Evaluating the Model Performance

October 15, 2024

1 Multiple Linear Regression

[11]:
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Data_1.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
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)

# Training the Multiple Linear Regression model on the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.
 ↪reshape(len(y_test),1)),1))

# Evaluating the Model Performance
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

```
[[431.43 431.23]
 [458.56 460.01]
 [462.75 461.14]
 …
 [469.52 473.26]
 [442.42 438.  ]
```

```
 [461.88 463.28]]
```

[11]:
```
0.9325315554761303
```

2 Polynomial Regression

[12]:
```
# Polynomial Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('Data_1.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
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)
# Training the Polynomial Regression model on the Training set

from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(X_train)
regressor = LinearRegression()
regressor.fit(X_poly, y_train)

# Predicting the Test set results
y_pred = regressor.predict(poly_reg.transform(X_test))
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.
 ↪reshape(len(y_test),1)),1))
# Evaluating the Model Performance
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

```
[[433.94 431.23]
 [457.9  460.01]
 [460.52 461.14]
 …
 [469.53 473.26]
 [438.27 438.  ]
 [461.67 463.28]]
```

[12]: 0.9458193146473887

3 Support Vector Regression (SVR)

[16]:
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('Data_1.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
y = y.reshape(len(y),1)


# Splitting the dataset into the Training set and Test set
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)
# Feature Scaling
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)

# Training the SVR model on the Training set
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = sc_y.inverse_transform(regressor.predict(sc_X.transform(X_test)).
 ↪reshape(-1,1))
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.
 ↪reshape(len(y_test),1)),1))

# Evaluating the Model Performance
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

```
C:\Users\ddaya\devi\anaconda3\Lib\site-
packages\sklearn\utils\validation.py:1143: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
[[434.05 431.23]
 [457.94 460.01]
 [461.03 461.14]
 …
 [470.6  473.26]
 [439.42 438.  ]
 [460.92 463.28]]
```

[16]:
```
0.948078404998626
```

4 Decision Tree Regression

[17]:
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Data_1.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
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)

# Training the Decision Tree Regression model on the Training set
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X_train, y_train)


# Predicting the Test set results
y_pred = regressor.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.
 ↪reshape(len(y_test),1)),1))

# Evaluating the Model Performance
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

```
[[431.28 431.23]
 [459.59 460.01]
 [460.06 461.14]
 …
```

```
 [471.46 473.26]
 [437.76 438.  ]
 [462.74 463.28]]
```

[17]:
```
0.922905874177941
```

5 Random Forest Regression

[18]:
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Data_1.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
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)

# Training the Random Forest Regression model on the whole dataset
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.
 ↪reshape(len(y_test),1)),1))


# Evaluating the Model Performance
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

```
[[434.05 431.23]
 [458.79 460.01]
 [463.02 461.14]
 …
 [469.48 473.26]
 [439.57 438.  ]
 [460.38 463.28]]
```

[18]: 0.9615908334363876

[ ]: