ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НИЖЕГОРОДСКИЙ
ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Р. Е. АЛЕКСЕЕВА

Кафедра «Прикладная математика»

Лабораторная работа №5

по дисциплине «Базы данных»

Тема: «Регрессия, регрессионная модель»

Построение модели предсказания: линейная модель, криволинейная модель, выбросы, подгонка, оценка качества.

Студент

_____ Валькова.Н.П.
(Подпись) (Фамилия, И., О.)

18- ПМ _____ …….
(Группа) (Дата сдачи)

Проверил

_____ Моисеев А.Е
(Подпись) (Фамилия, И., О.)

Отчет защищен «_____» _____2021__г.
с оценкой _____

Нижний Новгород, 2021

Оглавление

1.Введение.....3
2.Постановка задачи.....5
3. Решение.....6

Введение:

Линейная регрессия — используемая в статике регрессионная модель зависимости одной (объясняемой, зависимой) переменной от другой или нескольких других переменных (факторов, регрессоров, независимых переменных) с линейной функцией зависимости зависимости.

Цель линейной регрессии — поиск линии, которая наилучшим образом соответствует этим точкам. Математическое уравнение, которое оценивает линию простой (парной) линейной регрессии:

$$Y=a+bx.$$

Таким образом, решение линейной регрессии определяет значения для a и b, так что f (x) приближается как можно ближе к y.

Криволинейная регрессия

В большинстве случаев связь биологических признаков не бывает линейной, они изменяются либо с разной скоростью, либо в разных масштабах. Соответственно на графике форма такой связи отображается не прямой, а кривой линией. Например, геометрическая прогрессия роста численности популяции в оптимальных условиях. В подобных случаях эффективнее использовать не уравнения прямой линии, а разнообразные уравнения кривых линий.

Поскольку метод наименьших квадратов исходно ориентирован на линию (поиск уравнения линии, наименее удавленной ото всех эмпирических точек), прямой расчёт уравнений кривых в рамках регрессионного анализа невозможен. Натурные данные необходимо предварительно «выпрямить», т.е. сделать возможным вычисление линейного уравнения регрессии с тем, чтобы потом из него получить уравнение криволинейной связи.

Оценка качества:

Средневзвешенная квадратичная ошибка (MSE) равна остаточной сумме квадратов (RSS) деленной на

остаточную степень свободы, где остаточная степень свободы — это размер выборки минус число модельных параметров.

$$MSE = \frac{1}{n} SSE = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)} \right)^2$$

Коэффициент детерминации — это единица минус доля необъяснённой дисперсии (дисперсии случайной ошибки модели, или условной по факторам дисперсии зависимой переменной) в дисперсии зависимой переменной.

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\sum_{1=1}^{n} \left(y^{(i)} - \hat{y}^{(i)} \right)^2}{\sum_{1=1}^{n} \left(y^{(i)} - \mu_y \right)^2}$$

Постановка задачи:

1. Используя данные из Sberbank Russian Housing Market, скачанные с сайта kaggle построить предсказательные модели.

Решение

Для построения модели было выбрано и попарно рассмотрено 3 критерия:

- full_sq
- life_sq
- price_doc

Выбор осуществлялся на основании результатов расчёта корреляций проведённых в прошлой лабораторной работе.

full_sq	0,25
life_sq	
full_sq	0,63
price_doc	

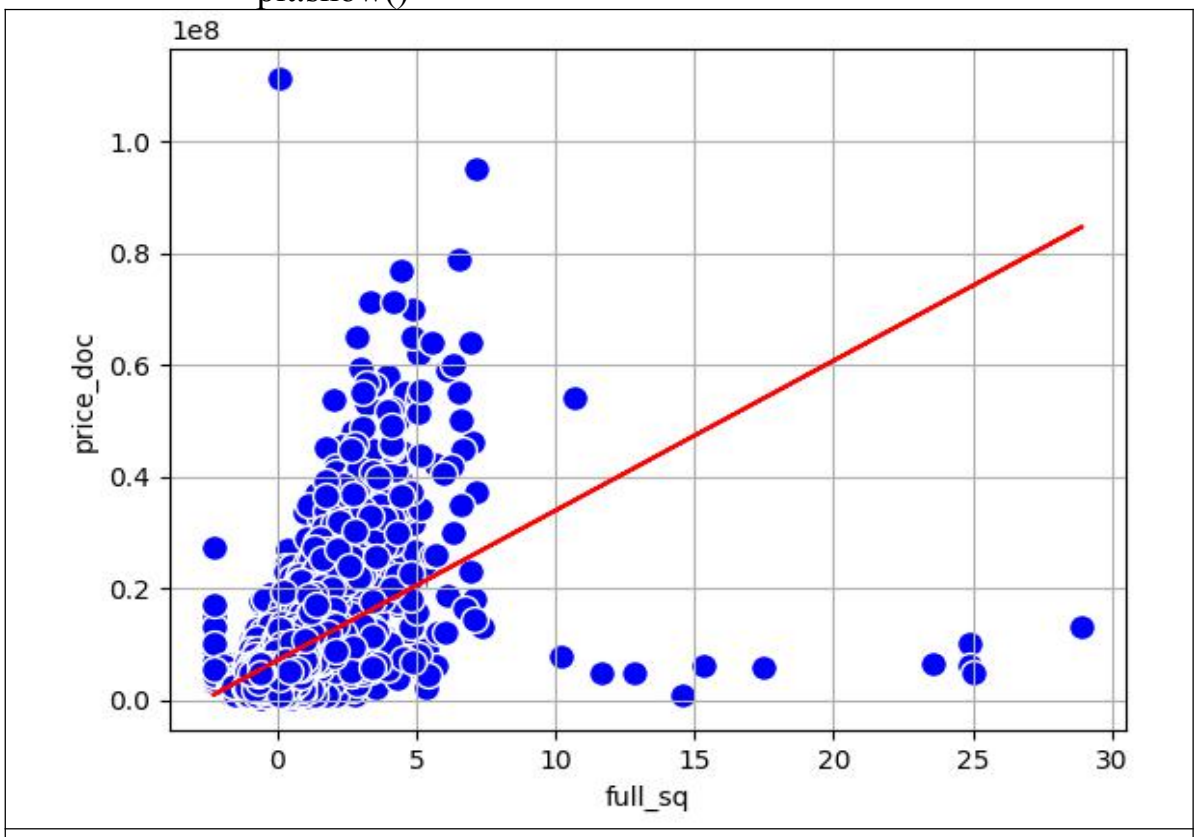
Импортируем:

