nonlinear_regression

July 22, 2020

1 Nonlinear regression exercises

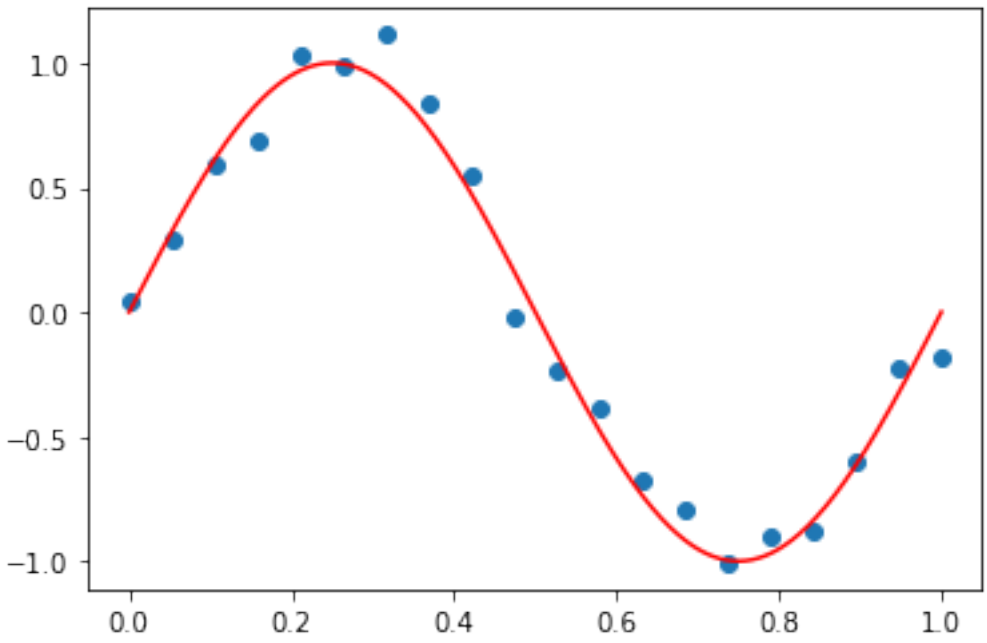
1.1 Linear regression review.

[1]:
```
# Import the necessary libraries.
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# Create data.
data_size = 20
X = np.linspace(0, 1, data_size)
# Include noise.
noise = np.random.uniform(low=-1.0, high=1.0, size=data_size) * 0.2
y = np.sin(2.0 * np.pi * X) + noise
```

[3]:
```
X_line = np.linspace(0, 1, 1000)
sin_X = np.sin(2.0 * np.pi * X_line)
```

[4]:
```
# Drawing functions for training data and correct answer data.
def plot_sin():
  plt.scatter(X, y)
  plt.plot(X_line, sin_X, 'red')
plot_sin()
```



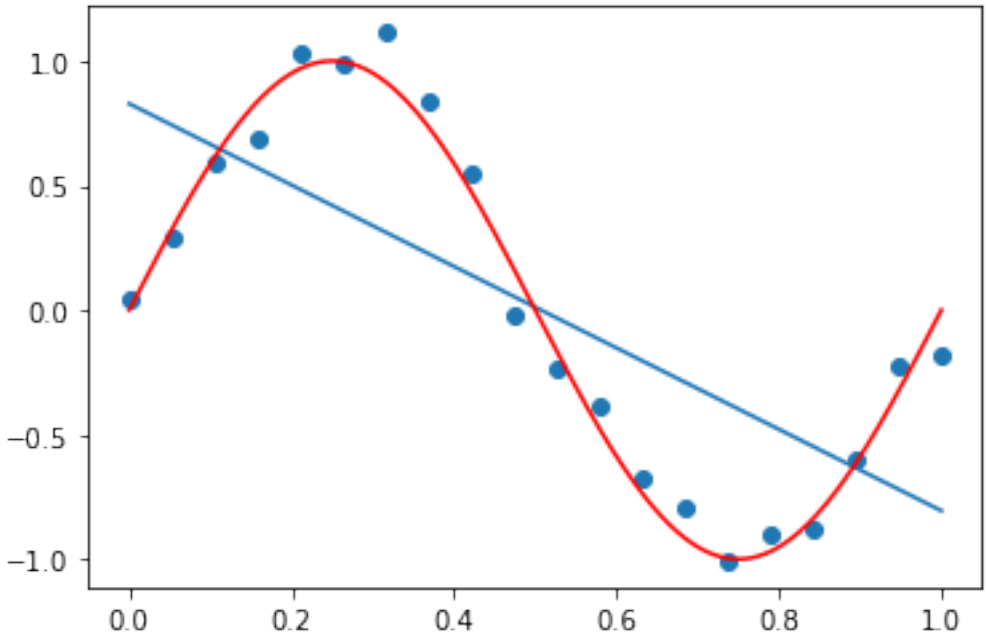
[5]:
```
from sklearn.linear_model import LinearRegression
```

