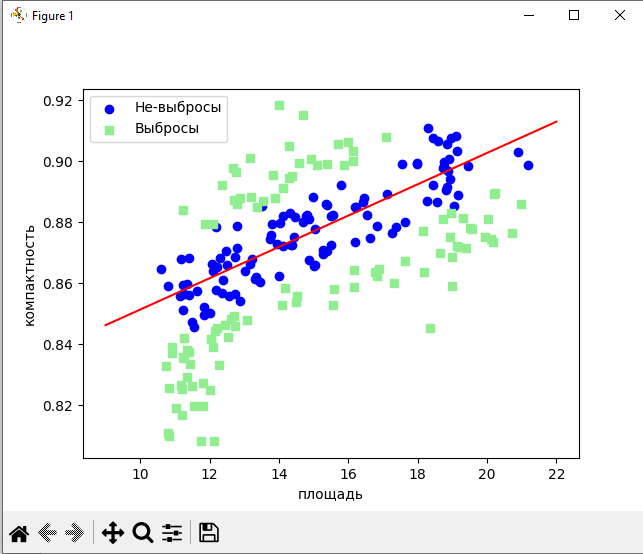
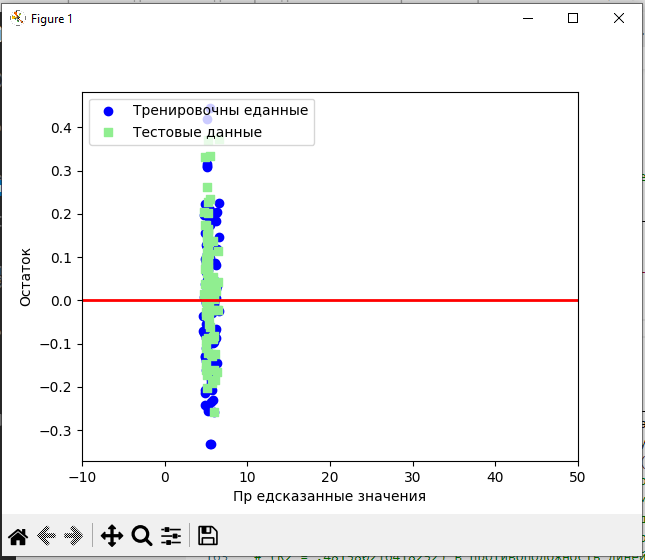


Как видно на графике выше, с уменьшением площади, увеличивается компактность.

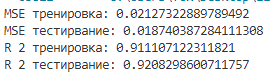


На рисунке выше показан метод RANSAC который выполняет подгонку регрессионной модели на подмножестве данных, так называемых не выбросах (inliers), т. е. хороших точках данных.



На рисунке выше видим график остатков с линией, проходящей через начало отсчета оси X.

В случае идеального предсказания остатки были бы строго нулями, с чем в реальных и практических приложениях мы, вероятно, никогда не столкнемся.

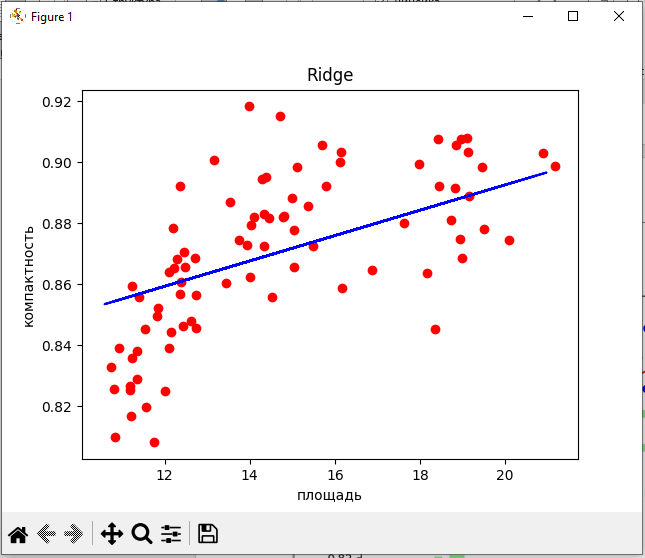


Мы увидим, что MSE на тренировочном наборе равна 0.02127322889789492, а MSE тестового набора намного больше со значением 0.018740387284111308,

что указывает на то, что наша модель переподогнана под тренировочные данные.