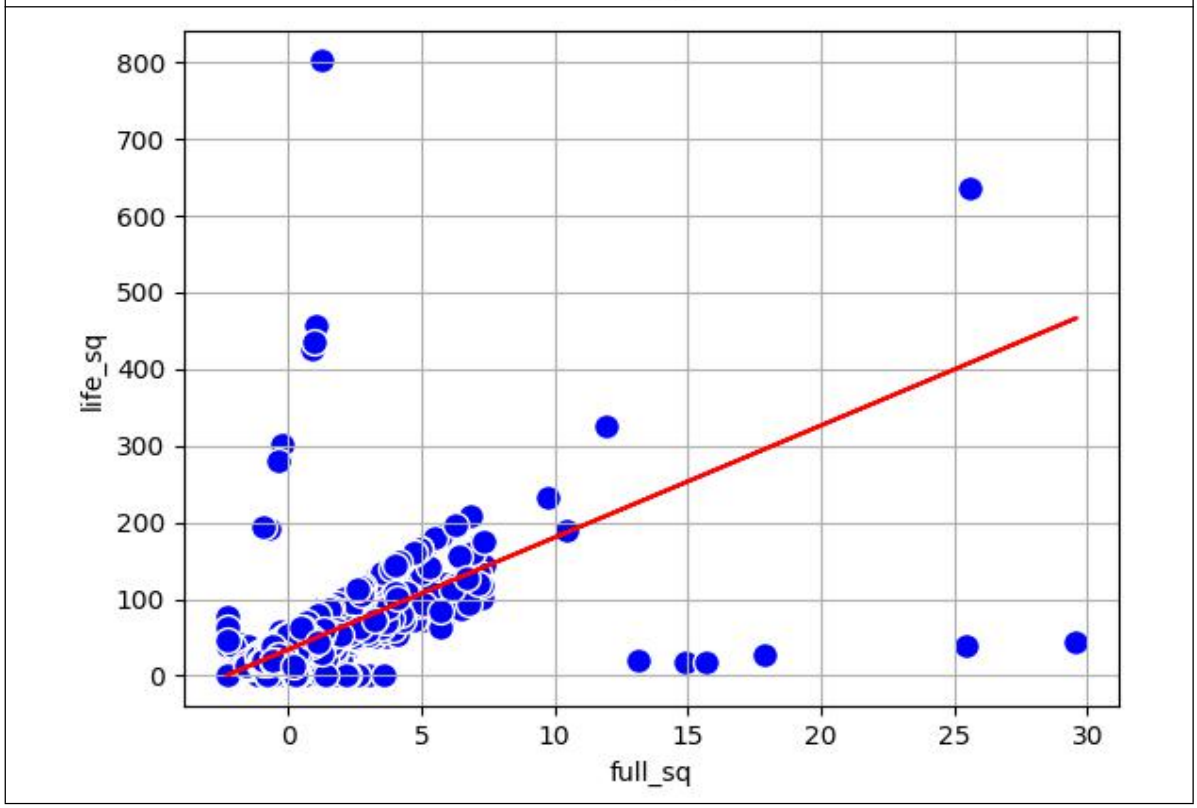
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import RANSACRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
```

Строим предсказательную модель (линейная регрессия):

```
def LinReg(X,y):
    lr = LinearRegression()
    lr.fit(X,y)
    plt.grid()
    plt.scatter(X, y, c='blue', marker='o',
    edgecolors='white', s=100)
    plt.plot(X, lr.predict(X), color='red')
    plt.xlabel(u'full_sq')
```

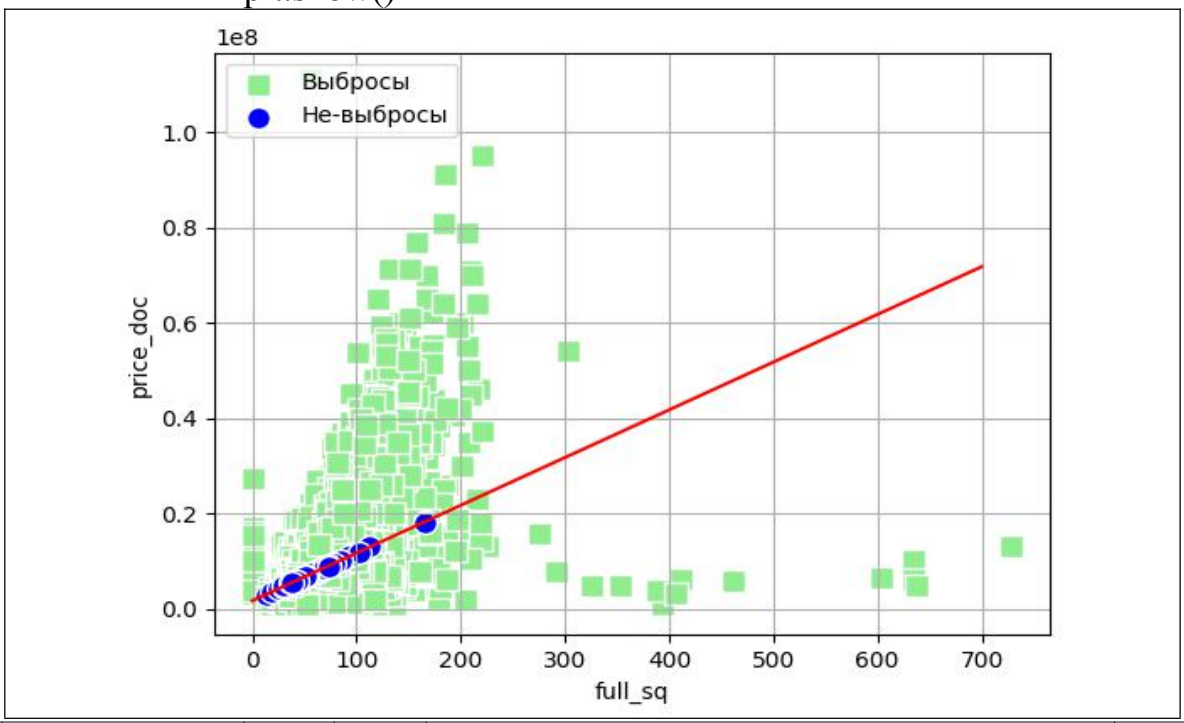
```
plt.ylabel(u'life_sq')
plt.savefig("LineReg1.png")
plt.show()
```

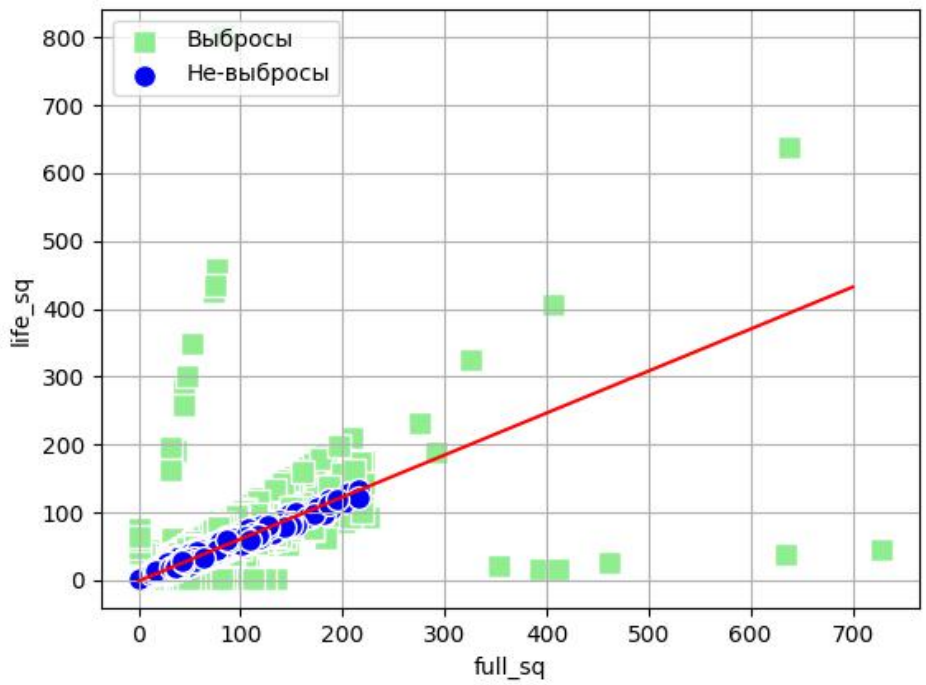




Подгонка модели, удаление выбросов, алгоритм RANSAC()
residual_threshold для price_doc = 25000, life_sq = 10.

```
def RANSAC(X,y):
        ransac = RANSACRegressor(LinearRegression(),
        max_trials=int(X.size/4), min_samples=int(X.size/8),
loss='absolute_loss',
        residual_threshold=10, random_state=0)
        ransac.fit(X,y)
        inlier_mask = ransac.inlier_mask_
        outlier_mask = np.logical_not(inlier_mask)
        X_fit = np.linspace(0, 700)[:, np.newaxis]
        plt.grid()
        plt.scatter(X[outlier_mask], y[outlier_mask], c =
