AIR QUALITY PREDICTION

importing the libraries

```
import pandas as pd
import seaborn as sns
%matplotlib inline
sklearn.preprocessing import StandardScaler # import normalization package
sklearn.model_selection import train_test_split # import train test split
# import linear regression package
sklearn.linear_model import LinearRegression
# import one hot encoder error and mean absolute error
sklearn.metrics import mean_squared_error, mean_absolute_error
# import random forest regressor
sklearn.ensemble import RandomForestRegressor
sklearn.svm import SVR # import support vector regressor
```

reading the dataset

It is a ';' seperated file so we'll specify it as a parameter for the read_csv function

```
df = pd.read_csv('
/home/husein/AUBAKO/DATASET/CO_FILES/Air%20Quality%2C.csv', sep=';')
df.head
```

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08
0	10/03/2004	18:00:00	2.6	1360.0	150.0	11.9	1046.0	166.0	1056.0	113.0	1692.0	
1	10/03/2004	19:00:00	2	1292.0	112.0	9.4	955.0	103.0	1174.0	92.0	1559.0	
2	10/03/2004	20:00:00	2.2	1402.0	88.0	9.0	939.0	131.0	1062.0	114.0	1549.0	
3	10/03/2004	21:00:00	2.2	1376.0	80.0	9.2	948.0	172.0	1092.0	123.0	1584.0	
4	10/03/2004	22:00:00	1.6	1272.0	51.0	6.5	836.0	131.0	1205.0	116.0	1490.0	

data preprocessing

The data contains null values. So we drop those rows and columns containing nulls.

```
df.drop(columns=['', axis=1, inplace=True)
df.dropna(how='all', axis=0, inplace=True)
```

The last few lines(specifically 9357 to 9471) of the dataset are empty and of no use. So we'll ignore them too

```
df.tail
```

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08.S5(O3)
9350	04/04/2005	10:00:00	3.1	1314.0	250.0	13.5	1101.0	472.0	539.0	190.0	1374.0	
9351	04/04/2005	11:00:00	2.4	1162.0	150.0	11.4	1027.0	353.0	604.0	179.0	1264.0	
9352	04/04/2005	12:00:00	2.4	1142.0	206.0	12.4	1063.0	293.0	603.0	175.0	1241.0	
9353	04/04/2005	13:00:00	2.1	1003.0	186.0	9.5	962.0	235.0	702.0	268.0	1041.0	
9354	04/04/2005	14:00:00	2.2	1071.0	167.0	11.3	1047.0	265.0	654.0	186.0	1129.0	

the values in our data don't contain decimal places but have weird commas in place of them. For example 9.4 is written as 9,4. We'll correct it using the following piece of code

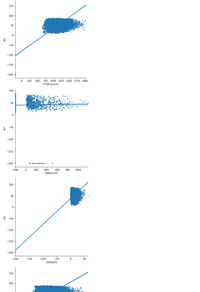
```
for col in cols:
    df[col].dtype == 'float64'
    str_s = pd.Series(df[col], str.replace(',', '.')
    float_s = []
    for value in str_s:values
        fv = float(value)
        float_s.append fv

    df[col] = pd.DataFrame(float_s

df.head
```

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08
0	10/03/2004	18:00:00	2.6	1360.0	150.0	11.9	1046.0	166.0	1056.0	113.0	1692.0	
1	10/03/2004	19:00:00	2.0	1292.0	112.0	9.4	955.0	103.0	1174.0	92.0	1559.0	
2	10/03/2004	20:00:00	2.2	1402.0	88.0	9.0	939.0	131.0	1062.0	114.0	1549.0	
3	10/03/2004	21:00:00	2.2	1376.0	80.0	9.2	948.0	172.0	1092.0	123.0	1584.0	
4	10/03/2004	22:00:00	1.6	1272.0	51.0	6.5	836.0	131.0	1205.0	116.0	1490.0	

data visualization

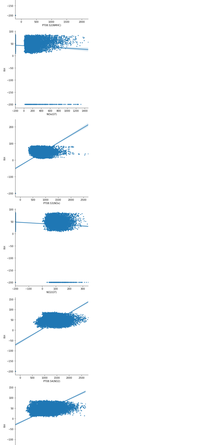
Understand co-relation between variables

```
plt.subplots(figsize=(
sns.heatmap df.corr(, annot=True, cmap='coolwarm')
plt.title 'Heatmap of co-relation between variables', fontsize=15)
plt.show
```



