Computer Engineering Department National University of Technology Islamabad, Pakistan

Introduction to Data Mining

Practice Exercise 05

Name: Muhammad Sami Uddin Rafay

Roll Number: F18604013

Submitted To: Dr. Kamran Javed

Date: 13 December 2020

Practice Exercise 05

Feature Extraction

Objective:

- To reduce dimensions of a weather data set to get the most relevant features.

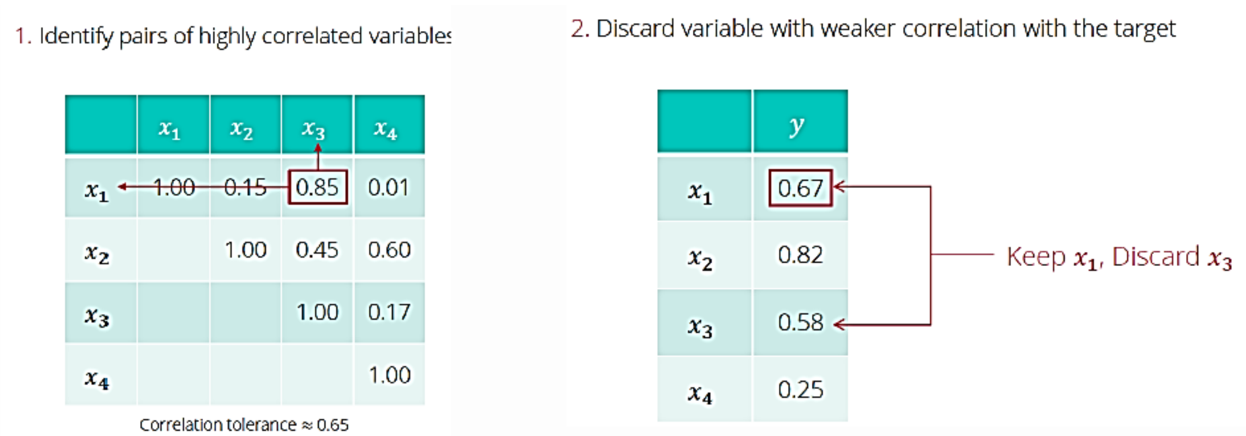
Equipment/Software Required:

- Python (Spyder 4.0 Anaconda Distribution)

Background:

Tasks:

1. Plot weather data.
2. Find missing value ratio for each column and if its greater than 40% then drop those features. $\left(\frac{\text{number of records with missing value}}{\text{number of total record}}\right)$.
3. Normalize remaining columns.
4. Plot the reduced data set.
5. Find standard deviation of all features and drop the features with minimum variance.
6. Plot the reduced data set.



	x_1	x_2	x_3	x_4
x_1	1.00	0.15	$\boxed{0.85}$	0.01
x_2		1.00	0.45	0.60
x_3			1.00	0.17
x_4				1.00

	y
x_1	$\boxed{0.67}$
x_2	0.82
x_3	0.58
x_4	0.25

7. Find correlation of normalized features with one another and analyze which features have high value of correlation. Now identify highly correlated features with the target and retain variables with higher value of correlation i.e., if correlation between two features is high then retain the one whose correlation value is higher with the target.

Code:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from scipy.stats import pearsonr

file=pd.read_excel(r'C:\Users\User\Desktop\Practical5.xlsx')
frame=pd.DataFrame(file)
print("Dimensions of Given Dataset : ",len(frame.columns))
```

```
print("\n")
print("Printing all columns of given Data Frame:")
print("\n")
print(frame.columns)
print("\n")