'lightgreen', marker = 's', edgecolors='white', s=100, label=
u'Выбросы')
        plt.scatter(X[inlier_mask], y[inlier_mask], c='blue',
                        marker='o', edgecolors='white', s=100,
label=u'Не-выбросы')
        plt.plot(X_fit, ransac.predict(X_fit), color='red')
        plt.xlabel(u'full_sq')
        plt.ylabel(u'life_sq')
        plt.legend(loc='upper left')
        plt.savefig("RANSAC1.png")
        plt.show()
```





Для определения выбросов в случае графика для цены и площади пришлось взять достаточно большой максимальный остаток(residual_threshold), при меньших значениях на графике не выбросы визуально все находились в одной точке.

Оценка качества модели (средневзвешенная квадратичная ошибка (MSE), коэффициент детерминации R 2, график), коэффициент детерминации R 2, график остатков):

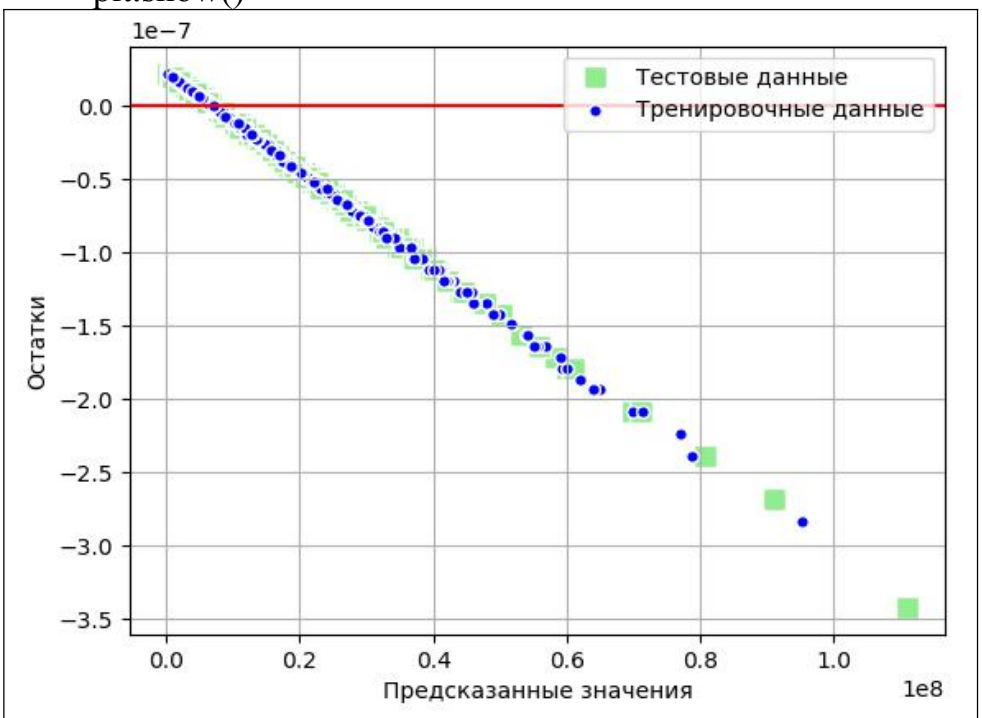
```
def Quality_estimation(df):
        X = df.iloc[:,:-1].values
        # y = df[['price_doc']].values
        y = df[['life_sq']].values

        X_train, X_test, y_train, y_test = train_test_split(X, y,
        test_size=0.3, random_state=0)

        lr = LinearRegression()
        lr.fit(X_train, y_train)
        y_train_pred = lr.predict(X_train)
        y_test_pred = lr.predict(X_test)

        print(u'MSE тренировка: ' +
str(mean_squared_error(y_train, y_train_pred)))
        print(u'MSE тестирование: ' +
```

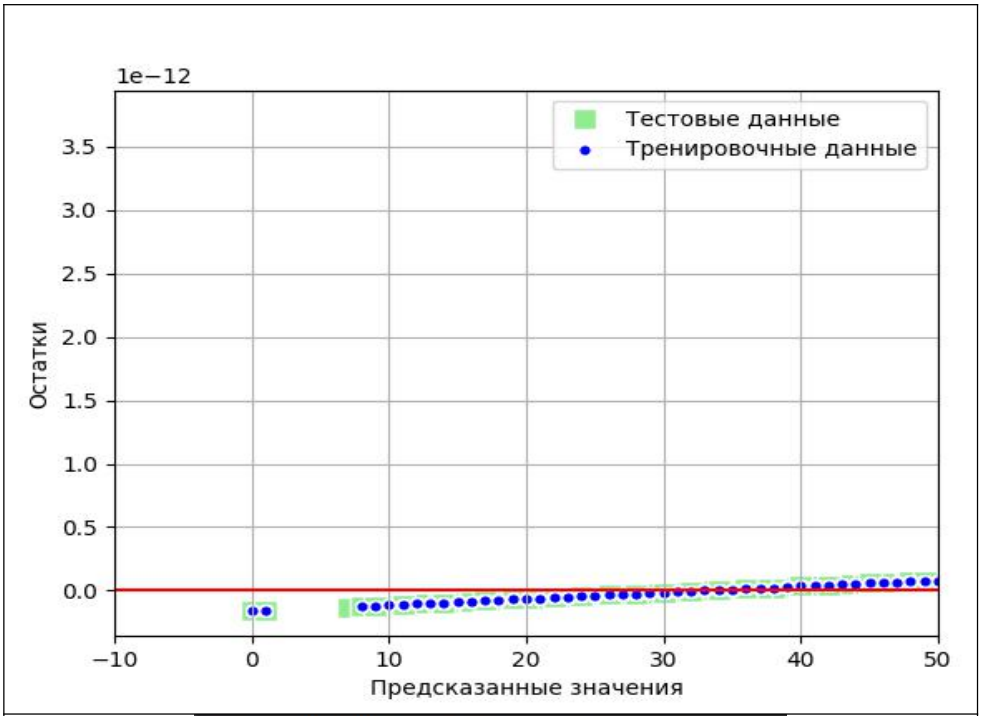
```
        str(mean_squared_error(y_test, y_test_pred)))
        print(u'\nR^2 тренировка: ' + str(r2_score(y_train,
        y_train_pred)))
        print(u'R^2 тестирование: ' +
str(r2_score(y_test,y_test_pred)))
        plt.grid()
        plt.xlim([-10, 50])
        plt.scatter(y_test_pred, y_test_pred - y_test, c
='lightgreen', marker = 's', edgecolors='white', s=100, label=
u'Тестовые данные')
        plt.scatter(y_train_pred, y_train_pred - y_train, c ='blue',
marker = 'o', edgecolors='white', s=30, label
=u'Тренировочные данные')
        plt.hlines(y=0, xmin=-10, xmax=50, color='red')
        # plt.axhline(y=0, xmin=-10, xmax=50, color='red')
        plt.xlabel(u'Предсказанные значения')
        plt.ylabel(u'Остатки')
        plt.legend(loc='upper right')
        plt.savefig("life_full1.png")
        plt.show()
```




```
MSE тренировка: 2.4890876623803176e-16
MSE тестирование: 2.7379153627157103e-16

