Нижегородский государственный университет им. Н.И. Лобачевского
Институт информационных технологий, математики и механики

Наглядный вероятностно-статистический анализ данных

Практическое задание 5

«Построение регрессии»

Пройдакова Екатерина Вадимовна,
доцент кафедры ТВиАД ИИТММ

Содержание

- Инструменты Python для построения регрессии
- Линейная регрессия
  - Пример: данные о результатах экзаменов
  - Простая линейная регрессия
  - Множественная линейная регрессия
  - Оценка качества модели
- Полиномиальная регрессия
  - Пример: данные о продаже домов
  - Регрессия нелинейная по независимым переменным и линейная по параметрам
- Практическое задание

1. ИНСТРУМЕНТЫ PYTHON ДЛЯ ПОСТРОЕНИЯ РЕГРЕССИИ

1. Инструменты Python для построения регрессии

- Библиотека scikit-learn (`sklearn`) содержит инструменты для работы с линейными регрессионными моделями (https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model)

Classical linear regressors

`linear_model.LinearRegression`([...])	Ordinary least squares Linear Regression.
`linear_model.Ridge`([alpha, fit_intercept, ...])	Linear least squares with l2 regularization.
`linear_model.RidgeCV`([alphas, ...])	Ridge regression with built-in cross-validation.
`linear_model.SGDRegressor`([loss, penalty, ...])	Linear model fitted by minimizing a regularized empirical loss with SGD

Regressors with variable selection

The following estimators have built-in variable selection fitting procedures, but any estimator using a L1 or elastic-net penalty also performs variable selection: typically `SGDRegressor` or `SGDClassifier` with an appropriate penalty.

`linear_model.ElasticNet`([alpha, l1_ratio, ...])	Linear regression with combined L1 and L2 priors as regularizer.
`linear_model.ElasticNetCV`([l1_ratio, eps, ...])	Elastic Net model with iterative fitting along a regularization path.
`linear_model.Lars`([fit_intercept, verbose, ...])	Least Angle Regression model a.k.a.
`linear_model.LarsCV`([fit_intercept, ...])	Cross-validated Least Angle Regression model.
`linear_model.Lasso`([alpha, fit_intercept, ...])	Linear Model trained with L1 prior as regularizer (aka the Lasso)
`linear_model.LassoCV`([eps, n_alphas, ...])	Lasso linear model with iterative fitting along a regularization path.
`linear_model.LassoLars`([alpha, ...])	Lasso model fit with Least Angle Regression a.k.a.
`linear_model.LassoLarsCV`([fit_intercept, ...])	Cross-validated Lasso, using the LARS algorithm.
`linear_model.LassoLarsIC`([criterion, ...])	Lasso model fit with Lars using BIC or AIC for model selection
`linear_model.OrthogonalMatchingPursuit`([...])	Orthogonal Matching Pursuit model (OMP)
`linear_model.OrthogonalMatchingPursuitCV`([...])	Cross-validated Orthogonal Matching Pursuit model (OMP).

1. Инструменты Python для построения регрессии

- Подмодуль `sklearn.linear_model.LinearRegression`: построение и верификация линейной регрессии на основе метода наименьших квадратов (МНК):
  - `fit()` : подгоняет линейную модель к выбранным данным
  - `get_params()` : возвращает значения параметров модели
  - `score()` : возвращает коэффициент детерминации
  - `predict()` : предсказывает значение на основе построенной модели

1. Инструменты Python для построения регрессии

- Подмодуль `sklearn.metrics.mean_squared_error`, `sklearn.metrics.r2_score`:
  вычисление метрик построенных моделей: средней квадратической ошибки и коэффициента детерминации

- Подмодуль `sklearn.model_selection.train_test_split`:
  метод `train_test_split()` разделения выборки на тренировочное множество для построения модели и валидационное множество для оценки качества модели при прогнозировании

2. ЛИНЕЙНАЯ РЕГРЕССИЯ

2. Линейная регрессия
2.1 Пример: данные о результатах экзаменов

- Имеются данные о результатах сдачи итоговых экзаменов по математике, чтению, письму, а также о поле, расовой принадлежности выпускника, уровне образования родителей, наличии льгот и прохождении подготовительного курса.