Try to understand degree of linearity between RH output and other input features

```
col_ = df.columns tolist
    y = col_
    sns.lmplot x=i, y='', data=df, markers='.',
```



model development

1. Linear Regression

Define Feature (X) and Target (y).

```
y = df.col_.drop('', 1) # drop the target
X = df.drop('', 1) # Features
```

Normalize Feature variable

```
sc = StandardScaler()
X_std = sc.fit_transform X # apply standardization
```

Train test split

split the data into train and test with test size and 20% and train size as 80

```
X_train X_test y_train y_test = train_test_split(
    X_std y test_size=0.2 random_state=
print 'training data',X_train.shape
print 'testing data',X_test.shape
```

Train the model

```
lr = LinearRegression()
lr_model = lr.fit X_train y_train # fit the linear model on train data
print 'intercept:', lr_model.intercept_
print 'coef:',
print 'score:', lr.score X_train
list zip X.columns tolist lr_model.coef_
```

```
training data (7485, 14)
testing data (1872, 14)
intercept: 0.9999999999999999
coef: [ 1.00000000e+00 -1.00000000e+00 -1.00000000e+00 -1.00000000e+00
 -1.00000000e+00 -1.00000000e+00 -1.00000000e+00 -1.00000000e+00
 -1.00000000e+00 -1.00000000e+00 -1.00000000e+00 -1.00000000e+00
 -1.00000000e+00 -1.00000000e+00]
score: 1.0
[('CO(GT)', 1.0000000000000000),
 ('PT08.S1(CO)', -1.0000000000000000),
 ('NMHC(GT)', -1.0000000000000000),
 ('C6H6(GT)', -1.0000000000000000),
 ('PT08.S2(NMHC)', -1.0000000000000000),
 ('NOx(GT)', -1.0000000000000000),
 ('PT08.S3(NOx)', -1.0000000000000000),
 ('NO2(GT)', -1.0000000000000000),
 ('PT08.S4(NO2)', -1.0000000000000000),
 ('PT08.S5(O3)', -1.0000000000000000),
 ('T', -1.0000000000000000),
 ('RH', -1.0000000000000000),
 ('AH', -1.0000000000000000)]
```

Root Mean Squared Error (RMSE) Value

```
y_pred = lr.model.predict X_test # predict using the model
rmse = np.sqrt mean_squared_error y_test, y_pred # calculate rmse
print 'RMSE:', rmse
```

```
RMSE: 0.0
```

model accuracy

```
print 'Mean Absolute Error:', metrics.r2_score y_test, y_pred # MAE
```

```
Mean Absolute Error: 1.0
```

Train the model

```
lr_rg = Ridge()
lr_model = lr.fit X_train, y_train # fit the model
print 'score:', lr.score X_train, y_train # fit model
y_pred_rf = lr_model.predict X_test # predict
```

Root Mean Squared Error (RMSE) Value

```
print 'RMSE:', mean_squared_error y_test, y_pred_rf np.sqrt
```

```
RMSE: 0.0
```

model accuracy

```
print 'Mean Absolute Error:', metrics.r2_score y_test, y_pred_rf # MAE
```

```
Mean Absolute Error: 1.0
```

3. Support Vector Machines (SVM)

```
X_train X_test y_train y_test = train_test_split
    X_std y test_size=0.2 random_state=
print 'training data',X_train.shape
print 'testing data',X_test.shape
```

Train the model

```
lr_rg = SVR
lr_model = lr.fit X_train, y_train # train the model
```

Root Mean Squared Error (RMSE) Value

```
y_pred_lr = lr_model.predict X_test # predict
print 'RMSE:', mean_squared_error y_test, y_pred_lr np.sqrt
```

```
RMSE: 0.0
```

model accuracy

```
print 'Mean Absolute Error:',
    metrics.r2_score y_test, y_pred_lr # MAE
```

```
Mean Absolute Error: 1.0
```

THE END