R^2 тренировка: 1.0
R^2 тестирование: 1.0
```

1	Вып.	Валькова.Н.П.			ЛР по предмету «Базы данных»-НГТУ-(18-ПМ)	Лист
2	Пров.	Моисеев А.Е				№
№		Ф.И.О.	Подп.	Дата		10



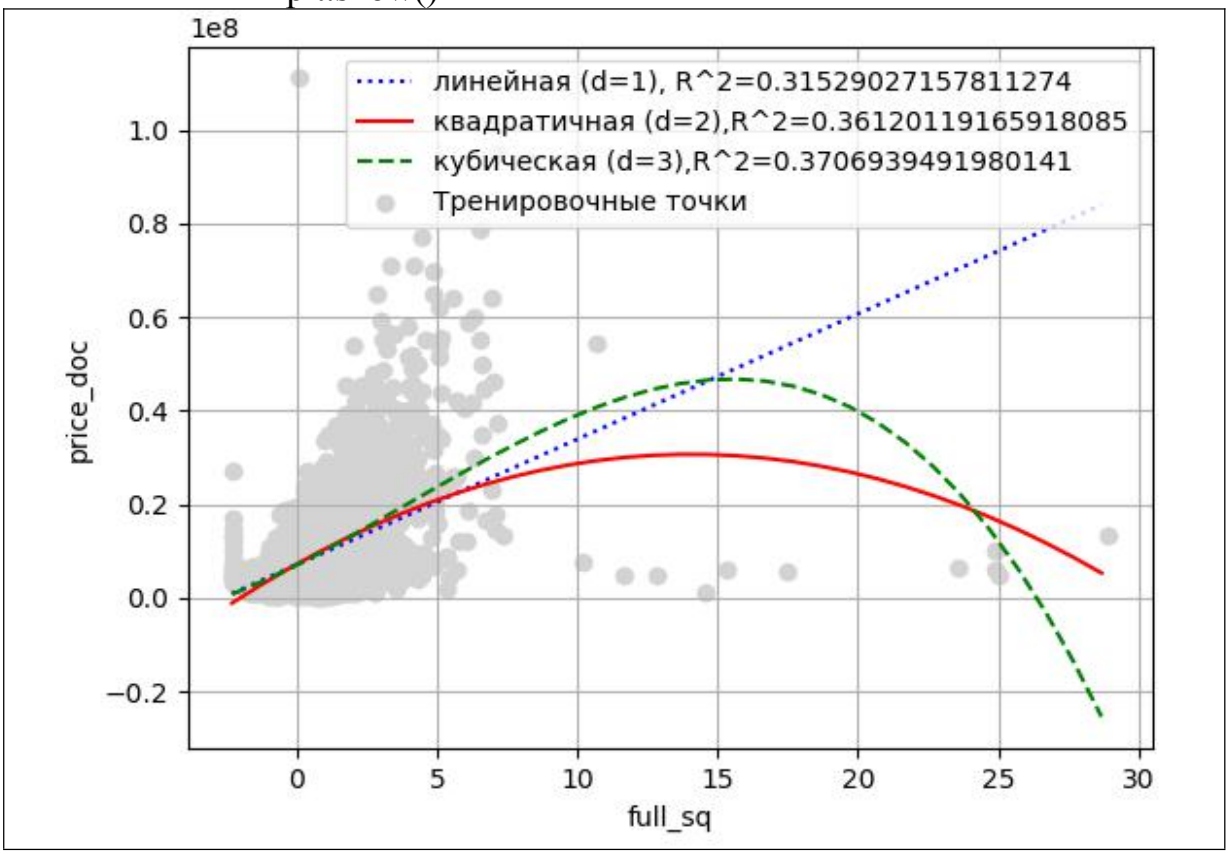

```
MSE тренировка: 1.0258937395019733e-26
MSE тестирование: 1.0392721403688816e-26

R^2 тренировка: 1.0
R^2 тестирование: 1.0
```

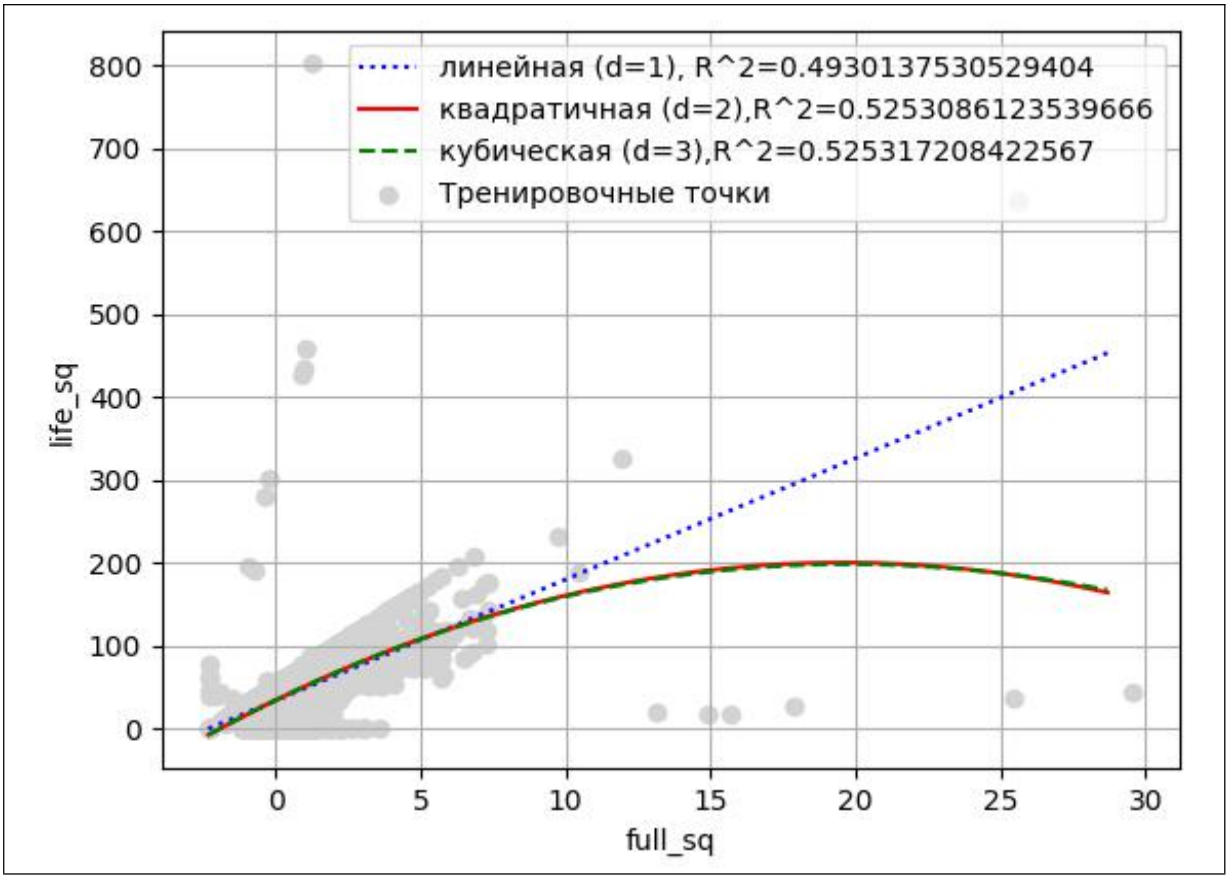
Строим полиномиальную предсказательную модель (линейная,квадратичная, кубическая):