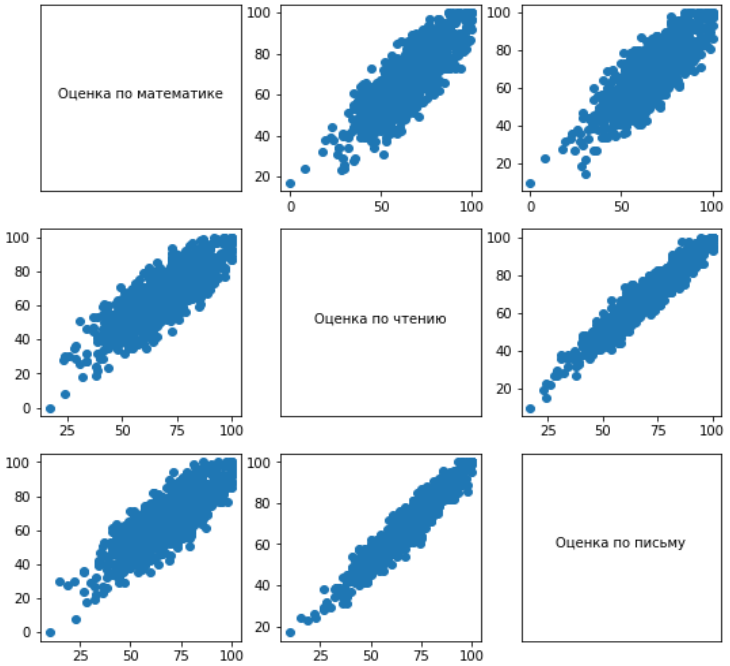
1	gender	race/ ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
2	female	group B	bachelor's degree	standard	none	72	72	74
3	female	group C	some college	standard	completed	69	90	88
4	female	group B	master's degree	standard	none	90	95	93
5	male	group A	associate's degree	free/reduced	none	47	57	44
6	male	group C	some college	standard	none	76	78	75
7	female	group B	associate's degree	standard	none	71	83	78
8	female	group B	some college	standard	completed	88	95	92

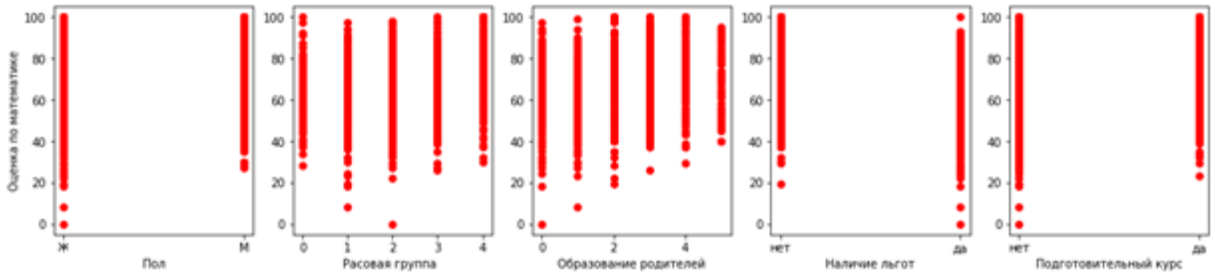
- Задача: выявить зависимость между приведенными характеристиками.
- Более узкая задача: построить модель линейной зависимости между величинами и прогнозировать значение одной величины (например, оценки по письму) на основе других (например, оценки по чтению)

2. Линейная регрессия
2.1 Пример: данные о результатах экзаменов

- Ранее произвели простейший корреляционный анализ (практическое занятие 4)

```
Математическое ожидание оценок по соответвующим предметам = [66.089, 69.169, 68.054]
Ковариационная матрица =
[[229.918998   180.99895796 184.93913313]
 [180.99895796 213.1656046  211.78666066]
 [184.93913313 211.78666066 230.90799199]]
Матрица коэффициентов корреляции =
[[1.         0.81757966 0.80264205]
 [0.81757966 1.         0.95459808]
 [0.80264205 0.95459808 1.        ]]
Смещенная оценка дисперсии оценок по математике = 229.68907899999996
Несмещенная оценка дисперсии оценок по математике = 229.91899799799796
```





