```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
%matplotlib inline
```

```
air_data = pd.read_excel('AirQualityUCI.xlsx')
air_data.head()
```

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(G
0	2004-03-10	18:00:00	2.6	1360.00	150	11.881723	1045.50	166
1	2004-03-10	19:00:00	2.0	1292.25	112	9.397165	954.75	103
2	2004-03-10	20:00:00	2.2	1402.00	88	8.997817	939.25	131
3	2004-03-10	21:00:00	2.2	1375.50	80	9.228796	948.25	172
4	2004-03-10	22:00:00	1.6	1272.25	51	6.518224	835.50	131

```
air_data.shape
```

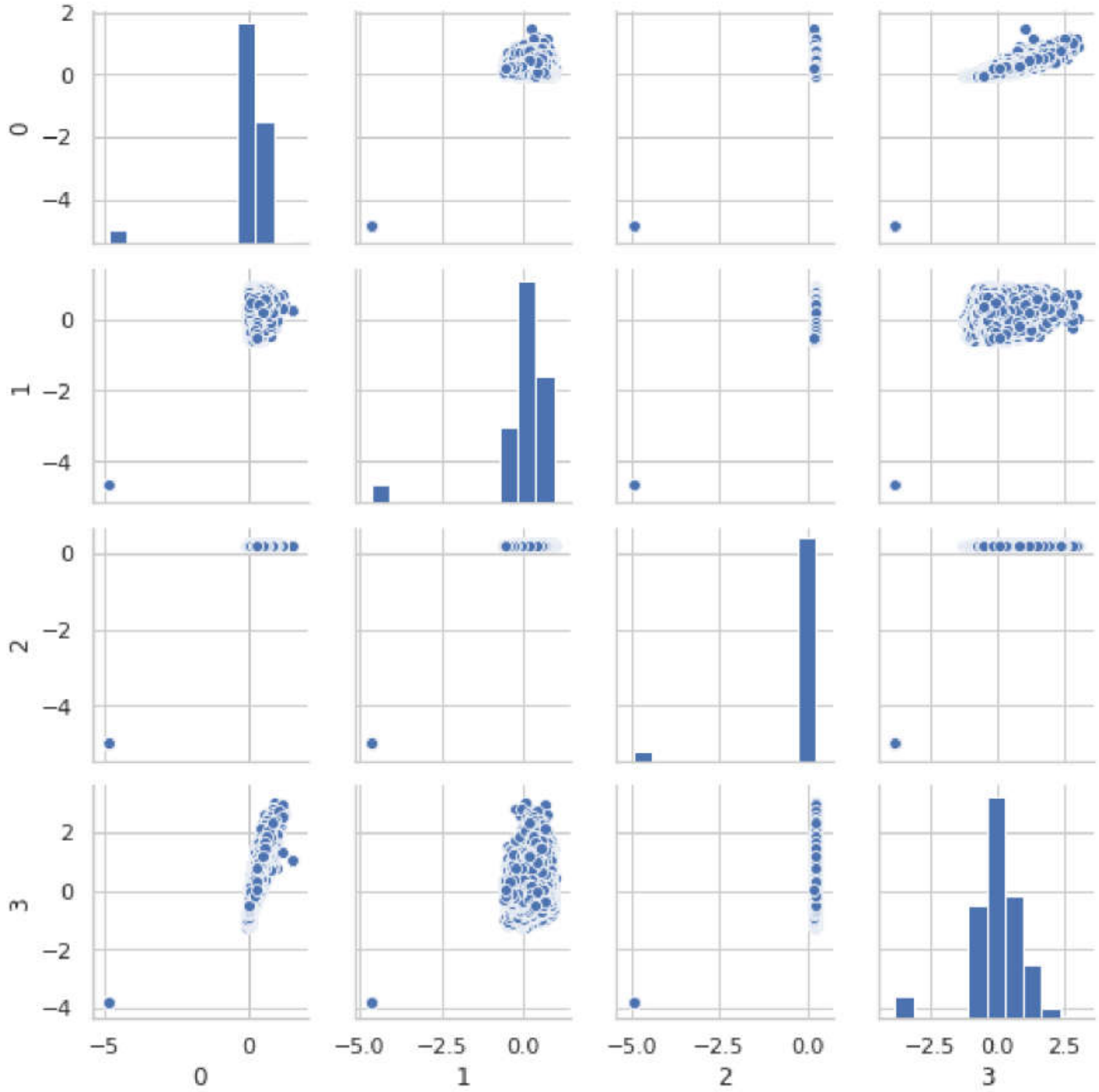
```
(9357, 15)
```

```
import seaborn as sns
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
sns.set(style='whitegrid', context='notebook')
features_plot = ['C6H6(GT)', 'RH', 'AH', 'PT08.S1(CO)']

data_to_plot = air_data[features_plot]
data_to_plot = scalar.fit_transform(data_to_plot)
data_to_plot = pd.DataFrame(data_to_plot)

sns.pairplot(data_to_plot, size=2.0);
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/axisgrid.py:2065: UserWarning: The `si
  warnings.warn(msg, UserWarning)
```



Step 1. Preprocessing data

```
air_data.dropna(axis=0, how='all')
```

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NO
0	2004-03-10	18:00:00	2.6	1360.00	150	11.881723	1045.50	
1	2004-03-10	19:00:00	2.0	1292.25	112	9.397165	954.75	
2	2004-03-10	20:00:00	2.2	1402.00	88	8.997817	939.25	
3	2004-03-10	21:00:00	2.2	1375.50	80	9.228796	948.25	
4	2004-03-10	22:00:00	1.6	1272.25	51	6.518224	835.50	
...	...	...	...	...	...	...	...	
9352	2005-04-04	10:00:00	3.1	1314.25	-200	13.529605	1101.25	
9353	2005-04-04	11:00:00	2.4	1162.50	-200	11.355157	1027.00	
9354	2005-04-04	12:00:00	2.4	1142.00	-200	12.374538	1062.50	
9355	2005-04-04	13:00:00	2.1	1002.50	-200	9.547187	960.50	
9356	2005-04-04	14:00:00	2.2	1070.75	-200	11.932060	1047.25	

Step 2. Features vs Labels

```
features = air_data

features = features.drop('Date', axis=1)
features = features.drop('Time', axis=1)
features = features.drop('C6H6(GT)', axis=1)
features = features.drop('PT08.S4(NO2)', axis=1)

labels = air_data['C6H6(GT)'].values

features = features.values
```

Step 3. Train and test portions

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.3)

print("X_trian shape --> {}".format(X_train.shape))
print("y_train shape --> {}".format(y_train.shape))
print("X_test shape --> {}".format(X_test.shape))
print("y_test shape --> {}".format(y_test.shape))
```

```
X_trian shape --> (6549, 11)
y_train shape --> (6549,)
X_test shape --> (2808, 11)
y_test shape --> (2808,)
```

Step 4. Regression

Step 4.1 Linear Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error

for i in range (25,35):
  X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.30,ran
  regressor = LinearRegression()
  regressor.fit(X_train, y_train)
  #regressor.predict(X_test)
  y_pred=regressor.predict(X_test)
  y_pred=y_pred.reshape((-1, 1))
  maxm=regressor.score(X_test, y_test)
  MSE=mean_squared_error(y_pred,y_test)
  MAE=mean_absolute_error(y_pred,y_test)

print(maxm)
print(MSE)
print(MAE)

from sklearn.metrics import mean_squared_error
from math import sqrt

rms = sqrt(mean_squared_error(y_pred, y_test))
rms
```

```
0.9992884871433486
1.1867347957963301
0.7909720964997654
1.0893735795384107
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt

rms = sqrt(mean_squared_error(y_pred,y_test))
rms
```

```
1.0893735795384107
```

Step 4.2 Support Vector Regression

```
from sklearn.model_selection import KFold
from sklearn.svm import SVR


support_regressor = SVR(kernel='rbf', C=1000)
support_regressor.fit(X_train, y_train)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/svm/base.py:193: FutureWarning: The de
  "avoid this warning.", FutureWarning)
SVR(C=1000, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
    gamma='auto_deprecated', kernel='rbf', max_iter=-1, shrinking=True,
    tol=0.001, verbose=False)
```

```
print("Coefficient of determination R^2 <-- on train set: {}".format(support_regressor.sco
```

```
Coefficient of determination R^2 <-- on train set: 0.9999942855092581
```

```
print("Coefficient of determination R^2 <-- on test set: {}".format(support_regressor.scor
```

```
Coefficient of determination R^2 <-- on test set: 0.27653837844191054
```

Step 4.3 Decision tree regression

```
from sklearn.tree import DecisionTreeRegressor


dtr = DecisionTreeRegressor()
dtr.fit(X_train, y_train)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

```
print("Coefficient of determination R^2 <-- on train set: {}".format(dtr.score(X_train, y_
```

```
Coefficient of determination R^2 <-- on train set: 1.0
```

```
print("Coefficient of determination R^2 <-- on test set: {}".format(dtr.score(X_test, y_te
```

```
Coefficient of determination R^2 <-- on test set: 0.9999975238435539
```

Step 4.4 Lasso regression

```
from sklearn.linear_model import Lasso


indiana_jones = Lasso(alpha=1.0)
indiana_jones.fit(X_train, y_train)
```

```
Lasso(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=1000,
      normalize=False, positive=False, precompute=False, random_state=None,
      selection='cyclic', tol=0.0001, warm_start=False)
```

```
print("Coefficient of determination R^2 <-- on train set : {}".format(indiana_jones.score(
```

```
Coefficient of determination R^2 <-- on train set : 0.999119306087109
```

```
print("Coefficient of determination R^2 <-- on test set: {}".format(indiana_jones.score(X_
```

```
Coefficient of determination R^2 <-- on test set: 0.9992290819896742
```

Step 5. Feature selection

