RegularizedLinearModels

August 19, 2022

[1]:
```
!pip install mglearn
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: mglearn in ./.local/lib/python3.7/site-packages
(0.1.9)
Requirement already satisfied: imageio in /usr/local/lib/python3.7/site-packages
(from mglearn) (2.8.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/site-
packages (from mglearn) (0.24.2)
Requirement already satisfied: cycler in /usr/local/lib/python3.7/site-packages
(from mglearn) (0.10.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/site-packages
(from mglearn) (0.14.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/site-packages
(from mglearn) (1.21.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/site-
packages (from mglearn) (3.5.1)
Requirement already satisfied: pillow in /usr/local/lib/python3.7/site-packages
(from mglearn) (7.1.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/site-packages
(from mglearn) (1.1.5)
Requirement already satisfied: six in /usr/local/lib/python3.7/site-packages
(from cycler->mglearn) (1.14.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.7/site-packages (from matplotlib->mglearn) (1.2.0)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.7/site-packages (from matplotlib->mglearn) (2.8.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.7/site-packages (from matplotlib->mglearn) (4.28.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/site-
packages (from matplotlib->mglearn) (21.0)
Requirement already satisfied: pyparsing>=2.2.1 in
/usr/local/lib/python3.7/site-packages (from matplotlib->mglearn) (2.4.6)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/site-
packages (from pandas->mglearn) (2019.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.7/site-packages (from scikit-learn->mglearn) (2.2.0)
Requirement already satisfied: scipy>=0.19.1 in /usr/local/lib/python3.7/site-
```

```
packages (from scikit-learn->mglearn) (1.4.1)
WARNING: You are using pip version 22.0.3; however, version 22.0.4 is
available.
You should consider upgrading via the '/usr/local/bin/python3 -m pip install
--upgrade pip' command.
```

[2]:
```
import mglearn
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

[3]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

[4]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

[5]:
```
Add_df=pd.read_csv('Advertising.csv')
Add_df
```

[5]:

	Unnamed: 0	TV	radio	newspaper	sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9
..	…	…	…	…	…
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

[200 rows x 5 columns]

[6]:
```
#X,y = mglearn.datasets.load_extended_boston()
X=Add_df[['TV','radio','newspaper']]
y=Add_df[['sales']]
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=0)
```

Linear Regression

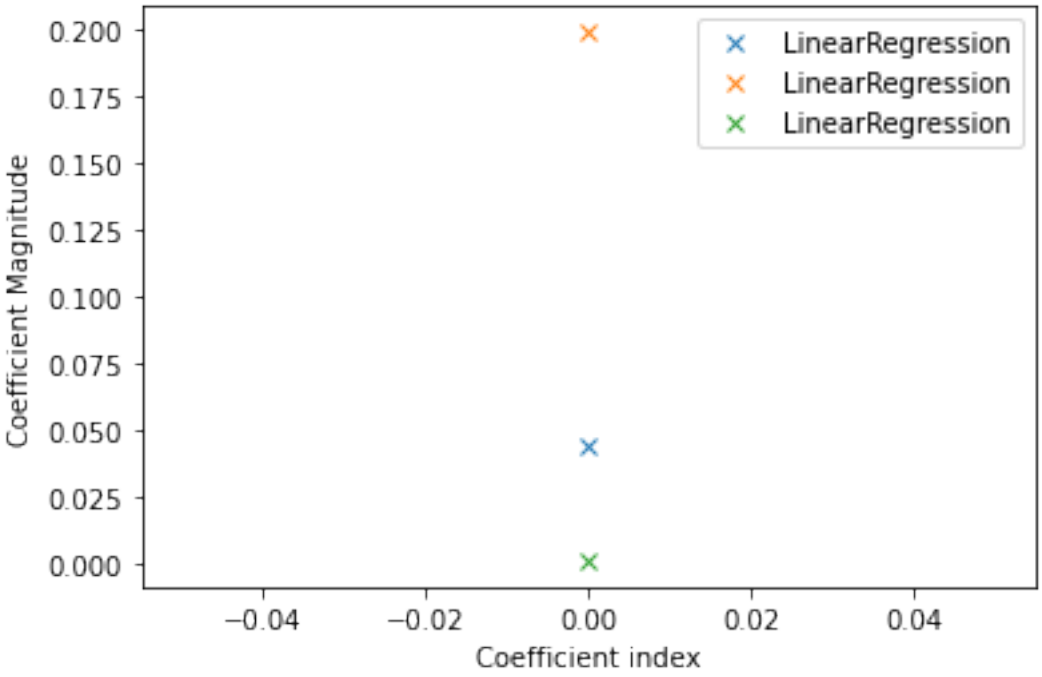
[7]:
```
my_linear_reg = LinearRegression().fit(X_train, y_train)
```

[8]:
```
my_preds_train = my_linear_reg.predict(X_train)
my_preds_test = my_linear_reg.predict(X_test)