2. Линейная регрессия
2.2 Простая линейная регрессия

```
50 # Разделение выборки на тренировочный набор и набор для валидации (проверки)
51 examsTr, examsVal = train_test_split(examsData, test_size = 0.2, train_size = 0.8)
52
53 # независимая переменая: оценка по математике
54 # зависимая переменная: оценка по письму
55 X = np.reshape(dataByType(examsTr, 5), (-1, 1)) # создаем из вектора вектор одномерных векторов
56 # обучение модели
57 linR = LinearRegression().fit(X, dataByType(examsTr, 7))
58 # коэффициенты регрессии
59 a = linR.coef_
60 b = linR.intercept_
61 print("Построена зависимость: Y = " + str(a[0]) + " X + " + str(b))
62 # валидация модели на проверочном множестве
63 Y_predict = linR.predict(np.reshape(dataByType(examsVal, 5), (-1, 1)))
64 # коэффициент детерминации R2
65 r2_1 = r2_score(dataByType(examsVal, 7), Y_predict)
66 # То же самое
67 linR.score(np.reshape(dataByType(examsVal, 6), (-1, 1)), dataByType(examsVal, 7))
68 print("Коэффициент детерминации = " + str(r2_1))
69 # Mean squared error - Средняя квадратическая ошибка
70 mse_1 = mean_squared_error(dataByType(examsVal, 7), Y_predict)
71 print("Средняя квадратическая ошибка (MSE) = " + str(mse_1))
72 # Residual sum of squares - Остаточная сумма квадратов
73 rss_1 = ((dataByType(examsVal, 7) - Y_predict)**2).sum()
74 print("Остаточная сумма квадратов (RSS) = " + str(rss_1))
75
```

2. Линейная регрессия
2.2 Простая линейная регрессия

- **Этап 1:** разделение имеющейся выборки (`examsData`) на две части: 80% наблюдений будут составлять тренировочные данные (`examsTr`), 20% – данные для валидации модели (`examsVal`)

```
50 # Разделение выборки на тренировочный набор и набор для валидации (проверки)
51 examsTr, examsVal = train_test_split(examsData, test_size = 0.2, train_size = 0.8)
52
```

- **Этап 2**: подгонка модели

```
53 # независимая переменая: оценка по математике
54 # зависимая переменная: оценка по письму
55 X = np.reshape(dataByType(examsTr, 5), (-1, 1)) # создаем из вектора вектор одномерных век
56 # обучение модели
57 linR = LinearRegression().fit(X, dataByType(examsTr, 7))
58 # коэффициенты регрессии
59 a = linR.coef_
60 b = linR.intercept_
61 print("Построена зависимость: Y = " + str(a[0]) + " X + " + str(b))
```

2. Линейная регрессия
2.2 Простая линейная регрессия

Метод `np.reshape()` используется для придания многомерным данным заданного формата.

Пример:

```
In [7]: dataByType(examsTr, 5)[0:10]
Out[7]: [65.0, 87.0, 29.0, 67.0, 58.0, 61.0, 79.0, 48.0, 54.0, 63.0]

In [8]: np.reshape(dataByType(examsTr, 5), (-1, 1))[0:10]
Out[8]:
array([[65.],
       [87.],
       [29.],
       [67.],
       [58.],
       [61.],
       [79.],
       [48.],
       [54.],
       [63.]])
```

Атрибуты `coef_` и `intercept_` дают значения для коэффициента наклона прямой регрессии и свободного члена соответственно