[6]:
```
# Predicted by linear regression.
lin_reg = LinearRegression().fit(X.reshape(-1, 1), y)
```

[7]:
```
# Check the regression variables and intercepts.
lin_reg.intercept_, lin_reg.coef_
```

[7]:
```
(0.8284329606722655, array([-1.63413944]))
```

[8]:
```
# Draw the model.
plt.plot(X_line, lin_reg.intercept_ + lin_reg.coef_ * X_line)
plot_sin()
```



1.2 polynomial regression

[9]:
```
# Predict the model by the square.
X_2 = X ** 2
```

[10]:
```
X_new = np.concatenate([X.reshape(-1, 1), X_2.reshape(-1, 1)], axis=1)
```

[11]:
```
X_2
```

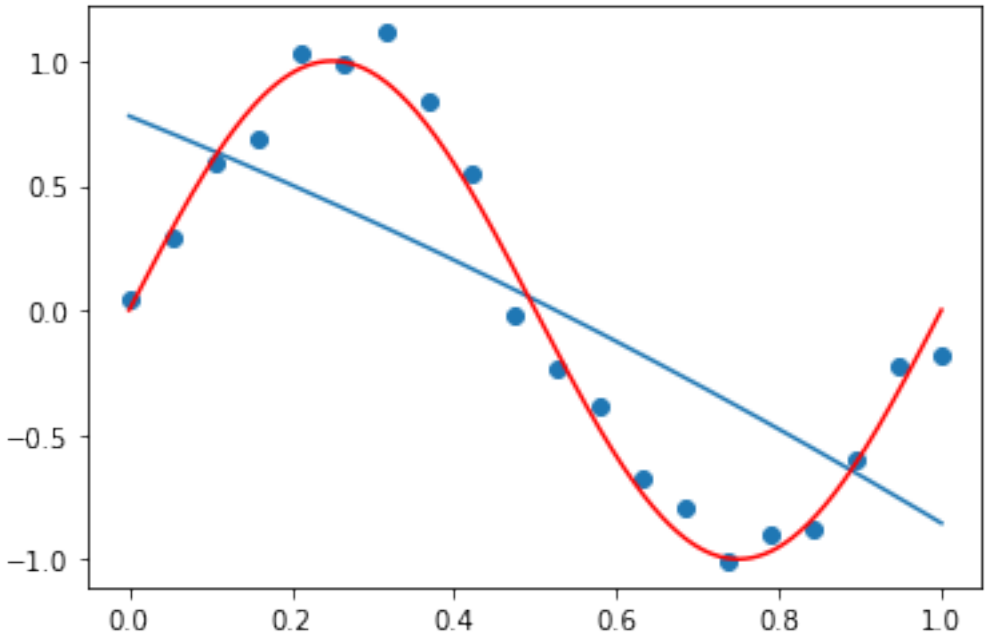
[11]:
```
array([0.        , 0.00277008, 0.01108033, 0.02493075, 0.04432133,
       0.06925208, 0.09972299, 0.13573407, 0.17728532, 0.22437673,
       0.27700831, 0.33518006, 0.39889197, 0.46814404, 0.54293629,
       0.6232687 , 0.70914127, 0.80055402, 0.89750693, 1.        ])
```

