IMPLEMENT A SYSTEM THAT CAN PREDICT THE PM2.5 LEVEL USING PREVIOUS DATA OF FIVE CHINESE CITIES DATA SET

TASK #1

Data preparation

Handling with the dataset.

G	H	I	J
PM_Dong	PM_Dong	PM_Nong	PM_US Po
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA

Beijing file had 4 PM values while other cities had only 3 incoming PM values

G	H	I	J
PM_Caota	PM_Shahe	Some_PM	PM_US Po
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA

So I decided to add one more column and fill it with NA values to have equal numbers of columns in each file to apply one algorithm to every file

Data preparation

```
import numpy as np
import pandas as p
import csv
#open all csv files one by one
files = ['./file_beijing.csv','./file_shenyang.csv','./file_guangzhou.csv',
'./file_shanghai.csv','./file_chengdu.csv']
#new csv file storage location
new_file = ['./new_file/file_beijing.csv','./new_file/file_shenyang.csv',
'./new_file/file_guangzhou.csv','./new_file/file_shanghai.csv','./new_file/file_chengdu.csv']
for cities in range(len(files)):
    f = open(new_file[cities], "w", newline='')
    csvwriter = csv.writer(f, delimiter=',')
    with open(files[cities], 'r') as myfile:
        column = csv.reader(myfile, delimiter=',')
        for col in list(myfile):
            column = col.split(",")
            #take the NULL values of 'PM_US Post' and fill with the average of neighbor values
            if (column[9] == 'NA'):
                us_post = 0
                diviser = 3
                for line in range(6,9):
                    if(column[line] != 'NA'):
                        us_post += (int(column[line]))
                    if(column[line] == 'NA'):
                        diviser = diviser - 1
                    if(diviser == 0):
                        diviser = 1
                        continue
                column[9] = str(int(us_post/diviser))
            csvwriter.writerow([column[1],column[2], column[3], column[4],column[5], column[9],
            column[10],column[11], column[12], column[13],column[14], column[15]])
```

This is the algorithm to calculate missing PM_US Post values at missing places from other PM stations. For example in case of

123	NA	138	NA
79	70	75	NA

NA$= int(\frac{123+138}{2})$ =130

NA = $int(\frac{79+70+75}{3})$ = 74

Numerical Data

Most of the data in columns were having format of ***int*** or ***float*** such as (year, month, day, PM_US Post, HUMI, PRES etc.)

Categorical Data

These types of data were converted to int format also, for exmaple: cbwd were converted into 1 or 0 values

Data Filling Principles

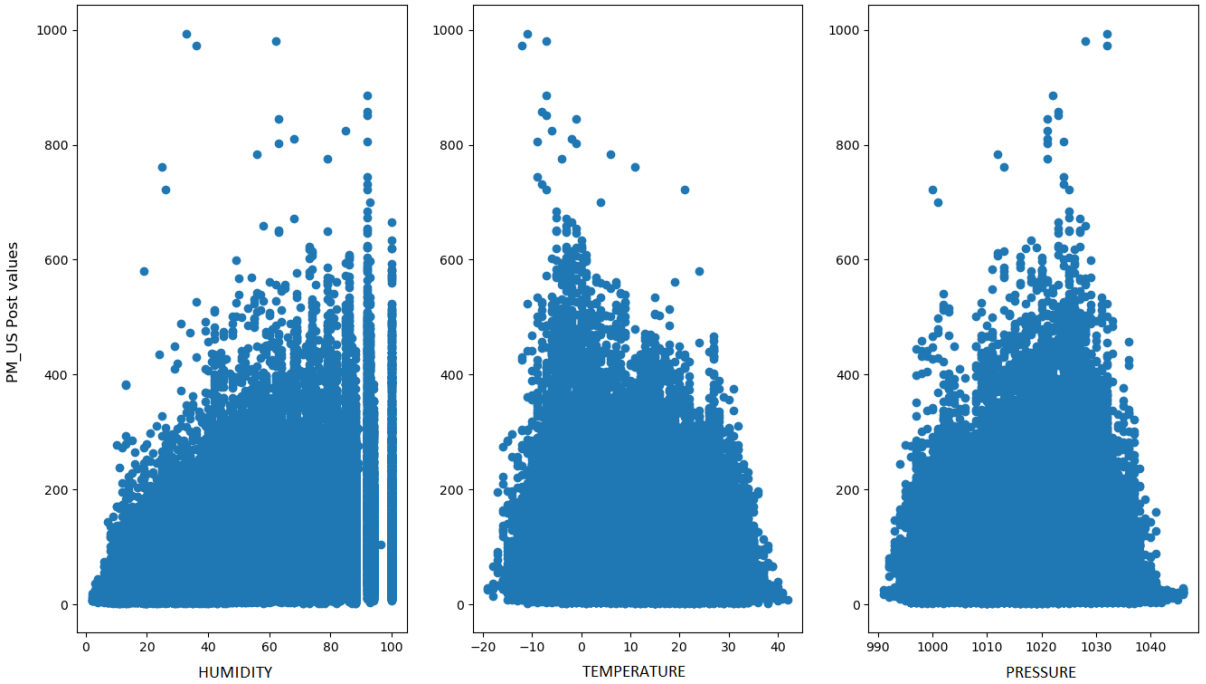
year	month	day	hour	season	PM_US Po	DEWP	HUMI	PRES	TEMP	cbwd	Iws
2010	1	1	0	4	0	-21	43	1021	-11	NW	1.79
2010	1	1	1	4	0	-21	47	1020	-12	NW	4.92
2010	1	1	2	4	0	-21	43	1019	-11	NW	6.71