2. Линейная регрессия
2.2 Простая линейная регрессия

- **Этап 3:** прогноз относительно значений зависимой переменной на валидационном множестве

```
62 # валидация модели на проверочном множестве
63 Y_predict = linR.predict(np.reshape(dataByType(examsVal, 5), (-1, 1)))
```

- **Этап 4**: подсчет метрик: коэффициент детерминации, средняя квадратическая ошибка, остаточная сумма квадратов (строки 65-74)

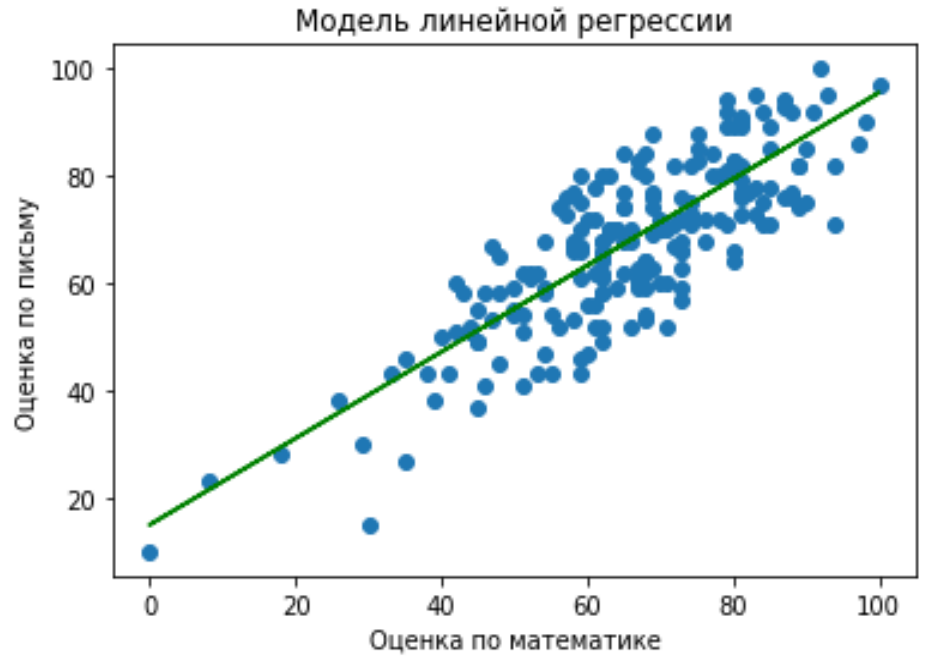
Имеем следующий результат выполнения представленного кода:

```
Построена зависимость: Y = 0.8057168430059412 X + 14.963002116684159
Коэффициент детерминации = 0.676276496700704
Средняя квадратическая ошибка (MSE) = 81.18286219979218
Остаточная сумма квадратов (RSS) = 16236.572439958436
```

2. Линейная регрессия
2.2 Простая линейная регрессия

- Визуализация данных: диаграмма рассеивания и прямая регрессии

```
76 plt.scatter(dataByType(examsVal, 5), dataByType(examsVal, 7))
77 plt.plot(dataByType(examsVal, 5), Y_predict, color = 'green')
78 plt.title("Модель линейной регрессии")
79 plt.xlabel("Оценка по математике")
80 plt.ylabel("Оценка по письму")
81 plt.show()
```



2. Линейная регрессия
2.3 Множественная линейная регрессия

- Серия моделей линейной регрессии. **Зависимая переменная всегда – оценка по письму**; независимые переменные:

  1) оценка по математике

  2) оценка по чтению

  3) личная характеристика выпускника, включающая пол, расу, уровень образования родителей, принадлежность к льготной группе, прохождение подготовительного курса

  4) личная характеристика и оценка по математике

  5) личная характеристика и оценки по математике и чтению.