print('TRAIN DATA RESULTS')
print("Mean Absolute Error on Train data : ", mean_absolute_error(y_train,␣
 ↪my_preds_train))
print("Mean Squared Error on Train data : ", mean_squared_error(y_train,␣
 ↪my_preds_train))
print('Training Set R squared score : ', my_linear_reg.score(X_train, y_train))

print('\nTEST DATA RESULTS')
print("Mean Absolute Error on Test data : ", mean_absolute_error(y_test,␣
 ↪my_preds_test))
print("Mean Squared Error on Test data : ", mean_squared_error(y_test,␣
 ↪my_preds_test))
print('Test Set R squared score : ', my_linear_reg.score(X_test, y_test))
```

```
TRAIN DATA RESULTS
Mean Absolute Error on Train data :  1.2248611099616429
Mean Squared Error on Train data :  2.438917493390848
Training Set R squared score :  0.9072183330817297

TEST DATA RESULTS
Mean Absolute Error on Test data :  1.3000320919235455
Mean Squared Error on Test data :  4.012497522917099
Test Set R squared score :  0.8576396745320893
```

[9]:
```
plt.plot(my_linear_reg.coef_,'x', label = 'LinearRegression')
plt.xlabel('Coefficient index')
plt.ylabel('Coefficient Magnitude')
plt.legend()
```

[9]:
```
<matplotlib.legend.Legend at 0x7f53e21d15d0>
```



Ridge Regression

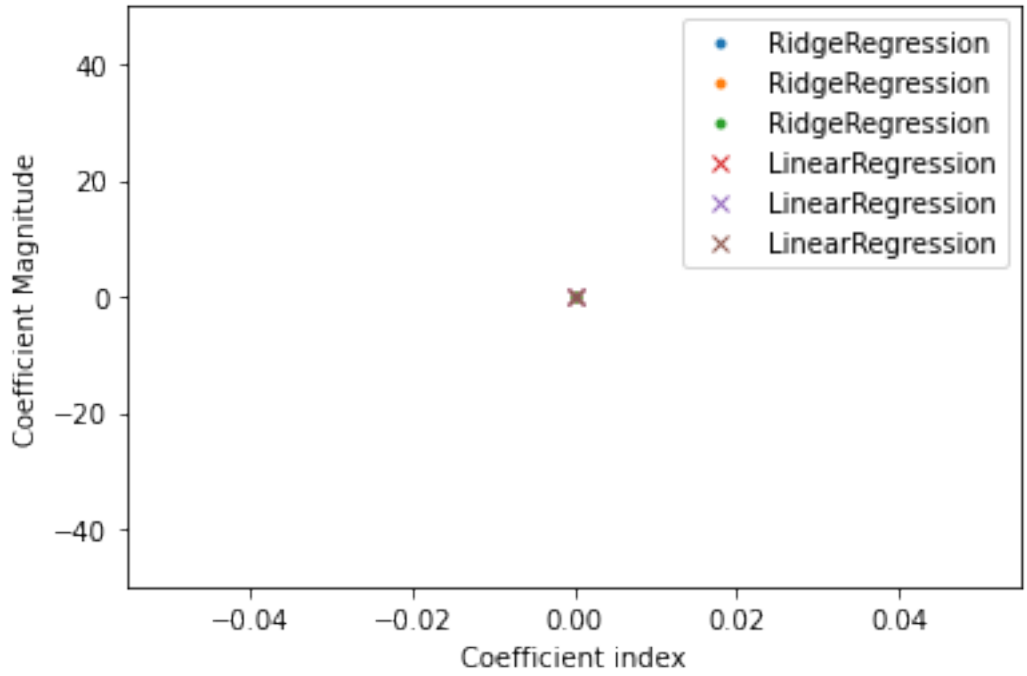
[10]:
```
from sklearn.linear_model import Ridge