The new created file was having this format because I have dropped other PM columns, first column and last two columns since there were containing meaningless information.

```
import numpy as np
import pandas as p
#dropping rows having NA values
new_file = ['./new_file/file_beijing.csv','./new_file/file_shenyang.csv',
'./new_file/file_guangzhou.csv','./new_file/file_shanghai.csv','./new_file/file_chengdu.csv']
for cities in new_file:
    new_f = p.read_csv(cities)
    #droping all rows having NA cells
    new_f = new_f.dropna(axis = 0, subset=['TEMP','season','Iws','PRES','DEWP',
    'HUMI', 'cbwd', 'month', 'day', 'hour', 'PM_US Post'])
    index_null = new_f[new_f['PM_US Post']==0].index
    new_f.drop(index_null, inplace=True)
    new_f.to_csv(cities, index=False)
    #store the result in separate files
```

Then I applied this algorithm to drop all NA containing rows.

Plotting the data




```
matplot.close('all')
a = data['HUMI']
b = data['TEMP']
c = data['PRES']
d = data['Iws']
f = data['PM_US Post']
matplot.figure(figsize=(16,9))
matplot.subplot(131)
matplot.scatter(a,f)
matplot.subplot(132)
matplot.scatter(b,f)
matplot.subplot(133)
matplot.scatter(d,f)
#matplot.subplot(221)
matplot.suptitle('PM_US Post values')

matplot.show()
```

This was the distribution of the data in terms of PM_US Post values to Humidity, Temperature and Pressure. As we can see there are some big values of US_PM Post which we can be considered as outliers

TRAINING AND TESTING

```python
import pandas as p
import numpy as np
import matplotlib.pyplot as matplot
import sklearn

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

from math import sqrt

final_csv = './dataset.csv'
new_file = [
'./new_file/file_beijing.csv',
'./new_file/file_shenyang.csv',
'./new_file/file_guangzhou.csv',
'./new_file/file_shanghai.csv',
'./new_file/file_chengdu.csv' ]
list_of_cities = []
for cities in new_file:
    data = p.read_csv(cities)
    list_of_cities.append(data)
data = p.concat(list_of_cities, sort=False)
#remove negative values and outliers from dataset (outliers)
for outliers, rows in data.iterrows():
    if (rows['HUMI'] < 0):
        data.drop(outliers, inplace=True)
    if (rows["PM_US Post"] > 800):
        data.drop(outliers, inplace=True)
data = p.concat([data, p.get_dummies(data['cbwd'], prefix='dir')], axis=1)
data.drop(['cbwd'], inplace=True, axis=1)
season = data.pop('season')
data['spring'] = (season==1) *  1
data['summer'] = (season==2) *  1
data['fall'] = (season==3) *  1
data['winter'] = (season==4) *  1
```