- В пунктах 3) – 5) строится модель множественной регрессии с несколькими независимыми переменными

```
111    # независимые переменые: первые 5 столбцов данных
112    X = getPartOfData(examsTr, range(5))
113    # подгонка(тренировка) модели
114    linR = LinearRegression().fit(X, dataByType(examsTr, 7))
```

2. Линейная регрессия
2.4 Оценка качества модели

Незав. переменные	R2	MSE	RSS
Матем.	0.676276	81.182862	16236.572440
Чтение	0.916312	20.987110	4197.421938
Хар-ка	0.345355	164.170820	32834.163926
Хар-ка + Матем.	0.902208	24.524229	4904.845807
Хар-ка + Матем. + Чтение	0.949028	12.782766	2556.553225

- Как известно из более ранних результатов, между оценками по письму и чтению существует более сильная линейная зависимость, чем между оценками по математике и письму –> прогноз на основе оценки по чтению является более точным, чем на основе оценки по математике (коэффициент детерминации ближе к единице, ошибки существенно меньше).
- Прогноз оценки только на основе общих характеристик выпускника является наименее точным, как и ожидается. Однако, добавляя эти данные к оценке по математики, можно значительно улучшить качество модели линейной регрессии.
- Имея в распоряжении личную характеристику выпускника и результаты экзаменов по математике и чтению, можно с высокой степенью точности предсказать оценку по письму: **коэффициент детерминации R2 = 0.949028** и близок к единице, **средняя квадратическая ошибка MSE = 12.782766** мала

3. ПОЛИНОМИАЛЬНАЯ РЕГРЕССИЯ

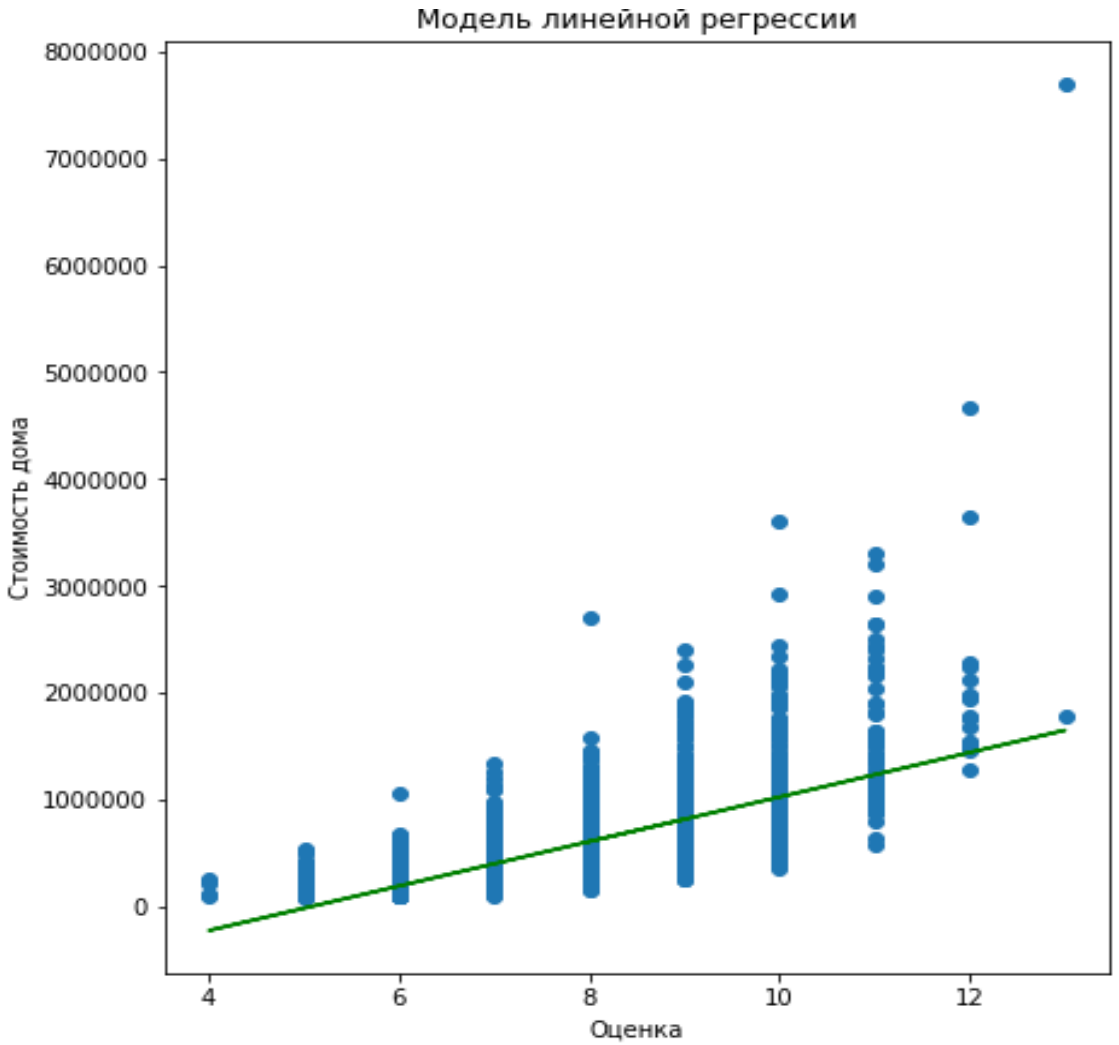
3. Полиномиальная регрессия
3.1 Пример: данные о продаже домов

