File Edit View Insert Cell Kernel Widgets Help Checkpoint created: 01:39:35 Trusted Python 3 ○

Run Markdown

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df = pd.read_csv("Data.csv")
df.head()
```

Out[2]:

	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

In [3]:
```
df.isna().sum()
```

Out[3]:
```
AT    0
V     0
AP    0
RH    0
PE    0
dtype: int64
```

Multiple Linear Regression

In [4]:
```
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

In [5]:
```
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)
```

In [6]:
```
from sklearn.linear_model import LinearRegression

lm = LinearRegression()
lm.fit(X_train, y_train)
lm_predictions = lm.predict(X_test)
```

In [7]:
```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

print(f"r2_score: {r2_score(y_test, lm_predictions)}")
print(f"mean_absolute_error: {mean_absolute_error(y_test, lm_predictions)}")
print(f"mean_squared_error: {mean_squared_error(y_test, lm_predictions)}")
print(f"Root Mean Square Error: {np.sqrt(mean_squared_error(y_test, lm_predictions))}")
```

```
r2_score: 0.9325315554761302
mean_absolute_error: 3.566564655203824
mean_squared_error: 19.73369930349765
Root Mean Square Error: 4.442262858442491
```

Polynomial Regression

In [8]:
```
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

In [9]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split

poly_feature = PolynomialFeatures(degree=4)
X = poly_feature.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [10]:
```
from sklearn.linear_model import LinearRegression

poly_lm = LinearRegression()
poly_lm.fit(X_train, y_train)
poly_predictions = poly_lm.predict(X_test)
```

In [11]:
```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

```
print(f"r2_score: {r2_score(y_test, poly_predictions)}")
print(f"mean_absolute_error: {mean_absolute_error(y_test, poly_predictions)}")
print(f"mean_squared_error: {mean_squared_error(y_test, poly_predictions)}")
print(f"Root Mean Square Error: {np.sqrt(mean_squared_error(y_test, poly_predictions))}")
```

```
r2_score: 0.9458193390165572
mean_absolute_error: 3.1360275695485944
mean_squared_error: 15.847184257134275
Root Mean Square Error: 3.9808522023725366
```

Support Vector Regression

1

In [12]:

```
X = df.iloc[:, :-1]
y = df.iloc[:, -1].to_frame()
```

In [13]:

```
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)
```

In [14]:

```
from sklearn.preprocessing import StandardScaler

sc_X = StandardScaler()
sc_y = StandardScaler()
X_train = sc_X.fit_transform(X_train)
y_train = sc_y.fit_transform(y_train)
X_test = sc_X.transform(X_test)
y_test = sc_y.transform(y_test)
```

Scale for X and y

In [15]:

```
from sklearn.svm import SVR

svr = SVR() # kernel = 'rbf'
svr.fit(X_train, y_train)
svr_predictions = svr.predict(X_test)
```

```
C:\Users\Perry\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning:
A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samp
les, ), for example using ravel().
  return f(*args, **kwargs)
```

In [16]:

```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

print(f"r2_score: {r2_score(y_test, svr_predictions)}")
print(f"mean_absolute_error: {mean_absolute_error(y_test, svr_predictions)}")
print(f"mean_squared_error: {mean_squared_error(y_test, svr_predictions)}")
print(f"Root Mean Square Error: {np.sqrt(mean_squared_error(y_test, svr_predictions))}")
```

```
r2_score: 0.9480784049986266
mean_absolute_error: 0.1756034994155729
mean_squared_error: 0.05220075426721192
Root Mean Square Error: 0.22847484383890476
```

2

In [17]:

```
X = df.iloc[:, :-1]
y = df.iloc[:, -1].to_frame()
```

In [18]:

```
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)
```

In [19]:

```
from sklearn.preprocessing import StandardScaler

sc_X = StandardScaler()
sc_y = StandardScaler()
X_train = sc_X.fit_transform(X_train)
y_train = sc_y.fit_transform(y_train)
```

Scale just only for X
(Right way, no need
scale for label)

In [20]:

```
from sklearn.svm import SVR

svr = SVR() # kernel = 'rbf'
svr.fit(X_train, y_train)
svr_predictions = svr.predict(sc_X.transform(X_test))
```

```
C:\Users\Perry\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning:
A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samp
les, ), for example using ravel().
  return f(*args, **kwargs)
```

In [21]:

```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

```
print(f"r2_score: {r2_score(y_test, sc_y.inverse_transform(svr_predictions))}")
print(f"mean_absolute_error: {mean_absolute_error(y_test, sc_y.inverse_transform(svr_predictions))}
print(f"mean_squared_error: {mean_squared_error(y_test, sc_y.inverse_transform(svr_predictions))}")
print(f"Root Mean Square Error: {np.sqrt(mean_squared_error(y_test, sc_y.inverse_transform(svr_pred
```

```
r2_score: 0.9480784049986266
mean_absolute_error: 2.9951783924512863
mean_squared_error: 15.186434937781799
Root Mean Square Error: 3.896977667087893
```

Decision Tree Regression

In [22]:

```
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

In [23]:

```
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)
```

In [24]:

```
from sklearn.tree import DecisionTreeRegressor

dtr = DecisionTreeRegressor(random_state=0)
dtr.fit(X_train, y_train)
dtr_predictions = dtr.predict(X_test)
```

In [25]:

```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

print(f"r2_score: {r2_score(y_test, dtr_predictions)}")
print(f"mean_absolute_error: {mean_absolute_error(y_test, dtr_predictions)}")
print(f"mean_squared_error: {mean_squared_error(y_test, dtr_predictions)}")
print(f"Root Mean Square Error: {np.sqrt(mean_squared_error(y_test, dtr_predictions))}")
```

```
r2_score: 0.922905874177941
mean_absolute_error: 3.103923719958203
mean_squared_error: 22.549093991640547
Root Mean Square Error: 4.748588631545224
```

Random Forest Regression

In [26]:

```
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

In [27]:

```
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)
```

In [28]:

```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(n_estimators=10, random_state=0)
rfr.fit(X_train, y_train)
rfr_predictions = rfr.predict(X_test)
```

In [29]:

```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

print(f"r2_score: {r2_score(y_test, rfr_predictions)}")
print(f"mean_absolute_error: {mean_absolute_error(y_test, rfr_predictions)}")
print(f"mean_squared_error: {mean_squared_error(y_test, rfr_predictions)}")
print(f"Root Mean Square Error: {np.sqrt(mean_squared_error(y_test, rfr_predictions))}")
```

```
r2_score: 0.9615908334363876
mean_absolute_error: 2.452366771159876
mean_squared_error: 11.234213991640557
Root Mean Square Error: 3.3517479009675766
```