После оценивания на тренировочных данных коэффициент детерминации R2 нашей модели составляет 0.911107122311821, что является неплохим результатом.

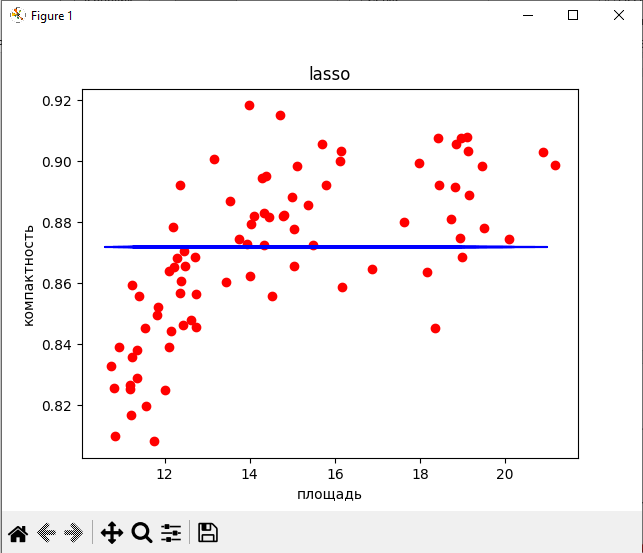
Однако R2 на тестовом наборе данных составил всего 0.9208298600711757

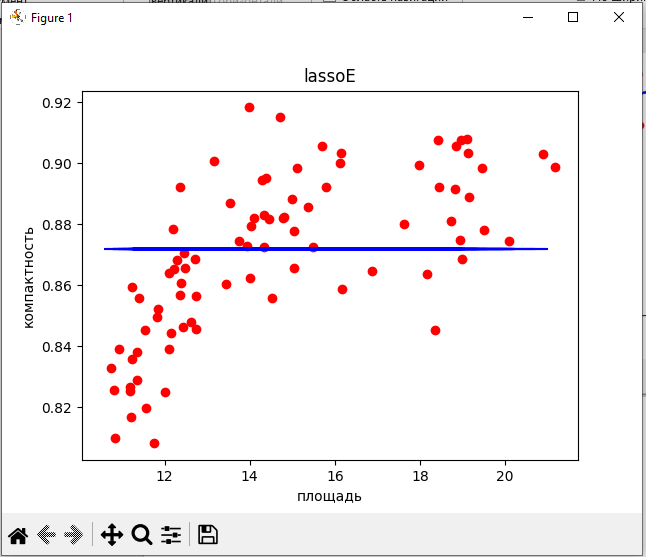


регуляризация - это один из подходов к решению проблемы переобучения путем добавления дополнительной информации и тем самым сжатия значений параметров модели, чтобы вызвать штраф за сложность. Самыми популярными подходами к регуляризованной линейной регрессии являются так называемый метод гребневой регрессии (гidge гegгession), метод lasso (оператор наименьшего абсолютного стягивания и отбора, least absolute shгinkage and selection орегаtог, lasso) и метод эластичной сети ( elastic пеt) 1

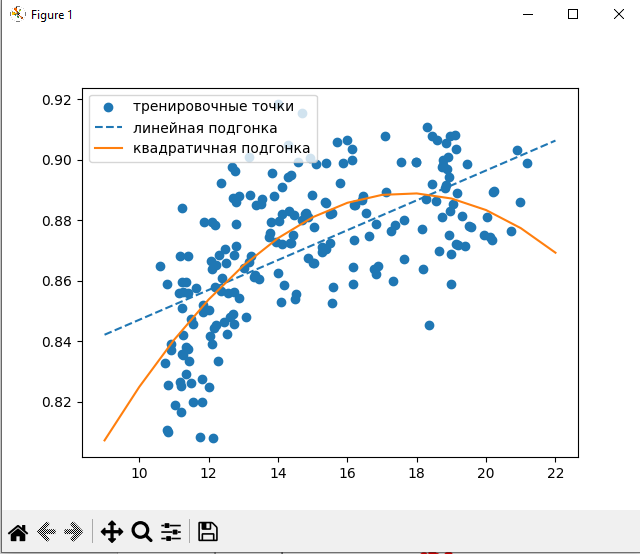
Гребневая регрессия — это модель с L2-штрафом, где к нашей функции стоимости

на основе МНК мы просто добавляем квадратичную сумму весов:

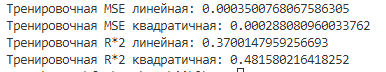




Например, если установить ll\_ratio равным 1.0, то регрессор эластичной сети ElasticNet будет равен lasso.