Pressure = frame["Pressure (millibars)"]
Temperature = frame["Apparent Temperature (C)"]
Wind_Bearing = frame["Wind Bearing (degrees)"]
Visibility = frame["Visibility (km)"]
Wind_Speed = frame["Wind Speed (km/h)"]
Humidity = frame["Humidity"]
Target = frame["Target"]

plt.figure(1, figsize=(8,6))

plt.subplot(421)
plt.plot(Target, 'k')
plt.xlabel("Time")
plt.ylabel("Target")
plt.grid()
plt.tight_layout()

plt.subplot(422)
plt.plot(Temperature, 'r')
plt.xlabel("Time")
plt.ylabel("Temperature")
plt.grid()
plt.tight_layout()

plt.subplot(423)
plt.plot(Humidity, 'b')
plt.xlabel("Time")
plt.ylabel("Humidity")
plt.grid()
plt.tight_layout()

plt.subplot(424)
plt.plot(Wind_Speed, 'g')
plt.xlabel("Time")
plt.ylabel("Wind Speed")
plt.grid()
plt.tight_layout()

plt.subplot(425)
plt.plot(Visibility, 'y')
plt.xlabel("Time")
plt.ylabel("Visibility")
plt.grid()
plt.tight_layout()

plt.subplot(426)
plt.plot(Pressure, 'c')
plt.xlabel("Time")
plt.ylabel("Pressure")
```

```
plt.grid()
plt.tight_layout()

plt.subplot(427)
plt.plot(Wind_Bearing, 'm')
plt.xlabel("Time")
plt.ylabel("Wind Bearing")
plt.grid()
plt.tight_layout()

plt.show()

# finding the percentage of missing values in each column dataframe
for feature in frame.columns:
    missing_values_Percentage=frame[feature].isnull().sum()/len(frame[feature])*100
    # Droping the Feature with missing values > 40.0%
    if (missing_values_Percentage>40.0):
        #size=size-1
        frame=frame.drop(feature, axis = 1)

print("Dimensions of Dataset after droping features : ",len(frame.columns))
print("\n")
print("Printing all Features of given Data Frame after dropping Features with Missing values > 40% :")
print("\n")
print(frame.columns)
print("\n")

# Normalize the reduced dataset
# There is command for normalization in python different libraries (Pandas, Scikit-learn)
# but doing this with self-developed algorithm is more understandable

def normalize(dataframe):
    for feature in dataframe.columns:
        dataframe[feature]=((dataframe[feature]-dataframe[feature].min())/(dataframe[feature].max()-
dataframe[feature].min()))
        #dataframe=dataframe[feature]
    return dataframe

Normalized_dataframe=normalize(frame)

print(Normalized_dataframe)


plt.figure(2, figsize=(7,5))

plt.subplot(321)
sns.distplot(Normalized_dataframe["Target"],color='k')
plt.xlabel("Normalized Target")
plt.grid()
plt.tight_layout()

plt.subplot(322)
sns.distplot(Normalized_dataframe["Apparent Temperature (C)"],color='r')
plt.xlabel("Normalized Apparent Temperature")
plt.grid()
```

```
plt.tight_layout()

plt.subplot(323)
sns.distplot(Normalized_dataframe["Humidity"],color='b')
plt.xlabel("Normalized Humidity")
plt.grid()
plt.tight_layout()

plt.subplot(324)
sns.distplot(Normalized_dataframe["Visibility (km)"],color='y')
plt.xlabel("Normalized Visibility")
plt.grid()
plt.tight_layout()

plt.subplot(325)
sns.distplot(Normalized_dataframe["Pressure (millibars)"], color='c')
plt.xlabel("Normalized Pressure")
plt.grid()
plt.tight_layout()

plt.show()
```

 # 5. Find standard deviation of all features and drop the features with minimum variance.

```
def standard_deviation(dataframe):
    std=np.array([])
    for feature in dataframe.columns:
        std=np.append(std,np.std(dataframe[feature], axis=0))
    return std

std=standard_deviation(frame)
print("\n")
print("Standard Deviation of each column : ")
print("\n")
print(std)
```

#Calculating Variance of each column

```
print(frame.var())

#Output:
#Target                    0.087380
#Apparent Temperature (C)    0.097572
#Humidity                  0.075583
#Visibility (km)            0.170906
#Pressure (millibars)        0.035434
#dtype: float64
```

 # As we can see that the Humidity and Pressure have vary low variance so we
 # will drop them