→ Required libraries

→ Import all files and store them at one data folder function

→ Removing outliers from dataset

→ Rearranging the dataset and increasing number of features

TRAINING AND TESTING

```
data.to_csv(final_csv)
# split our target column 'PM_US Post'
y = data['PM_US Post']
x_drop = data.drop(['PM_US Post'],axis =1)

#Starting train part
x_train, x_test, y_train, y_test = train_test_split(x_drop,
y, test_size=0.15, random_state=1234)

linerar_regression = LinearRegression()
linerar_regression.fit(x_train, y_train)

random_forest = RandomForestRegressor(n_estimators=200,
random_state=2000)
random_forest.fit(x_train, y_train)

#prediction x_test and y_test
linear_prediction = linerar_regression.predict(x_test)
linerar_regression_pred=linerar_regression.score(x_test, y_test)
print("Linear regression prediction: " , linerar_regression_pred)
mse_lr = sqrt(mean_squared_error(y_test, linear_prediction,))
r2_lr = r2_score(y_test, linear_prediction)
print("Mean squared error: ", mse_lr)
print("R2 score: ", r2_lr)
#random forest predictions
random_prediction = random_forest.predict(x_test)
random_forest_pred = random_forest.score(x_test, y_test)
print("Random Forest prediction: " , random_forest_pred)
print("Mean squared error: %.3f" %
sqrt(mean_squared_error(y_test, random_prediction)))
print("R2 score: %.3f" % r2_score(y_test, random_prediction))

matplot.title('Predicted vs Actual using Random Forest')
matplot.scatter(y_test, random_prediction)
matplot.xlabel("Actual values")
matplot.ylabel("Predicted values")
matplot.show()
```

→ Save the dataset and make PM_US Post and y of our function

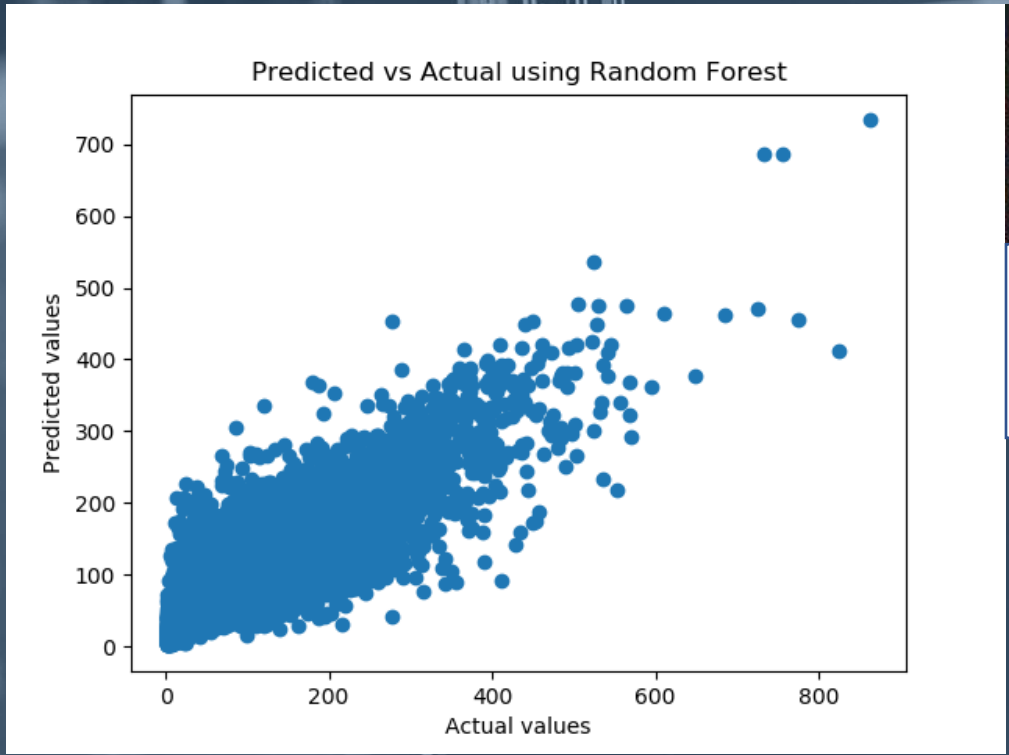
→ Train and test data separation and linear regression algorithm

→ Random forest algorithm

→ MSE and R2 score obtaining

→ Plotting the Random Forest results

RESULTS




```
(base) D:\Master's Course\Machine Learning\assignment\FiveCitiePMData>python train_regression.py
Linear regression prediction:  0.2098167548443245
Mean squared error:  60.33658736467998
R2 score:  0.2098167548443245
Random Forest prediction:  0.7702339794404618
Mean squared error: 32.536
R2 score: 0.770
```

The thing is it's not a linear problem and linear regression algorithm could hardly be applied and it's obvious the random forest algorithm is quite acceptable.