```
def PolyReg(X,y) :
        regr = LinearRegression()
        quad = PolynomialFeatures(degree=2)
        cubic = PolynomialFeatures(degree=3)
        X_quad = quad.fit_transform(X)
        X_cubic = cubic.fit_transform(X)
    #линейная
        X_fit = np.arange(X.min(), X.max(), 1)[:, np.newaxis]
        regr.fit(X,y)
        y_lin_fit = regr.predict(X_fit)
        linear_r2 = r2_score(y, regr.predict(X))
    #квадратичная
        regr.fit(X_quad,y)
        y_quad_fit = regr.predict(quad.fit_transform(X_fit))
        quad_r2 = r2_score(y, regr.predict(X_quad))
      #кубическая
        regr.fit(X_cubic,y)
        y_cubic_fit = regr.predict(cubic.fit_transform(X_fit))
```

```
        cubic_r2 = r2_score(y, regr.predict(X_cubic))
        plt.scatter(X, y, label=u'Тренировочные точки',
        color='lightgray')
        plt.plot(X_fit, y_lin_fit, label='линейная (d=1), R^2='
+str(linear_r2), color='blue', linestyle=':')
        plt.plot(X_fit, y_quad_fit, label='квадратичная
(d=2),R^2=' + str(quad_r2), color='red', linestyle='-')
        plt.plot(X_fit, y_cubic_fit, label='кубическая
(d=3),R^2=' + str(cubic_r2), color='green', linestyle='--')
        plt.grid()
        plt.legend(loc='upper right')
        plt.xlabel(u'full_sq')
        plt.ylabel(u'life_sq')
        plt.savefig("polarreg1.png")
        plt.show()
```

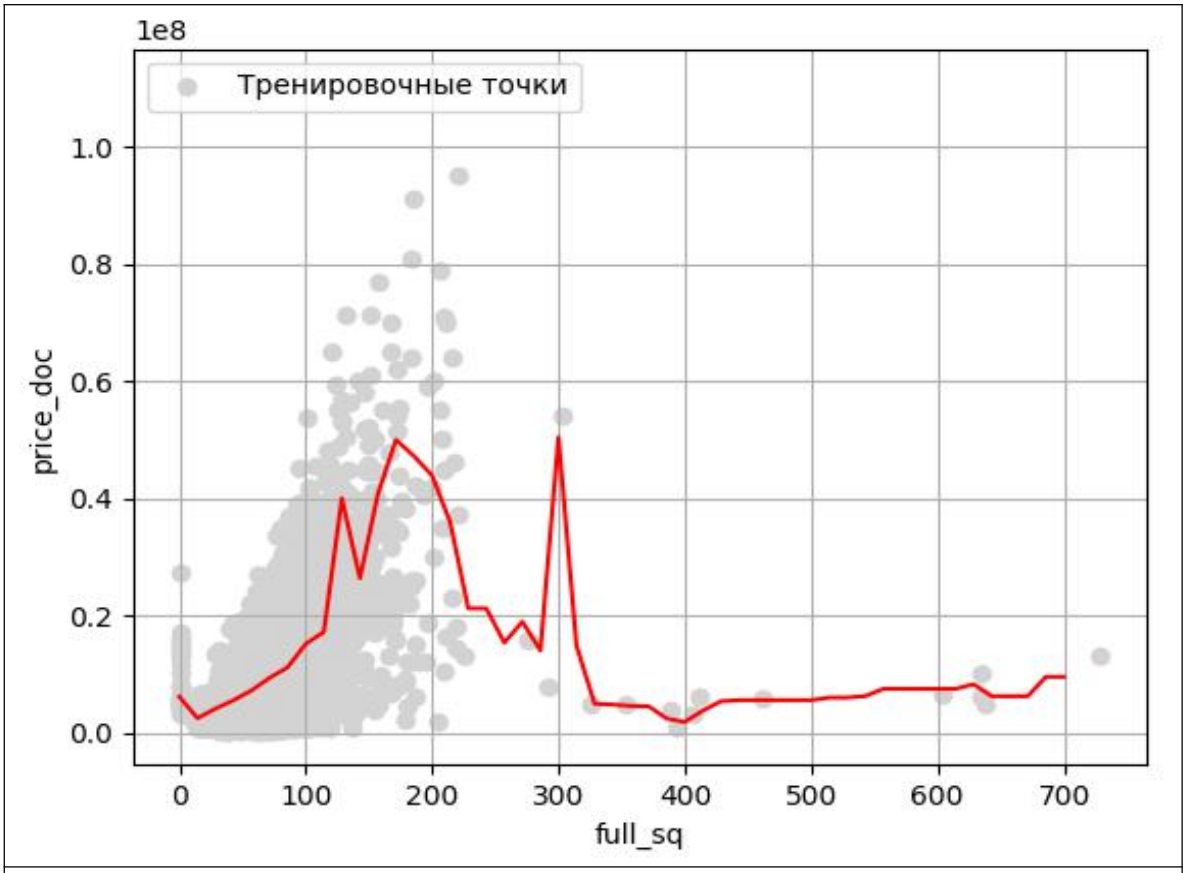


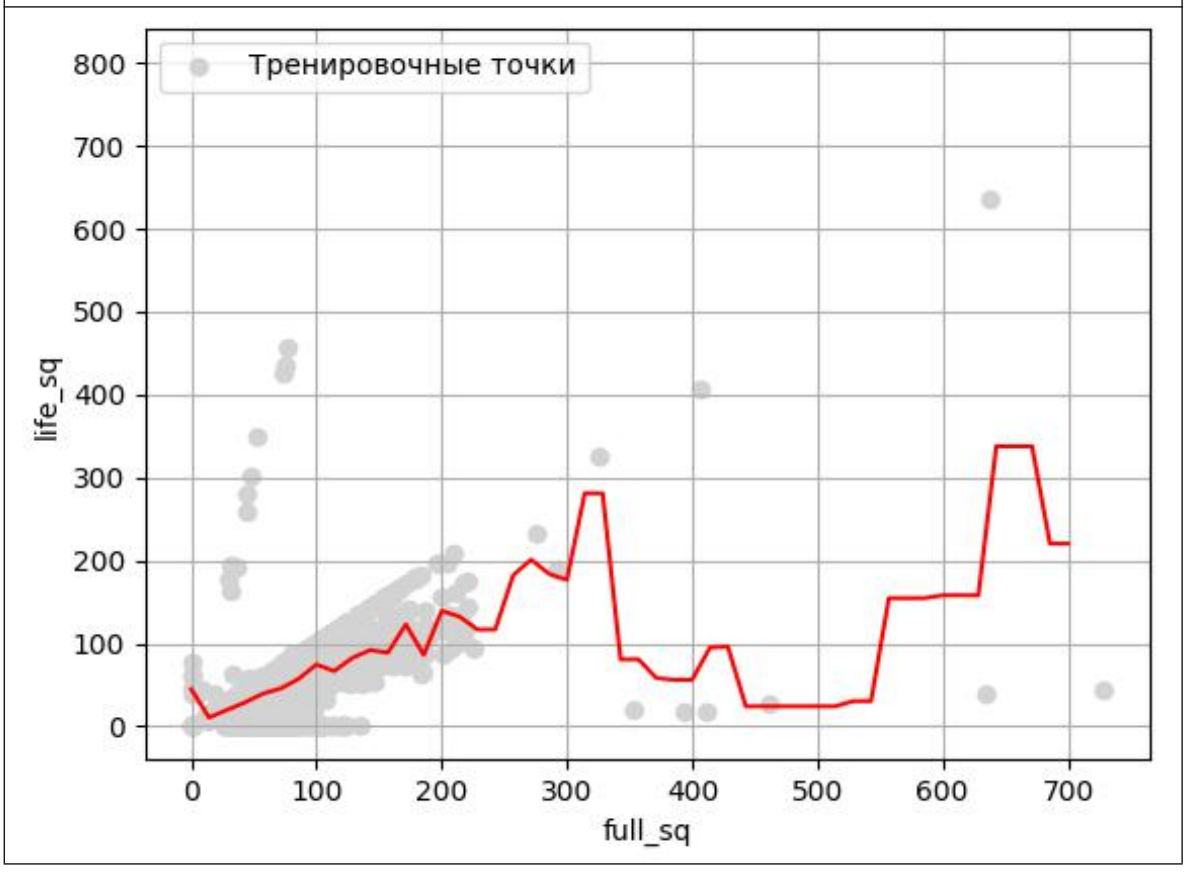
1	Вып.	Валькова.Н.П.			ЛР по предмету «Базы данных»-НГТУ-(18-ПМ)	Лист
2	Пров.	Моисеев А.Е				№
№		Ф.И.О.	Подп.	Дата		12



Строим предсказательную модель (RandomForestRegressor):