```
file.drop(['Pressure (millibars)'],inplace=True, axis=1)
file.drop(['Humidity'],inplace=True, axis=1)

plt.figure(3)
```

```
plt.subplot(221)
plt.plot(file["Target"])
plt.title("Target")

plt.subplot(222)
plt.plot(file["Visibility (km)"])
plt.title("Visibility (km)")

plt.subplot(223)
plt.plot(file["Apparent Temperature (C)"])
plt.title("Apparent Temperature (C)")

plt.subplot(224)
plt.plot(file["Visibility (km)"])
plt.title("Visibility (km)")


# Task No. 07

print("Pearson Correlation of all Normalized Features with Temperature")
print("\n")
TcorrP, _=pearsonr(Normalized_dataframe["Apparent Temperature (C)"], Normalized_dataframe["Pressure (millibars)"])
print(TcorrP)
TcorrT,_=pearsonr(Normalized_dataframe["Apparent Temperature (C)"], Normalized_dataframe["Target"])
print(TcorrT)
TcorrV,_=pearsonr(Normalized_dataframe["Apparent Temperature (C)"], Normalized_dataframe["Visibility (km)"])
print(TcorrV)
TcorrH, _=pearsonr(Normalized_dataframe["Apparent Temperature (C)"], Normalized_dataframe["Humidity"])
print(TcorrH)
print("\n")
print("\n")

print("Pearson Correlation of all Normalized Features with Humidity")
print("\n")
HcorrP, _=pearsonr(Normalized_dataframe["Humidity"], Normalized_dataframe["Pressure (millibars)"])
print(HcorrP)
HcorrT,_=pearsonr(Normalized_dataframe["Humidity"], Normalized_dataframe["Apparent Temperature (C)"])
print(HcorrT)
HcorrV,_=pearsonr(Normalized_dataframe["Humidity"], Normalized_dataframe["Visibility (km)"])
print(HcorrV)
HcorrT, _=pearsonr(Normalized_dataframe["Humidity"], Normalized_dataframe["Target"])
print(HcorrT)
print("\n")
print("\n")

print("Pearson Correlation of all Normalized Features with Pressure")
print("\n")
PcorrH, _=pearsonr(Normalized_dataframe["Pressure (millibars)"], Normalized_dataframe["Humidity"])
print(PcorrH)
PcorrT,_=pearsonr(Normalized_dataframe["Pressure (millibars)"], Normalized_dataframe["Apparent Temperature (C)"])
```

```
print(PcorrT)
PcorrV,_=pearsonr(Normalized_dataframe["Pressure (millibars)"], Normalized_dataframe["Visibility (km)"])
print(PcorrV)
PcorrT, _=pearsonr(Normalized_dataframe["Pressure (millibars)"], Normalized_dataframe["Target"])
print(PcorrT)
print("\n")
print("\n")

print("Pearson Correlation of all Normalized Features with Visibility")
print("\n")
VcorrH, _=pearsonr(Normalized_dataframe["Visibility (km)"], Normalized_dataframe["Humidity"])
print(VcorrH)
VcorrT,_=pearsonr(Normalized_dataframe["Visibility (km)"], Normalized_dataframe["Apparent Temperature (C)"])
print(VcorrT)
VcorrP,_=pearsonr(Normalized_dataframe["Visibility (km)"], Normalized_dataframe["Pressure (millibars)"])
print(VcorrP)
VcorrT, _=pearsonr(Normalized_dataframe["Visibility (km)"], Normalized_dataframe["Target"])
print(VcorrT)
print("\n")
print("\n")

print("Pearson Correlation of all Normalized Features with the Target")
print("\n")
TcorrP, _=pearsonr(Normalized_dataframe["Target"], Normalized_dataframe["Pressure (millibars)"])
print()
print(TcorrP)
TcorrT,_=pearsonr(Normalized_dataframe["Target"], Normalized_dataframe["Apparent Temperature (C)"])
print(TcorrT)
TcorrV,_=pearsonr(Normalized_dataframe["Target"], Normalized_dataframe["Visibility (km)"])
print(TcorrV)
TcorrH, _=pearsonr(Normalized_dataframe["Target"], Normalized_dataframe["Humidity"])
print(TcorrH)
print("\n")
print("\n")