my_ridge_model = Ridge().fit(X_train, y_train)

my_preds_train = my_ridge_model.predict(X_train)
my_preds_test = my_ridge_model.predict(X_test)

print('TRAIN DATA RESULTS')
print("Mean Absolute Error on Train data : ", mean_absolute_error(y_train,␣
 ↪my_preds_train))
print("Mean Squared Error on Train data : ", mean_squared_error(y_train,␣
 ↪my_preds_train))
print('Training Set R squared score : ', my_ridge_model.score(X_train, y_train))

print('\nTEST DATA RESULTS')
print("Mean Absolute Error on Test data : ", mean_absolute_error(y_test,␣
 ↪my_preds_test))
print("Mean Squared Error on Test data : ", mean_squared_error(y_test,␣
 ↪my_preds_test))
print('Test Set R squared score : ', my_ridge_model.score(X_test, y_test))
```

```
TRAIN DATA RESULTS
```

```
Mean Absolute Error on Train data :  1.2248673911523371
Mean Squared Error on Train data :  2.4389175021913676
Training Set R squared score :  0.9072183327469391

TEST DATA RESULTS
Mean Absolute Error on Test data :  1.3000186595256988
Mean Squared Error on Test data :  4.012394239227089
Test Set R squared score :  0.8576433389579591
```

[11]:
```
plt.plot(my_ridge_model.coef_, '.', label ='RidgeRegression')
plt.plot(my_linear_reg.coef_,'x', label ='LinearRegression')
plt.xlabel('Coefficient index')
plt.ylabel('Coefficient Magnitude')
plt.ylim(-50,50)
plt.legend()
```

[11]:
```
<matplotlib.legend.Legend at 0x7f53e21abbd0>
```



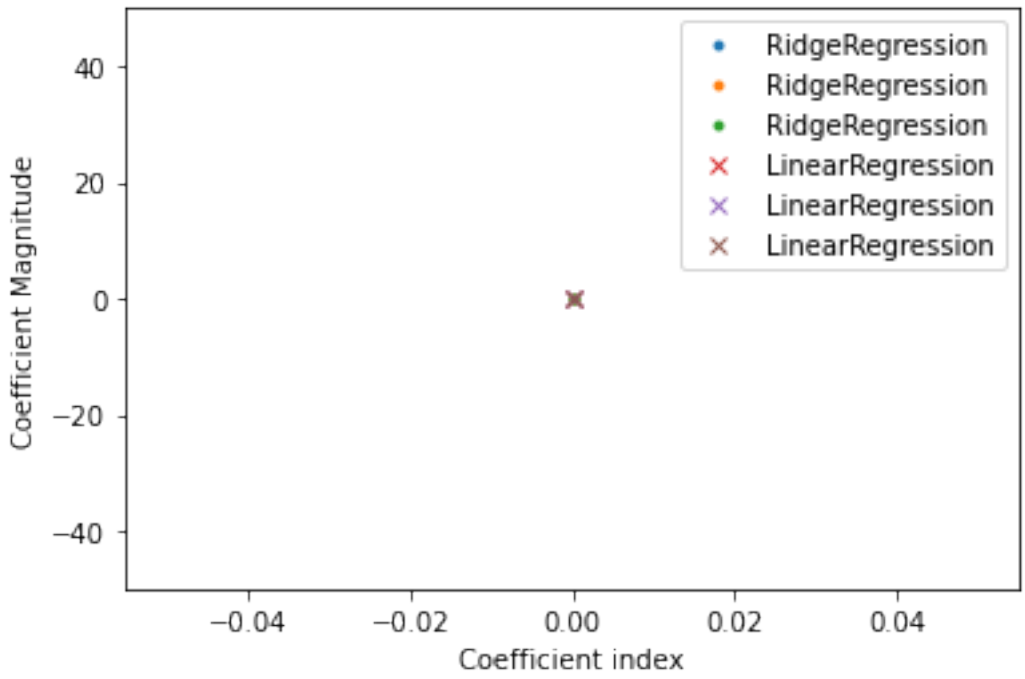
[12]:
```
my_ridge_model = Ridge(alpha=10).fit(X_train, y_train)