```
def RandomForest(X,y):
        X_fit = np.linspace(0, 700)[:, np.newaxis]
        plt.scatter(X, y, label=u'Тренировочные точки',
                            color='lightgray')
        regressor = RandomForestRegressor(n_estimators=10,
random_state=0)
        regressor.fit(X, y)
        plt.grid()
        y_pred = regressor.predict(X_fit)
        plt.plot(X_fit, y_pred,    color='red')
        plt.xlabel(u'full_sq')
        plt.ylabel(u'life_sq')
        plt.legend(loc='upper left')
        plt.savefig("RandomForest.png")
    plt.show()
```





1	Вып.	Валькова.Н.П.			ЛР по предмету «Базы данных»-НГТУ-(18-ПМ)	Лист
2	Пров.	Моисеев А.Е				№
№		Ф.И.О.	Подп.	Дата		14

Листинг:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import RANSACRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor

def RandomForest(X,y):
	X_fit = np.linspace(0, 700)[:, np.newaxis]
	plt.scatter(X, y, label=u'Тренировочные точки',
			color='lightgray')
	regressor = RandomForestRegressor(n_estimators=10,
random_state=0)
	regressor.fit(X, y)
	plt.grid()
	y_pred = regressor.predict(X_fit)
	plt.plot(X_fit, y_pred,    color='red')
	plt.xlabel(u'full_sq')
	plt.ylabel(u'life_sq')
	plt.legend(loc='upper left')
	plt.savefig("RandomForest.png")
	plt.show()


def LinReg(X,y):
	lr = LinearRegression()
	lr.fit(X,y)
	plt.grid()
	plt.scatter(X, y, c='blue', marker='o',
	edgecolors='white', s=100)
	plt.plot(X, lr.predict(X), color='red')
	plt.xlabel(u'full_sq')
	plt.ylabel(u'life_sq')
	plt.savefig("LineReg.png")
	plt.show()

def RANSAC(X,y):
```

```
        ransac = RANSACRegressor(LinearRegression(),
        max_trials=int(X.size/4), min_samples=int(X.size/8),
loss='absolute_loss',
        residual_threshold=25000, random_state=0)
        ransac.fit(X,y)
        inlier_mask = ransac.inlier_mask_
        outlier_mask = np.logical_not(inlier_mask)
        X_fit = np.linspace(0, 700)[:, np.newaxis]
        plt.grid()
        plt.scatter(X[outlier_mask], y[outlier_mask], c =
'lightgreen', marker = 's', edgecolors='white', s=100, label=
u'Выбросы')
        plt.scatter(X[inlier_mask], y[inlier_mask], c='blue',
                            marker='o', edgecolors='white', s=100,
label=u'Не-выбросы')
        plt.plot(X_fit, ransac.predict(X_fit), color='red')
        plt.xlabel(u'full_sq')
        plt.ylabel(u'life_sq')
        plt.legend(loc='upper left')
        plt.savefig("RANSAC.png")
        plt.show()

def Quality_estimation(df):
        X = df.iloc[:,:-1].values
        y = df[['price_doc']].values
        # y = df[['life_sq']].values

        X_train, X_test, y_train, y_test = train_test_split(X, y,
        test_size=0.3, random_state=0)



        lr = LinearRegression()
        lr.fit(X_train, y_train)
        y_train_pred = lr.predict(X_train)
        y_test_pred = lr.predict(X_test)

        print(u'MSE тренировка: ' +
str(mean_squared_error(y_train, y_train_pred)))
        print(u'MSE тестирование: ' +
        str(mean_squared_error(y_test, y_test_pred)))
        print(u'\nR^2 тренировка: ' + str(r2_score(y_train,
        y_train_pred)))
        print(u'R^2 тестирование: ' +
```

```
str(r2_score(y_test,y_test_pred)))
        plt.grid()
        plt.xlim([-10, 50])
        plt.scatter(y_test_pred, y_test_pred - y_test, c
='lightgreen', marker = 's', edgecolors='white', s=100, label=
u'Тестовые данные')
        plt.scatter(y_train_pred, y_train_pred - y_train, c ='blue',
marker = 'o', edgecolors='white', s=30, label
=u'Тренировочные данные')
        plt.hlines(y=0, xmin=-10, xmax=50, color='red')
        # plt.axhline(y=0, xmin=-10, xmax=50, color='red')
        plt.xlabel(u'Предсказанные значения')
        plt.ylabel(u'Остатки')
        plt.legend(loc='upper right')
        plt.savefig("qe.png")
        plt.show()

def PolyReg(X,y) :
          regr = LinearRegression()
          quad = PolynomialFeatures(degree=2)
          cubic = PolynomialFeatures(degree=3)
          X_quad = quad.fit_transform(X)
          X_cubic = cubic.fit_transform(X)
      #линейная
          X_fit = np.arange(X.min(), X.max(), 1)[:, np.newaxis]
          regr.fit(X,y)
          y_lin_fit = regr.predict(X_fit)
          linear_r2 = r2_score(y, regr.predict(X))
      #квадратичная
          regr.fit(X_quad,y)
          y_quad_fit = regr.predict(quad.fit_transform(X_fit))
          quad_r2 = r2_score(y, regr.predict(X_quad))
        #кубическая
          regr.fit(X_cubic,y)
          y_cubic_fit = regr.predict(cubic.fit_transform(X_fit))
          cubic_r2 = r2_score(y, regr.predict(X_cubic))
          plt.scatter(X, y, label=u'Тренировочные точки',
          color='lightgray')
          plt.plot(X_fit, y_lin_fit, label='линейная (d=1), R^2='
+str(linear_r2), color='blue', linestyle=':')
          plt.plot(X_fit, y_quad_fit, label='квадратичная
(d=2),R^2=' + str(quad_r2), color='red', linestyle='-')
          plt.plot(X_fit, y_cubic_fit, label='кубическая
```

```
(d=3),R^2=' + str(cubic_r2), color='green', linestyle='--')
        plt.grid()
        plt.legend(loc='upper right')
        plt.xlabel(u'full_sq')
        plt.ylabel(u'life_sq')
        plt.savefig("polarreg.png")
        plt.show()

if __name__ == '__main__':
    df = pd.read_csv("train.csv", index_col = 'id')
    # cols = ['price_doc', 'full_sq']
    cols = ['life_sq', 'full_sq']

    df = df[cols].dropna()

    X = df[['full_sq']].values
    # y = df[['price_doc']].values.ravel()
    y = df[['life_sq']].values.ravel()
    # Quality_estimation(df)
    # RANSAC(X, y)
    # LinReg(X,y)
    # PolyReg(X,y)
    RandomForest(X,y)
```