графике выше видно, что полиномиальная подгонка захватывает связь жду переменной отклика и объясняющей переменной намного лучше, м линейная подгонка:

****

# средневзвешенная квадатичная ошибка (MSE) в этой отдельно взятой миниатюрной задаче уменьшилась с 0.0003500768067586305 (линейная подгонка) до 0.000288080960033762 (квадратичная подгонка) , при этом коэффициент детерминации отражает более тесную подгонку к квадратичной модели (R2 = .481580216418252) в противоположность линейной подгонке (R2 = 0.3700147959256693).

**Код программы**

from matplotlib import cm

import numpy as np

import matplotlib.pyplot as plt

from sklearn.svm import SVC

import pandas as pd

from sklearn import linear\_model

from sklearn.model\_selection import train\_test\_split

from mlxtend.plotting import plot\_decision\_regions

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

url = "C:/Users/fox/Desktop/ii/data/seeds\_dataset.data"

names = ["площадь","периметр","компактность", "длина","ширина","асимметрия","длина канавки ядра","сорт"] #название атрибутов

target\_names = { 1:"Кама" , 2:"Роза" , 2:"Канадка" }

dataset = pd.read\_csv(url, names=names)

X = dataset[["площадь"]].values

y = dataset[["компактность"]].values

# X = dataset.iloc[:, :-1].values

# y = dataset.iloc[:, 7].values

dataset.head()

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.40,  random\_state = 1)

from sklearn.preprocessing import StandardScaler

scaler  = StandardScaler()

scaler.fit(X\_train)

X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

regressor = LinearRegression(  )

# обучить модель с помощью обучающих наборов

regressor.fit(X\_train, y\_train)

# прогонозы на тестовом наборе

y\_pred = regressor.predict(X\_test)

# визуализация

plt.scatter(X\_test, y\_test, color = 'red')

plt.plot(X\_train, regressor.predict(X\_train), color = 'blue')

plt.title('LinearRegression')

plt.xlabel('площадь')

plt.ylabel('компактность')

plt.show()

# -------------RANSAC------------------

from sklearn.linear\_model import RANSACRegressor

ransac = RANSACRegressor()

ransac.fit(X, y)

inlier\_mask = ransac.inlier\_mask\_

outlier\_mask = np.logical\_not(inlier\_mask)

line\_X = np.arange(9, 23, 1)

line\_y\_ransac = ransac.predict(line\_X[:, np.newaxis])

plt.scatter(X[inlier\_mask], y[inlier\_mask],

            c='blue', marker='o', label='He-выбросы')

plt.scatter(X[outlier\_mask], y[outlier\_mask],

            c='lightgreen', marker='s', label='Выбросы')

plt.plot(line\_X, line\_y\_ransac, color='red')

plt.xlabel("площадь")

plt.ylabel('компактность')

plt.legend(loc='upper left')

plt.show()

# -------------оценивание------------------

# -------------остаточный график------------------

X = dataset.iloc[:, :-2].values

y = dataset['длина канавки ядра'].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.40,  random\_state = 1)

slr = LinearRegression ()

slr.fit (X\_train, y\_train)

y\_train\_pred = slr.predict(X\_train)

y\_test\_pred = slr.predict(X\_test)

plt.scatter(y\_train\_pred, y\_train\_pred - y\_train,

            c='blue', marker= 'o', label='Тренировочны еданные' )

plt .scatter (y\_test\_pred, y\_test\_pred - y\_test,

            c= 'lightgreen' , marker= 's',label= 'Тестовые данные ')

plt.xlabel('Пр едсказанные значения')

plt.ylabel('Остатoк' )

plt.legend(loc='upper left')

plt.hlines(y=0, xmin=-10 , xmax=50 , lw=2, color= 'red')

plt.xlim([- 10 , 50 ])

plt.show()