my_preds_train = my_ridge_model.predict(X_train)
my_preds_test = my_ridge_model.predict(X_test)

print('TRAIN DATA RESULTS')
```

```
print("Mean Absolute Error on Train data : ", mean_absolute_error(y_train,␣
 ↪my_preds_train))
print("Mean Squared Error on Train data : ", mean_squared_error(y_train,␣
 ↪my_preds_train))
print('Training Set R squared score : ', my_ridge_model.score(X_train, y_train))

print('\nTEST DATA RESULTS')
print("Mean Absolute Error on Test data : ", mean_absolute_error(y_test,␣
 ↪my_preds_test))
print("Mean Squared Error on Test data : ", mean_squared_error(y_test,␣
 ↪my_preds_test))
print('Test Set R squared score : ', my_ridge_model.score(X_test, y_test))
```

```
TRAIN DATA RESULTS
Mean Absolute Error on Train data :  1.2249239026391954
Mean Squared Error on Train data :  2.4389183728899853
Training Set R squared score :  0.907218299623691

TEST DATA RESULTS
Mean Absolute Error on Test data :  1.2998977975461328
Mean Squared Error on Test data :  4.011465658712957
Test Set R squared score :  0.8576762842802826
```

[13]:
```
plt.plot(my_ridge_model.coef_, '.', label ='RidgeRegression')
plt.plot(my_linear_reg.coef_,'x', label ='LinearRegression')
plt.xlabel('Coefficient index')
plt.ylabel('Coefficient Magnitude')
plt.ylim(-50,50)
plt.legend()
```

[13]: <matplotlib.legend.Legend at 0x7f53ddf79e50>



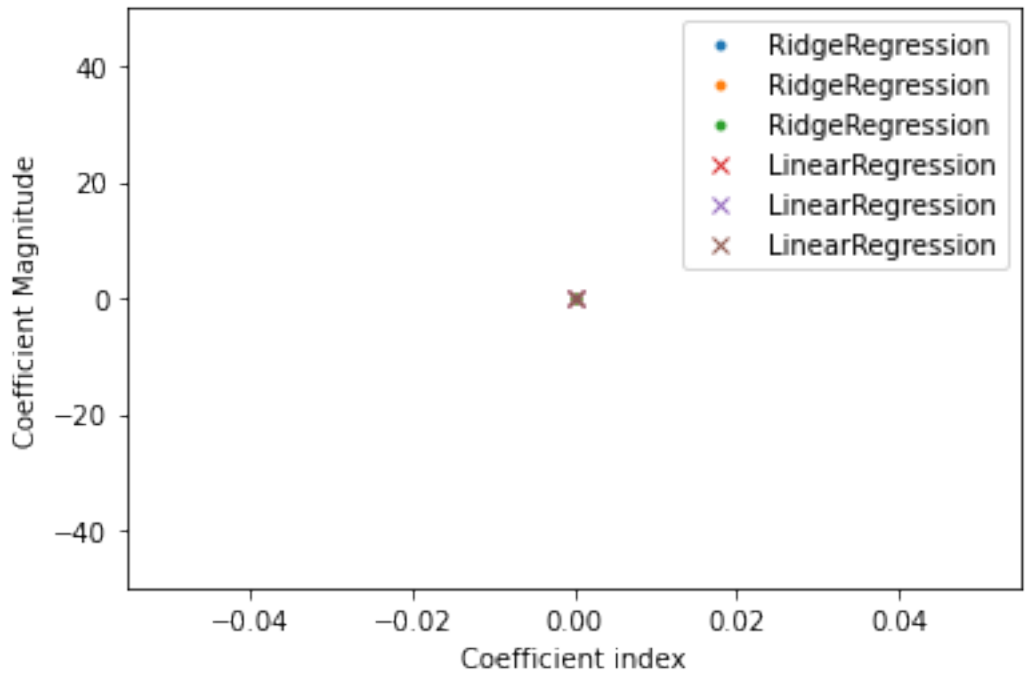
[14]:
```
from sklearn.linear_model import Ridge