######################################################################################
Target=np.array(Target)
Temperature=np.array(Temperature)
Humidity=np.array(Humidity)

fig = plt.figure(4)
ax = plt.axes(projection='3d')
ax.scatter(Target, Temperature, Humidity, c='c')
ax.set_xlabel("Target")
ax.set_ylabel("Temperature")
ax.set_zlabel("Humidity")
ax.set_title('3d Scatter plot')
plt.show()
```

Output:

Dimensions of Given Dataset : 7

Printing all columns of given Data Frame:

```
Index(['Target', 'Apparent Temperature (C)', 'Humidity', 'Wind Speed (km/h)',
       'Visibility (km)', 'Pressure (millibars)', 'Wind Bearing (degrees)'],
      dtype='object')
```

Dimensions of Dataset after droping features : 5

Printing all Features of given Data Frame after dropping Features with Missing values > 40% :

```
Index(['Target', 'Apparent Temperature (C)', 'Humidity', 'Visibility (km)',
       'Pressure (millibars)'],
      dtype='object')
```

Standard Deviation of each column :

```
[0.29063236 0.30711476 0.27030354 0.40645883 0.18507453]
Target                      0.087380
Apparent Temperature (C)    0.097572
Humidity                    0.075583
Visibility (km)             0.170906
Pressure (millibars)        0.035434
dtype: float64
```

Pearson Correlation of all Normalized Features with Temperature

```
0.35684843008351375
0.9861203749987968
-0.5956830754448121
-0.8669382956595971
```

Pearson Correlation of all Normalized Features with Humidity

```
-0.26426160487859685
-0.8669382956595971
0.3538413899391059
-0.8849829679901399
```

Pearson Correlation of all Normalized Features with Pressure