[12]:
```
lin_reg_2 = LinearRegression().fit(X_new, y)
```

[13]:
```
lin_reg_2.intercept_, lin_reg_2.coef_
```

[13]:
```
(0.7786933756057179, array([-1.31912206, -0.31501737]))
```

[14]:
```
plt.plot(X_line, lin_reg_2.intercept_ + lin_reg_2.coef_[0] * X_line + lin_reg_2.
 ↪coef_[1] * X_line ** 2)
plot_sin()
```



[15]:
```
from sklearn.preprocessing import PolynomialFeatures
```

[16]:
```
# Polynomial regression.
poly = PolynomialFeatures(degree=3)
poly.fit(X.reshape(-1, 1))
X_poly_3 = poly.transform(X.reshape(-1, 1))
```

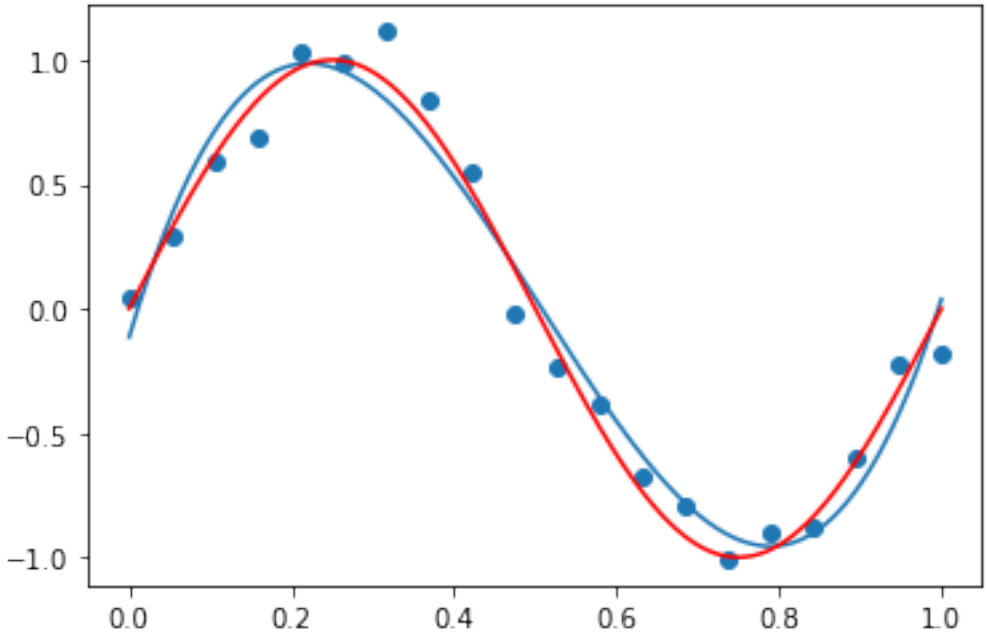
[17]:
```
X_poly_3
```

[17]:
```
array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.00000000e+00, 5.26315789e-02, 2.77008310e-03, 1.45793847e-04],
       [1.00000000e+00, 1.05263158e-01, 1.10803324e-02, 1.16635078e-03],
       [1.00000000e+00, 1.57894737e-01, 2.49307479e-02, 3.93643388e-03],
       [1.00000000e+00, 2.10526316e-01, 4.43213296e-02, 9.33080624e-03],
       [1.00000000e+00, 2.63157895e-01, 6.92520776e-02, 1.82242309e-02],
       [1.00000000e+00, 3.15789474e-01, 9.97229917e-02, 3.14914711e-02],
       [1.00000000e+00, 3.68421053e-01, 1.35734072e-01, 5.00072897e-02],
       [1.00000000e+00, 4.21052632e-01, 1.77285319e-01, 7.46464499e-02],
       [1.00000000e+00, 4.73684211e-01, 2.24376731e-01, 1.06283715e-01],
       [1.00000000e+00, 5.26315789e-01, 2.77008310e-01, 1.45793847e-01],
       [1.00000000e+00, 5.78947368e-01, 3.35180055e-01, 1.94051611e-01],