my_ridge_model = Ridge(alpha=0.3).fit(X_train, y_train)

my_preds_train = my_ridge_model.predict(X_train)
my_preds_test = my_ridge_model.predict(X_test)

print('TRAIN DATA RESULTS')
print("Mean Absolute Error on Train data : ", mean_absolute_error(y_train,␣
 ↪my_preds_train))
print("Mean Squared Error on Train data : ", mean_squared_error(y_train,␣
 ↪my_preds_train))
print('Training Set R squared score : ', my_ridge_model.score(X_train, y_train))

print('\nTEST DATA RESULTS')
print("Mean Absolute Error on Test data : ", mean_absolute_error(y_test,␣
 ↪my_preds_test))
print("Mean Squared Error on Test data : ", mean_squared_error(y_test,␣
 ↪my_preds_test))
print('Test Set R squared score : ', my_ridge_model.score(X_test, y_test))
```

```
TRAIN DATA RESULTS
Mean Absolute Error on Train data :  1.224862994363735
Mean Squared Error on Train data :  2.438917494182934
Training Set R squared score :  0.907218333051597
```

```
TEST DATA RESULTS
Mean Absolute Error on Test data :  1.3000280621351021
Mean Squared Error on Test data :  4.012466535539211
Test Set R squared score :  0.8576407739404195
```

[15]:
```
plt.plot(my_ridge_model.coef_, '.', label ='RidgeRegression')
plt.plot(my_linear_reg.coef_,'x', label ='LinearRegression')
plt.xlabel('Coefficient index')
plt.ylabel('Coefficient Magnitude')
plt.ylim(-50,50)
plt.legend()
```

[15]:
```
<matplotlib.legend.Legend at 0x7f53dde69910>
```



Lasso Regression

[16]:
```
from sklearn.linear_model import Lasso
```

[17]:
```
my_lasso = Lasso().fit(X_train, y_train)

my_preds_train = my_lasso.predict(X_train)
my_preds_test = my_lasso.predict(X_test)

print('TRAIN DATA RESULTS')
```

```
print("Mean Absolute Error on Train data : ", mean_absolute_error(y_train,␣
 ↪my_preds_train))
print("Mean Squared Error on Train data : ", mean_squared_error(y_train,␣
 ↪my_preds_train))
print('Training Set R squared score : ', my_lasso.score(X_train, y_train))

print('\nTEST DATA RESULTS')
print("Mean Absolute Error on Test data : ", mean_absolute_error(y_test,␣
 ↪my_preds_test))
print("Mean Squared Error on Test data : ", mean_squared_error(y_test,␣
 ↪my_preds_test))
print('Test Set R squared score : ', my_lasso.score(X_test, y_test))
```

