

```
final_data.drop(columns=clm_to_drop,inplace=True)
# Those values were dropped and the changes were made in the original data frame since inplac
```

```
final_data
```

	forecast_windspeed_kts	station_call_id	capacity_persons	fuel_consumptio
0	4	JP_0_205007010506	114	
1	7	JP_0_205007010619	114	
2	6	JP_0_205007010709	114	
3	8	JP_0_205007010821	114	
4	13	JP_0_205007010909	114	
...	...	...	...	
12489	1	World_3_205007300754	1066	
12490	1	World_3_205007301936	1066	
12491	0	World_3_205007310454	1066	
12492	2	World_3_205007311854	1066	
12493	6	World_3_205008010636	1066	

12494 rows × 9 columns

Divide DATA into Training and Testing

```
traning_data = final_data.merge(data_actuals[['arrival_delay_seconds','station_call_id']],on=
test_data = final_data[~final_data['station_call_id'].isin(traning_data['station_call_id'].tc
```

```
# Display the traning_data
traning_data
```

	forecast_windspeed_kts	station_call_id	capacity_persons	fuel_consumption_
0	4	JP_0_205007010506	114	
1	7	JP_0_205007010619	114	
2	6	JP_0_205007010709	114	
3	8	JP_0_205007010821	114	
4	13	JP_0_205007010909	114	
...	...	...	...	
9919	0	World_3_205007232154	1066	
9920	0	World_3_205007241154	1066	
9921	0	World_3_205007242336	1066	
9922	1	World_3_205007250854	1066	
9923	0	World_3_205007252254	1066	

9924 rows × 10 columns

```
# Display the test_data
```

```
from sklearn.preprocessing import LabelEncoder,StandardScaler
```

```
# Drop station_call_id from traning_data and test_data dataframe
```

```
traning_data.drop(columns=['station_call_id'],inplace=True)
test_data.drop(columns=['station_call_id'],inplace=True)
```

```
/usr/local/lib/python3.6/dist-packages/pandas/core/frame.py:4174: SettingWithCopyWarnin
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user
  errors=errors,
```

```
traning_data.head()
```

	forecast_windspeed_kts	capacity_persons	fuel_consumption_gm_per_mile	max_altitud
0	4	114	1.87	8
1	7	114	1.87	8
2	6	114	1.87	8
3	8	114	1.87	8

```
test_data.head()
```

	forecast_windspeed_kts	capacity_persons	fuel_consumption_gm_per_mile	max_altit
15	10	114	1.87	
28	2	114	1.87	
145	19	114	1.87	
273	11	114	1.87	
422	20	114	1.87	

```
features = list(traning_data.columns)
```

```
features
```

```
['forecast_windspeed_kts',
 'capacity_persons',
 'fuel_consumption_gm_per_mile',
 'max_altitude_ft',
 'max_capacity_fuel_gm',
 'min_capacity_fuel_gm',
 'operating_speed_mph',
 'optimal_altitude_ft',
 'arrival_delay_seconds']
```

remove() is an inbuilt function in Python programming language that removes a given object from the list. It does not return any value.

```
#remove the arrival_delay_seconds  object from the list

features.remove('arrival_delay_seconds')
```

```
features
```

```
['forecast_windspeed_kts',
      'capacity_persons',
      'fuel_consumption_gm_per_mile',
      'max_altitude_ft',
      'max_capacity_fuel_gm',
      'min_capacity_fuel_gm',
      'operating_speed_mph',
      'optimal_altitude_ft']
```

Training part as train X and Train Y

```
trainX,trainY = traning_data[features].values,traning_data['arrival_delay_seconds'].values
```

```
testX = test_data.values
```

```
trainY
```

```
     array([  2,   5,   2, ..., 657, 596, 686])
```

```
trainX.shape,trainY.shape,testX.shape
```

```
     ((9924, 8), (9924,), (2570, 8))
```

Normalization

- Need of Normalization
- Type of Normalization
- Min-Max Normalization:

z= (x-min)/max-min

- Standard Normalization : Z Normalization(Standardization):

z=x-x(mean)/standered deviation

- Nean or Normal Mean

```
StandardScaler?
```

```
scaler = StandardScaler()
```

Z = x-x.mean()/x.std()

```
scaler.fit(trainX)
```

```
    StandardScaler(copy=True, with_mean=True, with_std=True)
```

```
scaler.transform?
```

```
trainX_norm = scaler.transform(trainX)
testX_norm = scaler.transform(testX)
```

```
trainX.max(),trainX.min(),trainX.mean()
```

```
    (28280.0, 0.0, 2008.0340089681579)
```

```
trainX_norm.max(),trainX_norm.min(),trainX_norm.mean()
```

```
    (6.617838095190163, -1.4182271125805397, -1.0023779021202541e-17)
```

Label Encoding

DATA

```
 S.No   Country   Age    Salary


 0   India   44   72000
 1      US   34   65000
 2   Japan   46   98000
 3      US   35   45000
 4   Japan   23   34000
```

[ ] ↳ *5 cells hidden*

Missing Values

```
dataset = pd.read_csv('data/pima-indians-diabetes1.csv', header=None)
```

```
dataset.head()
```

	0	1	2	3	4	5	6	7	8
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0

```
dataset.describe()
```

	0	1	2	3	4	5	6
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000

```
num_missing = (dataset[[1,2,3,4,5,7]] == 0).sum()
```

```
num_missing
```

```
1      5
2     35
3    227
4    374
5     11
7      0
dtype: int64
```

```
dataset[[1,2,3,4,5]] = dataset[[1,2,3,4,5]].replace(0, np.nan)
dataset.isnull().sum()
```

```
0      0
1      5
2     35
3    227
4    374
5     11
6      0
7      0
8      0
dtype: int64
```

```
dataset.head()
```

	0	1	2	3	4	5	6	7	8
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1

Remove rows

[ ] ↳ *1 cell hidden*

Replace missing value

[ ] ↳ *8 cells hidden*

Class Imbalance - SMOTE

https://imbalanced-learn.readthedocs.io/en/stable/over_sampling.html

image.png

```
!pip install imblearn --user # in windows
```

```
Requirement already satisfied: imblearn in /usr/local/lib/python3.6/dist-packages (0.0)
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: numpy>=1.8.2 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: scipy>=0.13.3 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: scikit-learn>=0.20 in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (
```

```
import imblearn
from collections import Counter
from sklearn.datasets import make_classification
from imblearn.over_sampling import SMOTE # doctest: +NORMALIZE_WHITESPACE
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/externals/six.py:31: FutureWarning: The
  "(https://pypi.org/project/six/).", FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning:
  warnings.warn(message, FutureWarning)
```

```
X, y = make_classification(n_classes=2, class_sep=2,
weights=[0.1, 0.9], n_informative=3, n_redundant=1, flip_y=0,
n_features=20, n_clusters_per_class=1, n_samples=1000, random_state=10)
```

```
print('Original dataset shape %s' % Counter(y))
```

```
    Original dataset shape Counter({1: 900, 0: 100})
```

```
sm = SMOTE(random_state=42)
```

```
X_res, y_res = sm.fit_resample(X, y)
```

```
    /usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning:
      warnings.warn(msg, category=FutureWarning)
```

```
print('Resampled dataset shape %s' % Counter(y_res))
```

```
    Resampled dataset shape Counter({0: 900, 1: 900})
```

Check out https://rikunert.com/SMOTE_explained for details