       [1.00000000e+00, 6.31578947e-01, 3.98891967e-01, 2.51931768e-01],
       [1.00000000e+00, 6.84210526e-01, 4.68144044e-01, 3.20309083e-01],
       [1.00000000e+00, 7.36842105e-01, 5.42936288e-01, 4.00058318e-01],
       [1.00000000e+00, 7.89473684e-01, 6.23268698e-01, 4.92054235e-01],
       [1.00000000e+00, 8.42105263e-01, 7.09141274e-01, 5.97171599e-01],
```

```
       [1.00000000e+00, 8.94736842e-01, 8.00554017e-01, 7.16285173e-01],
       [1.00000000e+00, 9.47368421e-01, 8.97506925e-01, 8.50269719e-01],
       [1.00000000e+00, 1.00000000e+00, 1.00000000e+00, 1.00000000e+00]])
```

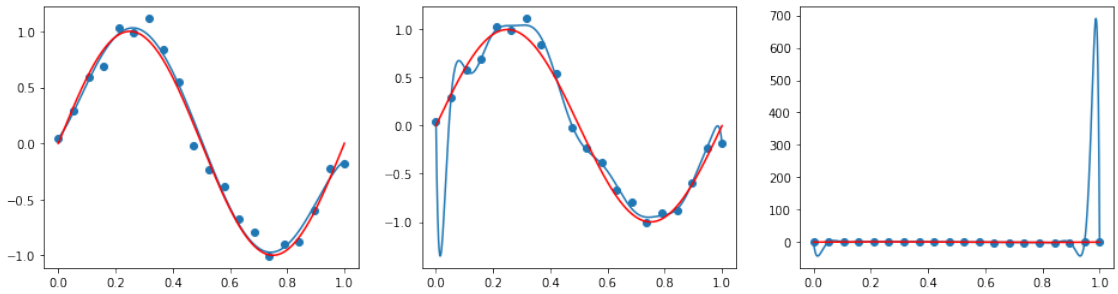
[18]:
```
lin_reg_3 = LinearRegression().fit(X_poly_3, y)
```

[19]:
```
X_line_poly_3 = poly.fit_transform(X_line.reshape(-1, 1))
plt.plot(X_line, lin_reg_3.predict(X_line_poly_3))
plot_sin()
```



[20]:
```
# Check overfitting.
fig, axes = plt.subplots(1, 3, figsize=(16, 4))

for degree, ax in zip([5, 15, 25], axes):
  poly = PolynomialFeatures(degree=degree)
  X_poly = poly.fit_transform(X.reshape(-1, 1))
  lin_reg = LinearRegression().fit(X_poly, y)
  X_line_poly = poly.fit_transform(X_line.reshape(-1, 1))
  ax.plot(X_line, lin_reg.predict(X_line_poly))
  ax.scatter(X, y)
  ax.plot(X_line, sin_X, 'red')
```



1.3 Ridge and Lasso regressions

[21]:
```
!pip install -q mglearn
import mglearn
import pandas as pd
from sklearn.model_selection import  train_test_split
```