```
TRAIN DATA RESULTS
Mean Absolute Error on Train data :  1.2311116369303716
Mean Squared Error on Train data :  2.444052917547911
Training Set R squared score :  0.9070229705018492

TEST DATA RESULTS
Mean Absolute Error on Test data :  1.2794792729550337
Mean Squared Error on Test data :  3.9254309985586335
Test Set R squared score :  0.860728727839721
```

[18]:
```
np.sum(my_lasso.coef_!=0)/104
```

[18]:
```
0.019230769230769232
```

[19]:
```
my_lasso = Lasso(alpha=.01).fit(X_train, y_train)

my_preds_train = my_lasso.predict(X_train)
my_preds_test = my_lasso.predict(X_test)

print('TRAIN DATA RESULTS')
print("Mean Absolute Error on Train data : ", mean_absolute_error(y_train,␣
 ↪my_preds_train))
print("Mean Squared Error on Train data : ", mean_squared_error(y_train,␣
 ↪my_preds_train))
print('Training Set R squared score : ', my_lasso.score(X_train, y_train))

print('\nTEST DATA RESULTS')
print("Mean Absolute Error on Test data : ", mean_absolute_error(y_test,␣
 ↪my_preds_test))
print("Mean Squared Error on Test data : ", mean_squared_error(y_test,␣
 ↪my_preds_test))
print('Test Set R squared score : ', my_lasso.score(X_test, y_test))
```

```
TRAIN DATA RESULTS
```

```
Mean Absolute Error on Train data :  1.224907791561
Mean Squared Error on Train data :  2.4389180381307614
Training Set R squared score :  0.9072183123586519

TEST DATA RESULTS
Mean Absolute Error on Test data :  1.2998106396429687
Mean Squared Error on Test data :  4.0115293632680356
Test Set R squared score :  0.8576740240916737
```

[20]:
```
my_lasso = Lasso(alpha=.01, max_iter=100000).fit(X_train, y_train)

my_preds_train = my_lasso.predict(X_train)
my_preds_test = my_lasso.predict(X_test)

print('TRAIN DATA RESULTS')
print("Mean Absolute Error on Train data : ", mean_absolute_error(y_train,␣
 ↪my_preds_train))
print("Mean Squared Error on Train data : ", mean_squared_error(y_train,␣
 ↪my_preds_train))
print('Training Set R squared score : ', my_lasso.score(X_train, y_train))

print('\nTEST DATA RESULTS')
print("Mean Absolute Error on Test data : ", mean_absolute_error(y_test,␣
 ↪my_preds_test))
print("Mean Squared Error on Test data : ", mean_squared_error(y_test,␣
 ↪my_preds_test))
print('Test Set R squared score : ', my_lasso.score(X_test, y_test))
```

```
TRAIN DATA RESULTS
Mean Absolute Error on Train data :  1.224907791561
Mean Squared Error on Train data :  2.4389180381307614
Training Set R squared score :  0.9072183123586519

TEST DATA RESULTS
Mean Absolute Error on Test data :  1.2998106396429687
Mean Squared Error on Test data :  4.0115293632680356
Test Set R squared score :  0.8576740240916737
```

[21]:
```
np.sum(my_lasso.coef_!=0)
```

[21]: 3

[22]:
```
my_lasso = Lasso(alpha=.0001, max_iter=100000).fit(X_train, y_train)

my_preds_train = my_lasso.predict(X_train)
my_preds_test = my_lasso.predict(X_test)
```

```
print('TRAIN DATA RESULTS')
print("Mean Absolute Error on Train data : ", mean_absolute_error(y_train,␣
 ↪my_preds_train))
print("Mean Squared Error on Train data : ", mean_squared_error(y_train,␣
 ↪my_preds_train))
print('Training Set R squared score : ', my_lasso.score(X_train, y_train))

print('\nTEST DATA RESULTS')
print("Mean Absolute Error on Test data : ", mean_absolute_error(y_test,␣
 ↪my_preds_test))
print("Mean Squared Error on Test data : ", mean_squared_error(y_test,␣
 ↪my_preds_test))
print('Test Set R squared score : ', my_lasso.score(X_test, y_test))
```

