Importing libraries

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

Reading data

```
dataset = pd.read_csv('Powerplant_Data.csv')
# print(dataset.isnull().values.any()) - gave false, so no missing values
print(dataset.head())
x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

```
      AT      V       AP     RH      PE
0  14.96  41.76  1024.07  73.17  463.26
1  25.18  62.96  1020.04  59.08  444.37
2   5.11  39.40  1012.16  92.14  488.56
3  20.86  57.32  1010.24  76.64  446.48
4  10.82  37.50  1009.23  96.62  473.90
```

- Ambient Pressure (AP) in the range 992.89-1033.30 milibar,
- Relative Humidity (RH) in the range 25.56% to 100.16%
- Exhaust Vacuum (V) in teh range 25.36-81.56 cm Hg
- Net hourly electrical energy output (EP)

we are trying to predict EP based on the other features there is no categorical data so we don't need to do any encoding

Spliting the data set for training and testing

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

Multiple Regression

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(x_train, y_train)
```

```
     LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
              normalize=False)
```

Polynomial Regression

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
x_poly_train = poly_reg.fit_transform(x_train)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(x_poly_train, y_train)
```

```
     LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
              normalize=False)
```

SVR

```
from sklearn.svm import SVR
# have to standardize x_train and y_train
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
sc_y = StandardScaler()
x_train_std = sc_x.fit_transform(x_train)
# y has to be a matraix
y_train_std = y_train.reshape(len(y_train), 1)
y_train_std = sc_y.fit_transform(y_train_std)

svr = SVR(kernel='rbf')
svr.fit(x_train_std, y_train_std)
```

```
     C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:761: DataConversi
       y = column_or_1d(y, warn=True)
     SVR(C=1.0, 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)
```

Decision Tree

```
from sklearn.tree import DecisionTreeRegressor
tree_reg = DecisionTreeRegressor(random_state=0)
tree_reg = DecisionTreeRegressor(random_state=0)
tree_reg.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=0, splitter='best')
```

Random Forest

```
from sklearn.ensemble import RandomForestRegressor
forest_reg = RandomForestRegressor(n_estimators=10, random_state=0)
forest_reg.fit(x_train, y_train)
```

```
     RandomForestRegressor(bootstrap=True, 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=10, n_jobs=None,
                oob_score=False, random_state=0, verbose=0, warm_start=False)
```

Model evaluation

```
from sklearn import metrics
# multiple linear regression
lin_reg_score = metrics.r2_score(y_test, lin_reg.predict(x_test))
print("Multiple Linear regression score: ", lin_reg_score)

# random forest regression
forest_reg_score = metrics.r2_score(y_test, forest_reg.predict(x_test))
# from 10 to 500, it didn't improve much
#TODO: how do you set the number of instances
print("Random Forest regression score: ", forest_reg_score)

#decision regression
tree_reg_score = metrics.r2_score(y_test, tree_reg.predict(x_test))
print("Decision Tree regression score: ", tree_reg_score)

# random forest regression
x_poly_test = poly_reg.fit_transform(x_test)
poly_reg_score = metrics.r2_score(y_test, lin_reg_2.predict(x_poly_test))
print("Polynomial regression score: ", poly_reg_score)

# svr
y_pred_svr = sc_y.inverse_transform(svr.predict(sc_x.transform(x_test)))
svr_score = metrics.r2_score(y_test, y_pred_svr)
print("SVR score: ", svr_score)
print("SVR score: ", svr_score)

     Multiple Linear regression score:  0.9325315554761303
     Random Forest regression score:  0.9615980699813017
     Decision Tree regression score:  0.9226091050550043
     Polynomial regression score:  0.945819341122773
     SVR score:  0.9480784049986264
```