#--------------------среднеквадр

from sklearn.metrics import mean\_squared\_error

print('MSE тренировка:', mean\_squared\_error(y\_train, y\_train\_pred))

print('MSE тестирвание:', mean\_squared\_error(y\_test , y\_test\_pred))

# Мы увидим, что MSE на тренировочном наборе равна 0.02127322889789492, а MSE тестового набора намного больше со значением 0.018740387284111308,

# что указывает на то, что наша модель переподогнана под тренировочные данные.

from sklearn.metrics import r2\_score

print('R 2 тренировка:', r2\_score(y\_train, y\_train\_pred))

print('R 2 тестирвание:', r2\_score(y\_test , y\_test\_pred))

# После оценивания на тренировочных данных коэффициент детерминации R2 нашей модели состав яет 0.911107122311821, что является неплохим результатом.

# Однако R2 на тестовом наборе данных составил всего 0.9208298600711757

#-------------------применение регуляризованых методов

X = dataset[["площадь"]].values

y = dataset[["компактность"]].values

dataset.head()

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.40,  random\_state = 1)

#--------------------гребнева

from sklearn.linear\_model import Ridge

ridge = Ridge(alpha=1.0)

# обучить модель с помощью обучающих наборов

ridge.fit(X\_train, y\_train)

# прогонозы на тестовом наборе

y\_pred = ridge.predict(X\_test)

# визуализация

plt.scatter(X\_test, y\_test, color = 'red')

plt.plot(X\_train, ridge.predict(X\_train), color = 'blue')

plt.title('Ridge')

plt.xlabel('площадь')

plt.ylabel('компактность')

plt.show()

#--------------------lasso

from sklearn.linear\_model import Lasso

lasso = Lasso(alpha=1.0)

# обучить модель с помощью обучающих наборов

lasso.fit(X\_train, y\_train)

# прогонозы на тестовом наборе

y\_pred = lasso.predict(X\_test)

# визуализация

plt.scatter(X\_test, y\_test, color = 'red')

plt.plot(X\_train, lasso.predict(X\_train), color = 'blue')

plt.title('lasso')

plt.xlabel('площадь')

plt.ylabel('компактность')

plt.show()

#--------------------эластичная сеть

from sklearn.linear\_model import ElasticNet

lassoE = ElasticNet(alpha=1.0, l1\_ratio=0.5)

# обучить модель с помощью обучающих наборов

lassoE.fit(X\_train, y\_train)

# прогонозы на тестовом наборе

y\_pred = lassoE.predict(X\_test)

# визуализация

plt.scatter(X\_test, y\_test, color = 'red')

plt.plot(X\_train, lassoE.predict(X\_train), color = 'blue')

plt.title('lassoE')

plt.xlabel('площадь')

plt.ylabel('компактность')

plt.show()

#--------------------полиномиальная

from sklearn.preprocessing import PolynomialFeatures

lr = LinearRegression()

pr = LinearRegression()

quadratic = PolynomialFeatures(degree=2)

X\_quad = quadratic.fit\_transform(X)

# Для равнения выполнить подгонку простой линейной регрессионной модели :

lr.fit(X, y)

X\_fit = np.arange(9, 23, 1) [:, np.newaxis]

y\_lin\_fit = lr.predict(X\_fit)

# полнить подгонку множественной регрессионной модели на преобразованных при наках я полиномиальной регр ссии:

pr.fit (X\_quad, y)

y\_quad\_fit = pr.predict (quadratic.fit\_transform(X\_fit ))

plt.scatter(X, y, label='тренировочные точки')

plt.plot (X\_fit, y\_lin\_fit,

            label='линейная подгонка', linestyle='--')

plt.plot (X\_fit, y\_quad\_fit,

            label='квадратичная подгонка')

plt.legend(loc='upper left')

plt.show()

y\_lin\_pred = lr.predict (X)

y\_quad\_pred = pr.predict (X\_quad)

print('Тренировочная MSE линейная:', (mean\_squared\_error(y, y\_lin\_pred)))

print('Тренировочная MSE квадратичная:', (mean\_squared\_error(y, y\_quad\_pred)))

print('Тренировочная R\*2 линейная:', (r2\_score(y, y\_lin\_pred)))

print('Тренировочная R\*2 квадратичная:', (r2\_score(y, y\_quad\_pred)))