```
-0.26426160487859685
```

0.35684843008351375
-0.5236246574793217
0.4009226334203969

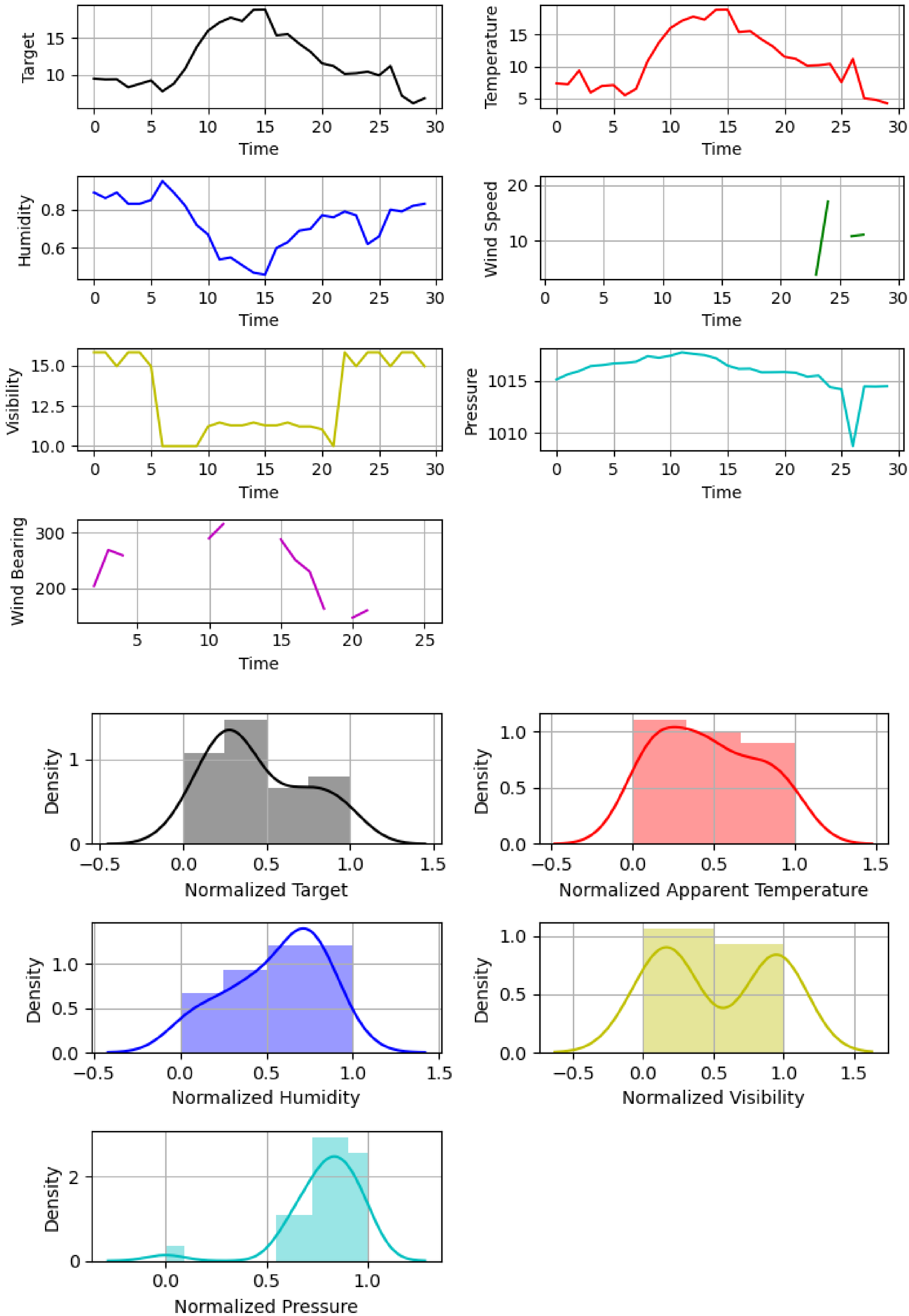
Pearson Correlation of all Normalized Features with Visibility

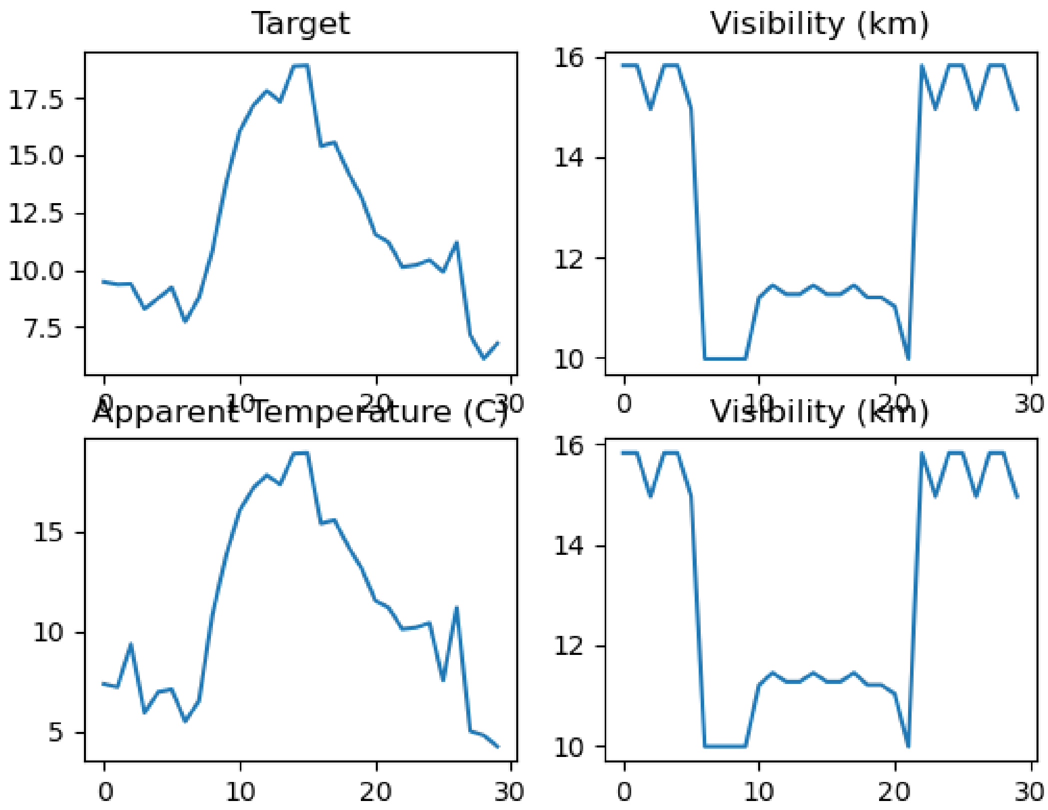
0.3538413899391059
-0.5956830754448121
-0.5236246574793217
-0.5898066921031413

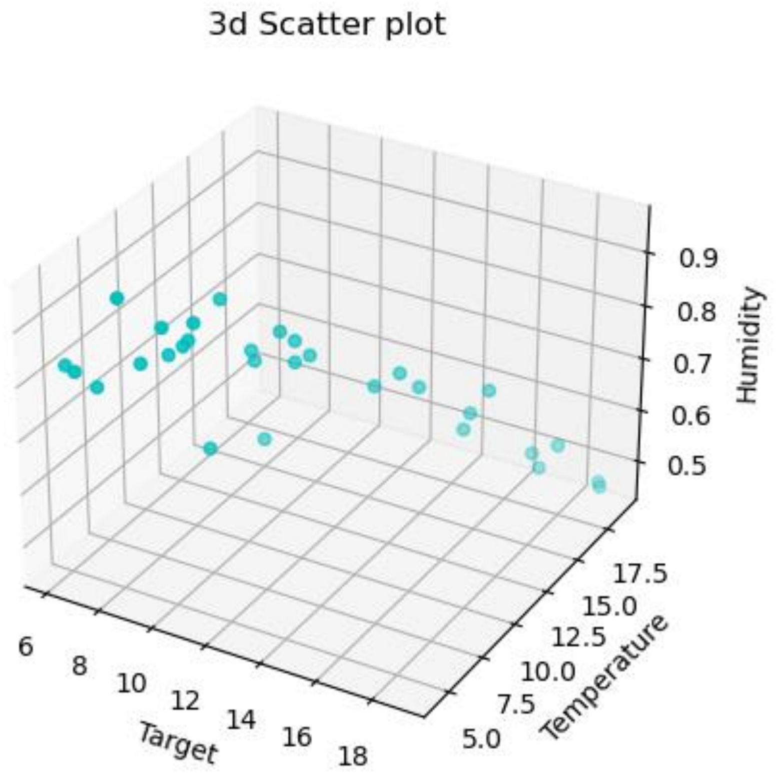
Pearson Correlation of all Normalized Features with the Target

0.4009226334203969
0.9861203749987968
-0.5898066921031413
-0.8849829679901399

Graphs:


Target
Time
Temperature
Time
Humidity
Time
Wind Speed
Time
Visibility
Time
Pressure
Time
Wind Bearing
Time
Density
Normalized Target
Density
Normalized Apparent Temperature
Density
Normalized Humidity
Density
Normalized Visibility
Density
Normalized Pressure


Target
Visibility (km)
Apparent Temperature (C)
Visibility (km)


3d Scatter plot
Target
Temperature
Humidity

Results and Discussions:

In this practical I learned dimensionality reduction which is very helpful for feature extraction and this reduced dataset is the actual input of an machine learning model for obtaining respective outcomes.

The python packages I used in this practical: -

- ✓ NumPy
- ✓ matplotlib
- ✓ pandas

Conclusion:

What I found about Feature Extraction is used when we need to reduce dimensions of a dataset without losing important and respective information by dropping features having missing values greater than 40% and having low variance correlation.
.py file is attached.