- Имеются наблюдения за несколькими величинами для проданных домов, включая, стоимость дома, жилую площадь, количество этажей, общую оценку дома (от 1 до 10) и т.д. Данные взяты с сайта https://www.kaggle.com/harlfoxem/housesalesprediction
- Задача: построение прогноза о стоимости дома на основе общей оценки дома

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	sqft_living15	sqft_lot15
2	712930052	20141013	221900	3	1	1180	5650	1	0	0	3	7	1180	0	1955	0	1340	5650
3	641410019	20141209	538000	3	2.25	2570	7242	2	0	0	3	7	2170	400	1951	1991	1690	7639
4	563150040	20150225	180000	2	1	770	10000	1	0	0	3	6	770	0	1933	0	2720	8062
5	248720087	20141209	604000	4	3	1960	5000	1	0	0	5	7	1050	910	1965	0	1360	5000
6	195440051	20150218	510000	3	2	1680	8080	1	0	0	3	8	1680	0	1987	0	1800	7503

3. Полиномиальная регрессия
3.1 Пример: данные о продаже домов

- Диаграмма рассеяния дает основание подозревать наличие нелинейной регрессии по независимым переменным
- Начнем с полиномиальной регрессии, степень полинома неизвестна



3. Полиномиальная регрессия
3.2 Регрессия линейная по параметрам

- Часто полиномиальная регрессия одновременно является линейной регрессией по оцениваемому параметру
- В этом случае задачу построения полиномиальной регрессии решаем уже известными методами

```
190 def polynomicSample(data, degree):
191     res = []
192     for rec in data:
193         for every in rec:
194             for d in range(degree):
195                 res.append(every**(d+1))
196     res = np.reshape(res, (len(data), -1))
197     return res
```

3. Полиномиальная регрессия
3.2 Регрессия линейная по параметрам

Функция `polynomicSample()`

```
In [44]: np.reshape(dataByType(houseDataTr, 11), (-1, 1))[0:5]
Out[44]:
array([[10.],
       [ 8.],
       [ 8.],
       [ 8.],
       [ 7.]])

In [45]: polynomicSample(np.reshape(dataByType(houseDataTr, 11), (-1, 1)), 3)[0:5]
Out[45]:
array([[  10.,  100., 1000.],
       [   8.,   64.,  512.],
       [   8.,   64.,  512.],
       [   8.,   64.,  512.],
       [   7.,   49.,  343.]])
```

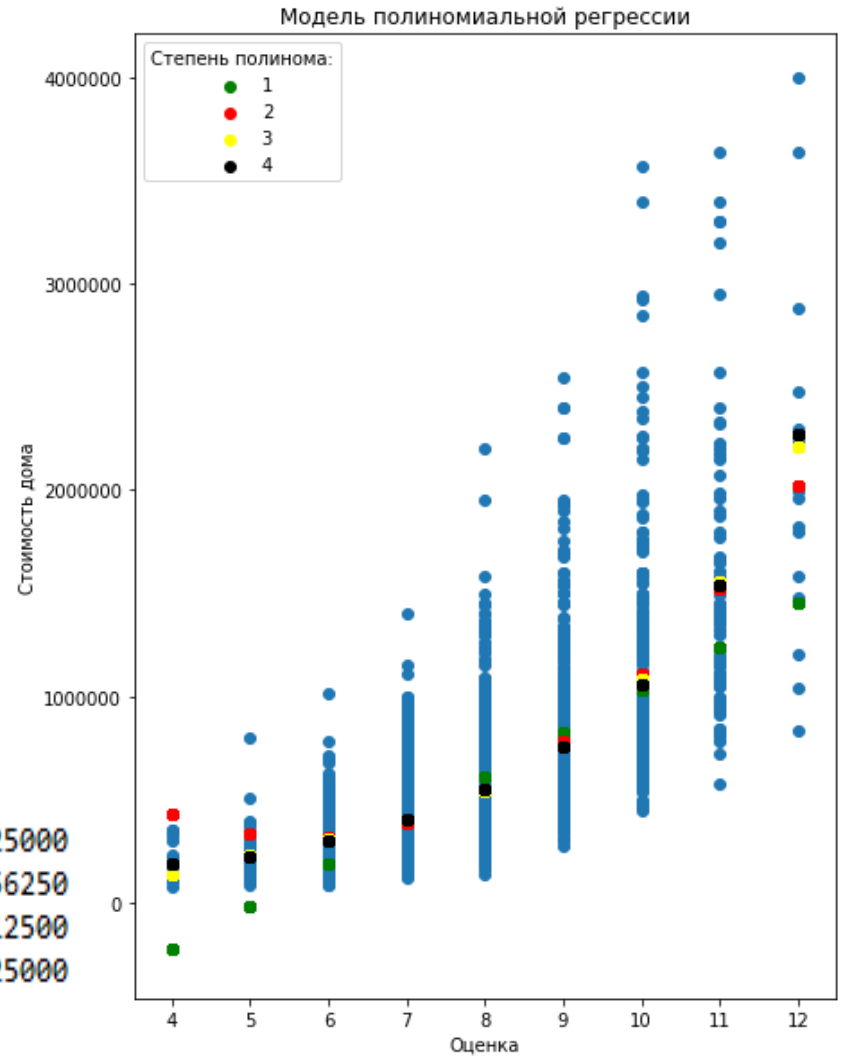
3. Полиномиальная регрессия
3.2 Регрессия линейная по параметрам

Построим **полиномиальную** регрессию, где в качестве **зависимой** переменной выступает **стоимость дома**, а в качестве **независимой – оценка дома**.

```
198
199 plts = []
200 plt.figure(figsize=(7, 10))
201 plt.scatter(dataByType(houseDataVal, 11), dataByType(houseDataVal, 2))
202 plts.append(plt.scatter(dataByType(houseDataVal, 11), Y_predict, color = 'green', label = "1"))
203 plt.title("Модель полиномиальной регрессии")
204 plt.xlabel("Оценка")
205 plt.ylabel("Стоимость дома")
206
207 colors = ["red", "yellow", "black"]
208 for d in [2, 3, 4]:
209     squareSample = polynomicSample(np.reshape(dataByType(houseDataTr, 11), (-1, 1)), d)
210     linR = LinearRegression().fit(squareSample, dataByType(houseDataTr, 2))
211     Y_predict = linR.predict(polynomicSample(np.reshape(dataByType(houseDataVal, 11), (-1,1)), d))
212     r2.append(r2_score(dataByType(houseDataVal, 2), Y_predict))
213     er.append(mean_squared_error(dataByType(houseDataVal, 2), Y_predict))
214     rss.append(((dataByType(houseDataVal, 2) - Y_predict)**2).sum())
215     plts.append(plt.scatter(dataByType(houseDataVal, 11), Y_predict, color = colors[d-2], label = str(d)))
216 plt.legend(title = "Степень полинома:", handles = plts, labels = ["1", "2", "3", "4"])
217 plt.show()
218
219 print("%18s %12s %18s %18s" % ("Степень полинома", "R2", "MSE", "RSS"))
220 models = [1, 2, 3, 4]
221 for i in range(len(r2)):
222     print("%18d %12f %18f  %18f" % (models[i], r2[i], er[i], rss[i]))
223
```

3. Полиномиальная регрессия
3.2 Регрессия линейная по параметрам

- При переходе к полиномиальной регрессии удалось достичь положительного результата, хоть и незначительного
- Модели полиномиальной регрессии степеней 2, 3, 4 незначительно различаются по качеству.
- Коэффициенты детерминации R2 все еще достаточно далек от единицы, а ошибки MSE достаточно велики - имеет смысл искать другие более точные модели нелинейной регрессии



Степень полинома	R2	MSE	RSS
1	0.442997	71253582352.469604	308029236509726.125000
2	0.497890	64231555024.029068	277673012368877.656250
3	0.498586	64142564967.215088	277288308353270.812500
4	0.497194	64320641195.064796	278058131886265.125000

4. ПРАКТИЧЕСКОЕ ЗАДАНИЕ

4. Практическое задание

- Данные – файл *05_Зачисление.xls* содержащий информацию об абитуриентах некоторой учебной магистерской программы.
  - Serial No – идентификатор,
  - GRE (Graduate Record Examinations) и TOEFL (Test of English as a Foreign Language) Score – оценки за экзамены GRE (0-140) и TOEFL(0-120),
  - University Rating – рейтинг университета, в котором абитуриент обучался в бакалавриате (0-5),
  - SOP (Statement of Purpose) и LOR (Letter of Recommendations) – оценка убедительности заявления абитуриента и рекомендательного письма,
  - CGPA (Undergraduate Grade Point Average) – средний балл в бакалавриате (0-10),
  - Research – наличие опыта исследований (0/1),
  - Chance of Admit – оценка шанса приема абитуриента (0-1).

Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
1	337	118	4	4.5	4.5	9.65	1	0.92
2	324	107	4	4	4.5	8.87	1	0.76
3	316	104	3	3	3.5	8	1	0.72
4	322	110	3	3.5	2.5	8.67	1	0.8

Источник данных: https://www.kaggle.com/mohansacharya/graduate-admissions

Mohan S Acharya, Asfia Armaan, Aneeta S Antony. *A Comparison of Regression Models for Prediction of Graduate Admissions*, IEEE International Conference on Computational Intelligence in Data Science, 2019.

4. Практическое задание

1. **Определить тип данных** для каждого столбца.
2. **Построить диаграммы рассеивания** для пар данных:

   GRE – Chance of Admit, TOEFL – Chance of Admit,

   University Rating – Chance of Admit, SOP – Chance of Admit,

   LOR – Chance of Admit, CGPA – Chance of Admit.

Привести визуальную оценку зависимости шанса поступления от указанных величин.

3. **Построить модель регрессии** (простой линейной / множественной линейной / полиномиальной по независимым переменным), которая даст наилучший прогноз шанса поступления абитуриента. В качестве независимых переменных может выбираться любое количество величин из пункта 2.

Выбрать оптимальную и обосновать оптимальность построенной модели с помощью разобранных метрик (R2, MSE, RSS)

4. Практическое задание

4.**Факультативная задача**

Написать вспомогательные методы – аналоги методов библиотеки scikit-learn:

- Метод разбиения выборки на два множества (тренировочное и валидационное) в заданном через параметры соотношении.
- Метод подгонки модели простой линейной регрессии на основе метода наименьших квадратов.
- Методы подсчета коэффициента детерминации и среднеквадратической ошибки.

Проверить правильность работы методов, сравнив результаты с результатами библиотечных методов, при построении линейной регрессии, где в качестве зависимой переменной выступает оценка GRE, а в качестве независимой – CGPA.