[22]:
```
X, y = mglearn.datasets.load_extended_boston()
```

[23]:
```
X
```

[23]:
```
array([[0.00000000e+00, 1.80000000e-01, 6.78152493e-02, …,
        1.00000000e+00, 8.96799117e-02, 8.04248656e-03],
       [2.35922539e-04, 0.00000000e+00, 2.42302053e-01, …,
        1.00000000e+00, 2.04470199e-01, 4.18080621e-02],
       [2.35697744e-04, 0.00000000e+00, 2.42302053e-01, …,
        9.79579831e-01, 6.28144504e-02, 4.02790570e-03],
       …,
       [6.11892474e-04, 0.00000000e+00, 4.20454545e-01, …,
        1.00000000e+00, 1.07891832e-01, 1.16406475e-02],
       [1.16072990e-03, 0.00000000e+00, 4.20454545e-01, …,
        9.82676920e-01, 1.29930407e-01, 1.71795127e-02],
       [4.61841693e-04, 0.00000000e+00, 4.20454545e-01, …,
        1.00000000e+00, 1.69701987e-01, 2.87987643e-02]])
```

[24]:
```
y
```

[24]:
```
array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
       18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
       15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2,
       13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 24.7,
       21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 18.9,
       35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 23.5,
       19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20. ,
       20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 22.2,
```

```
       23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 43.8,
       33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 19.4,
       21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22. ,
       20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 19.6,
       23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4, 13.4,
       15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3, 19.4,
       17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. , 22.7,
       25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6, 29.4,
       23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6, 50. ,
       32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3, 30.3,
       34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5, 24.4,
       20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5, 23. ,
       26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5, 24.3,
       31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. , 20.1,
       22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8, 29.6,
       42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8, 31. ,
       36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2, 32.4,
       32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2, 22. ,
       20.1, 23.2, 22.3, 24.8, 28.5, 37.3, 27.9, 23.9, 21.7, 28.6, 27.1,
       20.3, 22.5, 29. , 24.8, 22. , 26.4, 33.1, 36.1, 28.4, 33.4, 28.2,
       22.8, 20.3, 16.1, 22.1, 19.4, 21.6, 23.8, 16.2, 17.8, 19.8, 23.1,
       21. , 23.8, 23.1, 20.4, 18.5, 25. , 24.6, 23. , 22.2, 19.3, 22.6,
       19.8, 17.1, 19.4, 22.2, 20.7, 21.1, 19.5, 18.5, 20.6, 19. , 18.7,
       32.7, 16.5, 23.9, 31.2, 17.5, 17.2, 23.1, 24.5, 26.6, 22.9, 24.1,
       18.6, 30.1, 18.2, 20.6, 17.8, 21.7, 22.7, 22.6, 25. , 19.9, 20.8,
       16.8, 21.9, 27.5, 21.9, 23.1, 50. , 50. , 50. , 50. , 50. , 13.8,
       13.8, 15. , 13.9, 13.3, 13.1, 10.2, 10.4, 10.9, 11.3, 12.3,  8.8,
        7.2, 10.5,  7.4, 10.2, 11.5, 15.1, 23.2,  9.7, 13.8, 12.7, 13.1,
       12.5,  8.5,  5. ,  6.3,  5.6,  7.2, 12.1,  8.3,  8.5,  5. , 11.9,
       27.9, 17.2, 27.5, 15. , 17.2, 17.9, 16.3,  7. ,  7.2,  7.5, 10.4,
        8.8,  8.4, 16.7, 14.2, 20.8, 13.4, 11.7,  8.3, 10.2, 10.9, 11. ,
        9.5, 14.5, 14.1, 16.1, 14.3, 11.7, 13.4,  9.6,  8.7,  8.4, 12.8,
       10.5, 17.1, 18.4, 15.4, 10.8, 11.8, 14.9, 12.6, 14.1, 13. , 13.4,
       15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4, 17.7,
       19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6, 23.2,
       29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. , 21.8,
       20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8, 24.5,
       23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. , 11.9])
```

[25]:
```
df_X = pd.DataFrame(X)
df_y = pd.DataFrame(y)
```

[26]:
```
df_X
```

[26]:

	0	1	2	3	…	100	101	102	103
0	0.000000	0.18	0.067815	0.0	…	0.025759	1.000000	0.089680	0.008042
1	0.000236	0.00	0.242302	0.0	…	0.113111	1.000000	0.204470	0.041808

```
2    0.000236  0.00  0.242302  0.0  …  0.035109  0.979580  0.062814  0.004028
3    0.000293  0.00  0.063050  0.0  …  0.021667  0.988585  0.033197  0.001115
4    0.000705  0.00  0.063050  0.0  …  0.064464  1.000000  0.099338  0.009868
..        …    …       …    …   …       …         …         …         …
501  0.000633  0.00  0.420455  0.0  …  0.195787  0.975392  0.216382  0.048003
502  0.000438  0.00  0.420455  0.0  …  0.181239  1.000000  0.202815  0.041134
503  0.000612  0.00  0.420455  0.0  …  0.096414  1.000000  0.107892  0.011641
504  0.001161  0.00  0.420455  0.0  …  0.117127  0.982677  0.129930  0.017180
505  0.000462  0.00  0.420455  0.0  …  0.151649  1.000000  0.169702  0.028799

[506 rows x 104 columns]
```

[27]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

[28]:
```
lin_reg = LinearRegression().fit(X_train, y_train)
```

[29]:
```
def print_score(model):
  print(round(model.score(X_train, y_train), 3))
  print(round(model.score(X_test, y_test), 3))
```

[30]:
```
print_score(lin_reg)
```

```
0.952
0.607
```

[31]:
```
from sklearn.linear_model import Ridge, Lasso
```

1.4 Ridge regressions

[32]:
```
ridge = Ridge().fit(X_train, y_train)
```

[33]:
```
print_score(ridge)
```

```
0.886
0.753
```

[34]:
```
ridge_10 = Ridge(alpha=10).fit(X_train, y_train)
print_score(ridge_10)
```

```
0.788
0.636
```

[35]:
```
ridge_01 = Ridge(alpha=0.1).fit(X_train, y_train)
print_score(ridge_01)
```

```
0.928
0.772
```

[36]:
```
coefficients = pd.DataFrame({'lin_reg': lin_reg.coef_, 'ridge':ridge.coef_,␣
↪'ridge_10':ridge_10.coef_, 'ridge_01':ridge_01.coef_})
```

[37]:
```
coefficients
```

[37]:

	lin_reg	ridge	ridge_10	ridge_01
0	-412.710947	-1.413684	-0.811610	-2.333007
1	-52.243207	-1.556619	0.647609	-5.441929
2	-131.898815	-1.465434	-0.809783	-6.128655
3	-12.004136	-0.126616	0.311335	-0.778525
4	-15.510713	-0.079119	-0.684624	1.291530
..	…	…	…	…
99	35.361714	-2.361124	-1.769550	-0.559662
100	11.955343	0.043467	-2.036371	4.947479
101	0.677026	1.208860	1.079849	2.667765
102	2.734520	-6.325992	-3.952641	-6.907626
103	30.372001	10.360023	0.142449	24.258596

[104 rows x 4 columns]

1.5 Lasso regressions

[38]:
```
lasso = Lasso().fit(X_train, y_train)
```

[39]:
```
print_score(lasso)
```

```
0.293
0.209
```

[40]:
```
lasso_001 = Lasso(alpha=0.01, max_iter=10000).fit(X_train, y_train)
print_score(lasso_001)
```

```
0.896
0.766
```

[41]:
```
coefficients_lasso = pd.DataFrame({'lin_reg':lin_reg.coef_, 'lasso':lasso.
↪coef_, 'lasso_001':lasso_001.coef_})
```

[42]:
```
coefficients_lasso
```

[42]:

	lin_reg	lasso	lasso_001
0	-412.710947	-0.0	-0.000000
1	-52.243207	0.0	-0.000000
2	-131.898815	-0.0	-0.000000
3	-12.004136	0.0	0.000000
4	-15.510713	-0.0	-0.000000
..	…	…	…

```
99    35.361714   -0.0   0.000000
100   11.955343   -0.0  -0.000000
101    0.677026    0.0   0.344041
102    2.734520   -0.0  -8.246456
103   30.372001   -0.0  17.560672

[104 rows x 3 columns]
```