```
from sklearn.ensemble import ExtraTreesRegressor
```

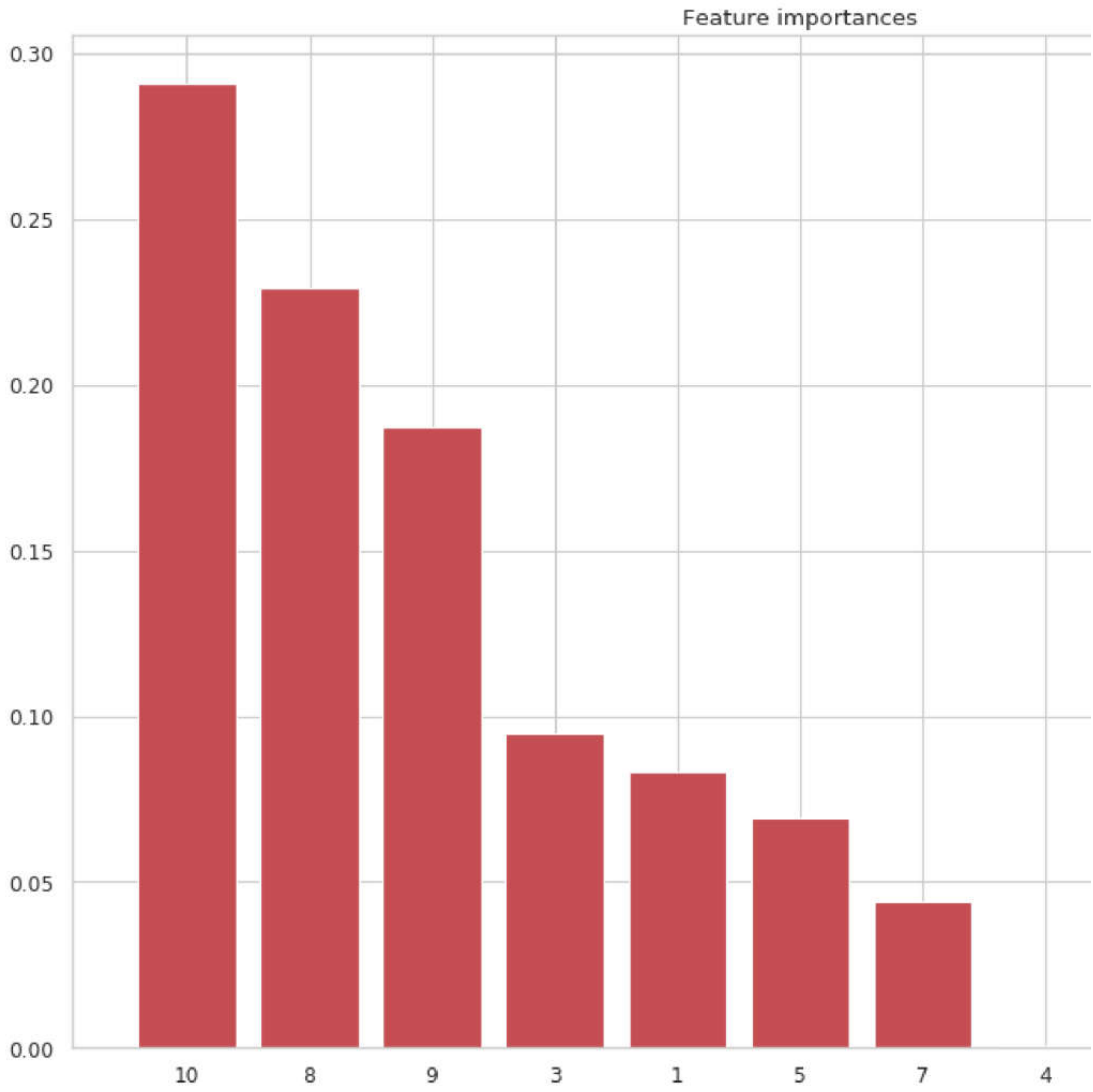
```
etr = ExtraTreesRegressor(n_estimators=300)
etr.fit(X_train, y_train)
```

```
ExtraTreesRegressor(bootstrap=False, criterion='mse', max_depth=None,
                    max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=300, n_jobs=None,
                    oob_score=False, random_state=None, verbose=0,
                    warm_start=False)
```

```
print(etr.feature_importances_)
indecis = np.argsort(etr.feature_importances_)[::-1]
```

```
[1.58384817e-04 8.28985786e-02 7.86497571e-06 9.49276808e-02
 5.10766906e-04 6.96420724e-02 2.02067310e-04 4.43440838e-02
 2.29323756e-01 1.87338358e-01 2.90646386e-01]
```

```
plt.figure(num=None, figsize=(14, 10), dpi=80, facecolor='w')
plt.title("Feature importances")
plt.bar(range(X_train.shape[1]), etr.feature_importances_[indecis],
       color="r", align="center")
plt.xticks(range(X_train.shape[1]), indecis)
plt.show()
```




```
plt.plot(y_pred, y_test)
plt.scatter(y_pred,y_test,c='red')
```

<matplotlib.collections.PathCollection at 0x7fb08578e0f0>