```
TRAIN DATA RESULTS
Mean Absolute Error on Train data :  1.2248615353698362
Mean Squared Error on Train data :  2.4389174934455125
Training Set R squared score :  0.9072183330796502

TEST DATA RESULTS
Mean Absolute Error on Test data :  1.3000298748964576
Mean Squared Error on Test data :  4.0124878110135
Test Set R squared score :  0.8576400191029581
```

[23]:
```
from sklearn.linear_model import ElasticNet
```

[24]:
```
my_ElasticNet = ElasticNet().fit(X_train, y_train)

my_preds_train = my_ElasticNet.predict(X_train)
my_preds_test = my_ElasticNet.predict(X_test)

print('TRAIN DATA RESULTS')
print("Mean Absolute Error on Train data : ", mean_absolute_error(y_train,␣
 ↪my_preds_train))
print("Mean Squared Error on Train data : ", mean_squared_error(y_train,␣
 ↪my_preds_train))
print('Training Set R squared score : ', my_ElasticNet.score(X_train, y_train))

print('\nTEST DATA RESULTS')
print("Mean Absolute Error on Test data : ", mean_absolute_error(y_test,␣
 ↪my_preds_test))
print("Mean Squared Error on Test data : ", mean_squared_error(y_test,␣
 ↪my_preds_test))
print('Test Set R squared score : ', my_ElasticNet.score(X_test, y_test))
```

```
TRAIN DATA RESULTS
Mean Absolute Error on Train data :  1.2284859098975165
```

```
Mean Squared Error on Train data :  2.4407282660433864
Training Set R squared score :  0.9071494473955317

TEST DATA RESULTS
Mean Absolute Error on Test data :  1.2879326693152395
Mean Squared Error on Test data :  3.9582815093844466
Test Set R squared score :  0.8595632169861342
```

[25]:
```
from sklearn.model_selection import GridSearchCV
```

[26]:
```
params = {'alpha':[0.001, 0.01, 0.1, 0.2, 0.5, 0.8, 1], 'max_iter':[1500,␣
 ↪10000, 100000]}
grid_search = GridSearchCV(my_ElasticNet, params, cv = 3)
grid_search.fit(X_train, y_train)
```

[26]:
```
GridSearchCV(cv=3, estimator=ElasticNet(),
             param_grid={'alpha': [0.001, 0.01, 0.1, 0.2, 0.5, 0.8, 1],
                         'max_iter': [1500, 10000, 100000]})
```

[27]:
```
grid_search.best_params_
```

[27]:
```
{'alpha': 1, 'max_iter': 1500}
```

[28]:
```
my_preds_train = grid_search.predict(X_train)
my_preds_test = grid_search.predict(X_test)

print('TRAIN DATA RESULTS')
print("Mean Absolute Error on Train data : ", mean_absolute_error(y_train,␣
 ↪my_preds_train))
print("Mean Squared Error on Train data : ", mean_squared_error(y_train,␣
 ↪my_preds_train))
print('Training Set R squared score : ', grid_search.score(X_train, y_train))

print('\nTEST DATA RESULTS')
print("Mean Absolute Error on Test data : ", mean_absolute_error(y_test,␣
 ↪my_preds_test))
print("Mean Squared Error on Test data : ", mean_squared_error(y_test,␣
 ↪my_preds_test))
print('Test Set R squared score : ', grid_search.score(X_test, y_test))
```

```
TRAIN DATA RESULTS
Mean Absolute Error on Train data :  1.2284859098975165
Mean Squared Error on Train data :  2.4407282660433864
Training Set R squared score :  0.9071494473955317

TEST DATA RESULTS
Mean Absolute Error on Test data :  1.2879326693152395
```

```
Mean Squared Error on Test data :  3.9582815093844466
Test Set R squared score :  0.8595632169861342
```

[ ]: