МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

КАФЕДРА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ И АНАЛИЗА ДАННЫХ

Отчет
о прохождении преддипломной практики

Румянцева Андрея Кирилловича
студента 4 курса, специальность
"прикладная математика"

Руководитель практики:
зав. кафедрой ММАД,
канд. физ.-мат. наук, доцент
Бодягин Игорь Александрович

Минск, 2019

Содержание

Задание на практику

- Провести аналитический обзор литературы методов статистического анализа данных при наличии классифицированных наблюдений с искажениями.
- Реализовать альтернативные встречаемые в литературе методы статистического анализа данных при наличии классифицированных наблюдений с искажениями.
- Провести сравнительный анализ реализованного в ходе курсового проекта метода с альтернативными.
- Обобщить все реализованные методы с линейной на полиномиальную регрессию.
- Подготовить отчет по преддипломной практике.

ВВЕДЕНИЕ

Целью преддипломной практики было продолжение исследования и улучшение оценок, построенных в курсовом проекте. Темой курсового проекта было *"Статистическое оценивание параметров линейной регрессии с выбросами при наличии группирования наблюдений"*.

В ходе курсового проекта оценки строили для модели линейной регрессии с выбросами:

$$y_i^{\widetilde{\varepsilon}} = (\xi_i)y_i + (1-\xi_i)\eta_i, \tag{1}$$

где ξ_i принимает значение, равное 1, с вероятностью $1-\widetilde{\varepsilon}$ и значение, равное 0, с вероятностью $\widetilde{\varepsilon}$, т.е.:

$$\begin{cases} p(\xi_i = 0) = \widetilde{\varepsilon}, \\ p(\xi_i = 1) = 1-\widetilde{\varepsilon}, \end{cases}, \tag{2}$$

η_i-случайная величина из некоторого вообще говоря неизвестного распределения.

$y_i^{\widetilde{\varepsilon}}$ – i-е наблюдение из N наблюдений(N-объем выборки), $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ регрессоры, $\{\beta_k, k = \overline{0,n}\}$– параметры регрессии, а ε_i – случайная ошибка i-го эксперемента, распределение которой подчиняется нормальному закону с нулевым математическим ожиданием и дисперсией σ^2.

$$y_i = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \ldots \\ \beta_n \end{pmatrix} \times \begin{pmatrix} 1 \\ x_{i1} \\ \ldots \\ x_{in} \end{pmatrix}^T + \varepsilon_i, \tag{3}$$

Параметр ξ_i имеет следующий содержательный смысл: если $\xi_i = 0$, то вместо истинного значения мы наблюдаем выброс, если $\xi_i = 1$, то наблюдается истинное значение. Переменную $\widetilde{\varepsilon}$ будем называть долей аномальных наблюдений. Величины ξ_i, x_i и η_i являются независимыми.

Каждый y_i принадлежит нормальному распределению:

$$y_i = f(x_i, \beta) + \varepsilon_i \sim \mathcal{N}(f(x_i, \beta), \sigma^2). \tag{4}$$

Разделим множество значений функции регрессии, т.е множество $\mathcal{R}$, на k полуинтервалов:

$$\mathcal{R} = (-\infty, a_1] \bigcup (a_1, a_2] \bigcup \cdots \bigcup (a_{k-1}, +\infty). \tag{5}$$

Обозначим полученные интервалы: $\nu_0, \dots, \nu_{k-1}$.

Далее в работе будем считать, что вместо истинных значений зависимых переменных y_i наблюдается только номер класса, к которому это наблюдение попало. Тогда для каждого y_i будем наблюдать лишь номер полуинтервала μ_i, в который он попал.

$$\mu_i = j, \text{если } y_i \text{ отнесли к полуинтервалу } \nu_j. \tag{6}$$

В курсовом проекте решалась задача статистического оценивания параметров модели $\{\beta_k, k = \overline{0, n}\}$ по известным группированным наблюдениям с аномалиями.

Для этого строилась функция правдоподобия:

$$\begin{aligned} l(\beta, \sigma^2, \nu_0, \dots, \nu_{k-1}) &= \ln(\prod_{i=1}^{n} P(\mu_i = j)) = & (7)\\ &= \sum_{i=1}^{n} \ln(P(\mu_i = j)). & (8) \end{aligned}$$

Для максимизирования функции правдоподобия решалась система уравнения:

$$\frac{\delta l}{\delta \beta} = 0, \tag{9}$$

где:

$$\begin{aligned} \frac{\delta l}{\delta \beta} &= \frac{\delta \sum_{i=1}^{n} \ln(P(\mu_i = j))}{\delta \beta} = \frac{\delta \sum_{i=1}^{n} \ln P(y_i \in \nu_{\mu_i})}{\delta \beta} = \\ &= \frac{\delta \sum_{i=1}^{n} \ln(\frac{1}{2}(\text{erf}(\frac{a_{\mu_i+1} - f(x_i,\beta)}{\sqrt{2}\sigma}) - \text{erf}(\frac{a_{\mu_i} - f(x_i,\beta)}{\sqrt{2}\sigma})))}{\delta \beta} = \\ &= \sum_{i=1}^{n} \Bigg((1 - (\delta_{\mu_i 0} + \delta_{\mu_i k-1})) \frac{(\text{erf'}(\frac{a_{\mu_i+1} - f(x_i,\beta)}{\sqrt{2}\sigma}) - \text{erf'}(\frac{a_{\mu_i} - f(x_i,\beta)}{\sqrt{2}\sigma}))}{(\text{erf}(\frac{a_{\mu_i+1} - f(x_i,\beta)}{\sqrt{2}\sigma}) - \text{erf}(\frac{a_{\mu_i} - f(x_i,\beta)}{\sqrt{2}\sigma}))} + \\ &\quad + (\delta_{\mu_i 0} + \delta_{\mu_i k-1}) \frac{\text{erf'}(\frac{a_{\mu_i} - f(x_i,\beta)}{\sqrt{2}\sigma})}{(1 + \text{erf}(\frac{a_{\mu_i} - f(x_i,\beta)}{\sqrt{2}\sigma}))} \Bigg) (-1) \frac{\delta f(x_i, \beta)}{\delta \beta}) = \quad (10) \end{aligned}$$

$$= -\sum_{i=1}^{n}\begin{pmatrix}1\\x_{i1}\\\dots\\x_{in}\end{pmatrix}\times\Big((1-(\delta_{\mu_i0}+\delta_{\mu_ik-1}))\frac{(\text{erf'}(\frac{a_{\mu_i+1}-f(x_i,\beta)}{\sqrt{2}\sigma})-\text{erf'}(\frac{a_{\mu_i}-f(x_i,\beta)}{\sqrt{2}\sigma}))}{(\text{erf}(\frac{a_{\mu_i+1}-f(x_i,\beta)}{\sqrt{2}\sigma})-\text{erf}(\frac{a_{\mu_i}-f(x_i,\beta)}{\sqrt{2}\sigma}))}+$$

$$+(\delta_{\mu_i0}+\delta_{\mu_ik-1})\frac{\text{erf'}(\frac{a_{\mu_i}-f(x_i,\beta)}{\sqrt{2}\sigma})}{(1+\text{erf}(\frac{a_{\mu_i}-f(x_i,\beta)}{\sqrt{2}\sigma}))}\Big).$$

δ_{ij} - символ Кронекера.

Уравнение (9) решалось методом секущих.

1 Изучение материала

В ходе выполнения преддипломной практики были изучены следующие источники, в которых описывались методы прогнозирования интервальных данных:

1. Linear regression analysis for interval-valued data based on the Lasso technique [9];
2. Interval linear regression methods based on minkowski difference. [10]

В источнике 2 предлагаются методы линейной регрессии, основанные на метрике Минковского.

В источнике 1 для каждого x_{ij} имеется середина и радиус интервала, в котором он лежит. Соответственно, каждый y_i определяется серединой некоторого интервала и его радиусом. То есть:

$$y_{i_M} = b_{M_0} + b_{M_1} x_{M_{i1}} + \cdots + b_{M_n} x_{M_{in}} + \varepsilon_{M_i} \tag{11}$$

$$y_{i_R} = b_{R_0} + b_{R_1} x_{R_{i1}} + \cdots + b_{R_n} x_{R_{in}} + \varepsilon_{R_i} \tag{12}$$

Решается задача:

$$\min_{b_{M_0},\ldots,b_{M_n},b_{R_0},\ldots,b_{R_n}} \sum_{i=1}^{N} [(\varepsilon_{M_i})^2 + (\varepsilon_{R_i})^2] \tag{13}$$

2 Реализация оценок

Пусть регрессоры нам известны точно. Рассмотрим *метод наименьших квадратов по центрам интервалов.* Метод заключается в следующем: пусть имеется μ_i - номер полуинтервала, в который попало очередное наблюдение y_i. Ему соответствует полуинтервал ν_{μ_i} (см ф.6), т.е. полуинтервал:

$$(a_{\nu_{\mu_i}}, a_{\nu_{\mu_i}+1}], \tag{14}$$

(считаем что $a_1 < y_i < a_{k-1}, i = \overline{1, n}$).

Найдем центральную точку этого интервала, т.е. точку

$$\check{y}_i = \frac{a_{\nu_{\mu_i}} + a_{\nu_{\mu_i}+1}}{2} \tag{15}$$

Построим для всех значений функции регрессии y_i значения $\check{y}_i$. Будем использовать в качестве значений функции регрессии полученные значений, а в качестве регрессоров x_i и построим МНК оценки параметров β.

Описанный метод наименьших квадратов по центрам интервалов был построен путем наследования от исходных оценок и переопределения соответствуюего метода fit().

3 Компьютерные эксперименты

3.1 Параметры модели и оценок

В ходе экспериментов использовались следующие параметры модели:

Параметры программы	
Переменная	значение
Размер выборки N	1000
Доля выбросов $\widetilde{\varepsilon}$	0.8
Параметры регрессии β	$(90, 4)$
Регрессоры x_i	$\sim U(-5, 5)$
ε_i	$\sim N(0, 16)$
η_i	$\sim N(100, 100)$
Величина K из пункта 2.3 курсового проекта	10

,

3.2 Сравнительный анализ построенной оценки с альтернативной

Если сравнить вариации оценок построенные на рис.2, можно увидеть, что оценки, построенные по методу, предложенному в курсовом проекте, показывают лучшие результаты

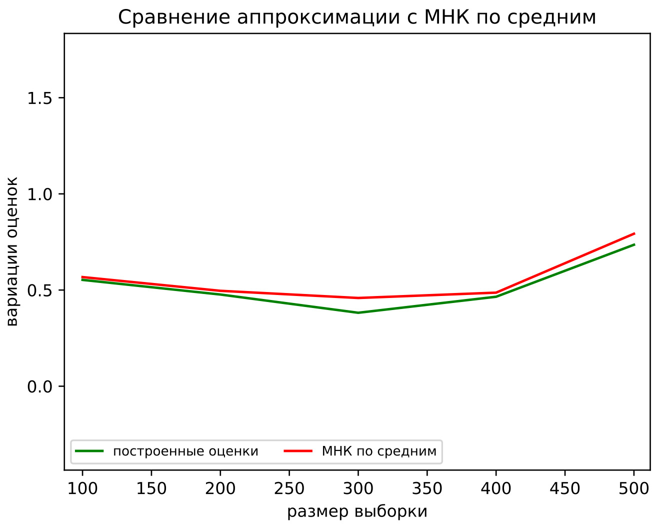

Рис. 1: Сравнение вариаций оценок

3.3 Дополнительные эксперименты

3.3.1 Эксперименты с переклассификацией выборки

В построенном методе использовался метод K-соседей.

На первом этапе для каждого x_i имели класс μ_i: т.е. пару (x_i, μ_i). Далее пытались переклассифицировать выборку. Для этого строилась новую выборка такого же объема N. Проходились по каждому элементу (x_i, μ_i) выборки и для этого наблюдения строилось новое:

$$(x_i, \check{\mu}_i), \tag{16}$$

где $\check{\mu}_i$ получен по методу K-соседей.

$$\check{\mu}_i = \arg\max_j \sum_{k \in V_i,\ k \neq i} \delta_{\check{\mu}_k j}\ , \tag{17}$$

где V_i множество индексов l первых K векторов x_l, отсортированных по возрастанию расстояния до вектора x_i.

После переклассификации выборки, применяли к ней функцию правдоподобия из уравнений (7-8), только теперь с использованием новых классов $\check{\mu}_i$ вместо μ_i. Аналогично максимизировали ее и находили новую оценку параметров $\hat{\beta}$.

В ходе преддипломной практики были построены эксперименты с изменением величины К для метода K-ближайших соседей, используемого в переклассификации. Параметры использовались такие же, как в ранее приведенной таблице.

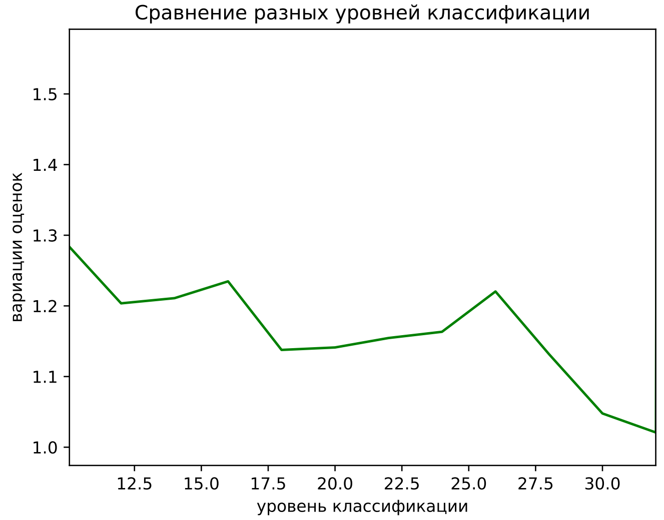

Рис. 2: Зависимость от K, упомянотого в пункте 2.3 курсового проекта

В результате получилось, что при увеличении константы K точность оценки параметров растёт. Но в ходе экспериментов оказалось, что нельзя делать константу K сильно большой: в противном случае точность аппроксимации падает.

Были проведены эксперименты, где использовалась вышеописанная переклассификация и когда нет. При этом на каждой итерации выборка увеличивалась.

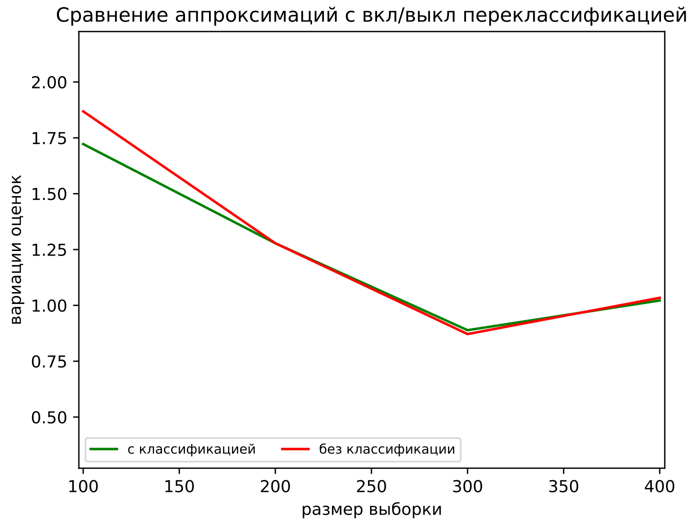

Рис. 3: Сравнение вариаций оценок когда используется и не используется переклассификация

3.3.2 Использование полиномиальной регрессии

Введем теперь модель полиномиальной регрессии.

$$
\begin{gathered}
y_i = \beta_0 + \beta_1 x_{i1}^1 + \beta_2 x_{i2}^2 + \dots + \beta_n x_{in}^n + \varepsilon_i, i = \overline{1, N}, \\
y_i = \sum_{l=1}^{n} x_{il}^{l-1} + \varepsilon_i, i = \overline{1, N}, \\
y_i = f(x_i, \beta) + \varepsilon_i, \\
f(x_i, \beta) = \beta_0 + \beta_1 x_{i1}^1 + \beta_2 x_{i2}^2 + \dots + \beta_n x_{in}^n
\end{gathered}
\tag{18}
$$

Построенные по формуле (18) y_i также как и в случае линейной регрессии будем использовать в формуле (1):

$$
y_i^{\widetilde{\varepsilon}} = (\xi_i) y_i + (1 - \xi_i)\eta_i, \tag{19}
$$

Несложно заметить, что построенные в курсовом проекте оценки никак не зависят от регрессоров, они выступают лишь как параметры, поэтому можно моделировать полиномиальную регрессию и применить к ней описанный метод.

Были построены графики, схожие с рис.1. В итоге получился такой график:

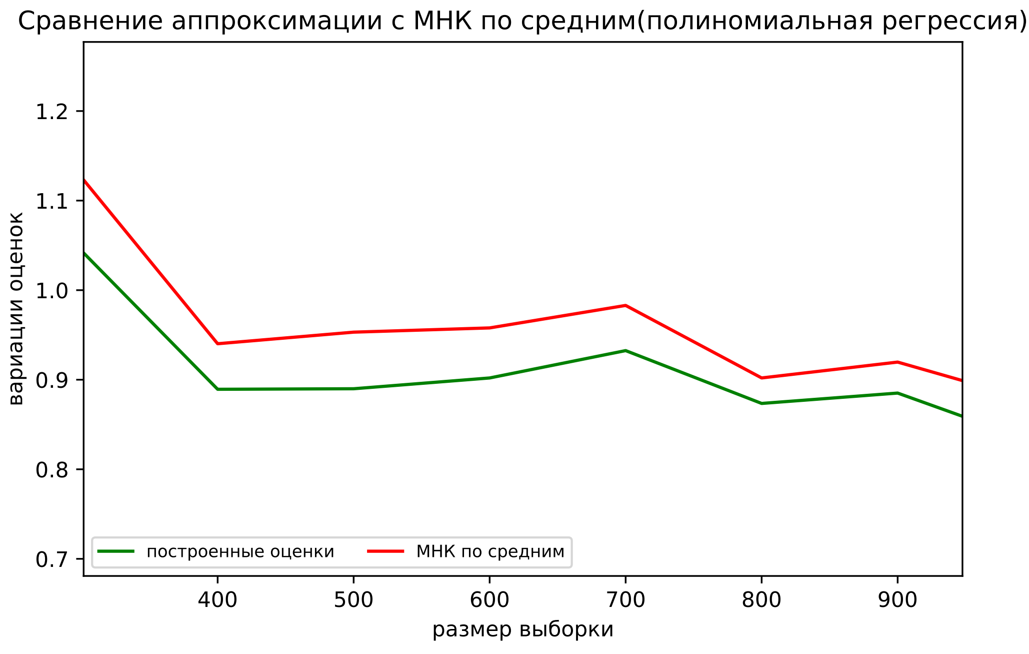

Рис. 4: Аппроксимация параметров в случае полиномиальной регрессии

Видим, что обе модели имеют схожее поведение при изменении объема выборки, но построенные новые оценки стабильно показывают лучший результат.

Заключение

В ходе преддипломной практики был проведен аналитический обзор литературы методов статистического анализа данных при наличии классифицированных наблюдений с искажениями. В результате был реализован альтернативный метод - *метод наименьших квадратов по центрам интервалов*.

Был проведен сравнительный анализ альтернативного метода с построенным в ходе курсового проекта. Были построены дополнительные графики: рис.1, рис.4. Оценки максимального правдоподобия с переклассификацией выборки показали лучшие результаты.

Над оценками максимального правдоподобия с переклассификацией выборки были осуществлены эксперименты, в которых изменялась константа K для метода $K-$ соседей (см. п. 3.3.1).

Реализованные методы были обобщены на случай полиномиальной регрессии.

По проведенным экспериментам видно, что оценки показывают не хуже результаты, чем альтернативные оценки. Можно добиться более точных результатов аппроксимации, если хорошо подобрать параметры оценок.

Список литературы

[1] Хьюбер Дж П. *Робастность в статистике:пер. с англ.* – М.:Мир, 1984.-304 с.

[2] Харин Ю.С., Зуев Н.М., Жук Е.Е. *Теория вероятностей, математическая и прикладная статистика: учебник* – Минск: БГУ, 2011.-463 с.

[3] Е. С Агеева, чл.-корр. НАН Беларуси Ю.С. Харин *Состоятельность оценки максимального правдопобия параметров множественной регрессии по классифицированным наблюдениям*

[4] John Fox, Sanford Weisberg *Robust Regression* – October 8, 2013

[5] А.В. Омельченко *Робастное оценивание параметров полиномиальной регрессии второго порядка* – Харьковский национальный университет радиоэлектроники, Украина, 2009

[6] Özlem Gürünlü Alma *Comparison of Robust Regression Methods in Linear Regression* – Int. J. Contemp. Math. Sciences, Vol. 6, 2011, no. 9, 409 - 421 с.

[7] Sergei Winitzki *A handy approximation for the error function and its inverse.*

[8] Мандрик П.А., Репников В.И., Фалейчик Б.В., *Численные методы* [Электронный ресурс].

[9] Paolo Giordani *Linear regression analysis for interval-valued data based on the Lasso technique* – Department of Statistical Sciences Sapienza University of Rome

[10] Masahiro Inuiguchi, Tetsuzo Tanino, *interval linear regression methods based on minkowski difference a bridge between traditional and interval linear regression models.* – KYBERNETIKA, volume 42, 2006 , number 4, pages 423 - 440

Приложение

Метод наименьших квадратов по центрам интервалов

```
class ApproximationGEMModelNaive(ApproximationGEMModelRedesigned):
    def fit(self):
        self.classify()

        def ex_generator(mu_data):
            for i in range(0, self.endogen.size):
                if mu_data[i] is None:
                    continue
                a_mu_i_plus_1 = mu_data[i] * Defines.INTERVAL_LENGTH
                a_mu_i = mu_data[i] * Defines.INTERVAL_LENGTH - Defines.INTERVAL_LENGTH
                yield (a_mu_i_plus_1 + a_mu_i) / 2

        naive_ex_data_positive = np.fromiter(ex_generator(self._np_freq_positive), float)
        naive_ex_data_negative = np.fromiter(ex_generator(self._np_freq_negative), float)

        naive_ex_data_full = np.append(naive_ex_data_positive, naive_ex_data_negative)

        z, resid, rank, sigma = np.linalg.lstsq(self.exogen, naive_ex_data_full, rcond=None)
        return z
```

Моделирование полиномиальной регрессии:

```
def modulate_polynomial_regression(regression_sample_quintity, regression_outlier_percentage):
    regression_parameters = ACCURATE_RESULT
    _x_points = np.zeros(shape=[regression_sample_quintity, len(regression_parameters)])
    _y_points = np.zeros(shape=regression_sample_quintity)

    def np_random_polynomial(size):
        _res = np.zeros(size)
        for i in range(0, size):
            _res[i] = random.uniform(-5, 5) ** (i + 1)

        return _res

    for i in range(0, regression_sample_quintity):
        _x_points[i] = np.append(np.ones(1), np_random_polynomial(len(ACCURATE_RESULT) - 1))
        if random.random() > regression_outlier_percentage / 100:
            _y_points[i] = (_x_points[i] * ACCURATE_RESULT) + np.random.normal(0, 4)
        else:
            _y_points[i] = np.random.normal(100.0, 15.0, size=1)

    return _x_points, _y_points
```

Моделирование линейной регрессии:

```
def modulateRegression(regression_sample_quintity, regression_outlier_percentage):
regression_parameters = ACCURATE_RESULT
_x_points = np.zeros(shape=[regression_sample_quintity, len(regression_parameters)])
_y_points = np.zeros(shape=regression_sample_quintity)

for i in range(0, regression_sample_quintity):
    if random.random() > regression_outlier_percentage / 100:
        _x_points[i] = np.append(np.ones(1), np.random.uniform(-5, 5, size=len(regression_parameters) - 1))
        _y_points[i] = (_x_points[i] * regression_parameters) + np.random.normal(0, 4)
    else:
        _x_points[i] = np.append(np.ones(1), np.random.uniform(-5, 5, size=len(regression_parameters) - 1))
        _y_points[i] = np.random.normal(100.0, 15.0, size=1)

return _x_points, _y_points
```

Метод наименьших квадратов по центрам интервалов:

```
def fit_data_naive_classic():
    sample_sizes = []
    all_results_classic = []
    all_results_naive = []
    for sample_size in range(SAMPLE_SIZE_MIN, SAMPLE_SIZE_MAX+1, SAMPLE_SIZE_STEP):
```

```
        successful_fit = False
        while not successful_fit:
            x_points, y_points = modulateRegression(sample_size, OUTLIER_PERCENTAGE)
            approx_model = groupingEstimates.GEM(x_points, y_points)
            approx_model_naive = groupingEstimatesNaive.GEM_N(x_points, y_points)
            try:
                result = approx_model.fit()
                print("GEM {}".format(result))
                result_naive = approx_model_naive.fit()
                print("GEM_N {}".format(result_naive))

                successful_fit = True

                all_results_classic.append(result)
                all_results_naive.append(result_naive)
                sample_sizes.append(sample_size)
            except KeyboardInterrupt:
                print("stopping...")
                np.save(NP_DATA_PATH + "gem_res_classic", all_results_classic)
                np.save(NP_DATA_PATH + "gem_res_naive", all_results_naive)
                np.save(NP_DATA_PATH + "gem_sizes", sample_sizes)
                quit()
            except Exception as e:
                print(e)
    np.save(NP_DATA_PATH + "gem_res_classic", all_results_classic)
    np.save(NP_DATA_PATH + "gem_res_naive", all_results_naive)
    np.save(NP_DATA_PATH + "gem_sizes", sample_sizes)
```

График с разным объемом выборки:

```
def plot_with_different_sample_size():
    sample_sizes = []
    all_results_with_classification = []
    all_results_without_classification = []

    x_points = None
    y_points = None

    for sample_size in range(SAMPLE_SIZE_MIN, SAMPLE_SIZE_MAX+1, SAMPLE_SIZE_STEP):
        successful_fit = False
        while not successful_fit:
            x_points_t, y_points_t = modulateRegression(sample_size, OUTLIER_PERCENTAGE)

            if x_points is None or y_points is None:
                x_points = x_points_t
                y_points = y_points_t
            else:
                x_points = np.append(x_points, x_points_t, axis=0)
                y_points = np.append(y_points, y_points_t, axis=0)

            approx_model = groupingEstimates.GEM(x_points, y_points)
            try:
                result = approx_model.fit()
                print("GEM {}".format(result))
                result_without = approx_model.fit_without_reclassification()
                print("GEM_without {}".format(result_without))

                successful_fit = True

                all_results_with_classification.append(result)
                all_results_without_classification.append(result_without)
                sample_sizes.append(sample_size)
            except KeyboardInterrupt:
                print("stopping...")
                np.save(NP_DATA_PATH + "gem_res_with", all_results_with_classification)
                np.save(NP_DATA_PATH + "gem_res_without", all_results_without_classification)
                np.save(NP_DATA_PATH + "gem_sizes_with_without", sample_sizes)
                quit()
            except Exception as e:
                print(e)
    np.save(NP_DATA_PATH + "gem_res_with", all_results_with_classification)
    np.save(NP_DATA_PATH + "gem_res_without", all_results_without_classification)
    np.save(NP_DATA_PATH + "gem_sizes_with_without", sample_sizes)
```

График с разным уровнем переклассификации:

```
def plot_with_different_reclassification_level():
    reclassification_levels = []
    all_results_with_classification = []
    recl_level_min = 10
    recl_level_max = 40

    x_points, y_points = modulateRegression(500, OUTLIER_PERCENTAGE)

    for recl_level in range(recl_level_min, recl_level_max + 1, 2):
        GroupingEstimatesDefines.RECLASSIFICATION_LEVEL = recl_level

        successful_fit = False
        while not successful_fit:
            approx_model = groupingEstimates.GEM(x_points, y_points)
            try:
                result = approx_model.fit()
                print("GEM {}".format(result))

                successful_fit = True

                all_results_with_classification.append(result)
                reclassification_levels.append(recl_level)
            except KeyboardInterrupt:
                print("stopping...")
                np.save(NP_DATA_PATH + "gem_with_dif_level_results", all_results_with_classification)
                np.save(NP_DATA_PATH + "gem_with_dif_level_levels", reclassification_levels)
                quit()
            except Exception as e:
                print(e)
    np.save(NP_DATA_PATH + "gem_with_dif_level_results", all_results_with_classification)
    np.save(NP_DATA_PATH + "gem_with_dif_level_levels", reclassification_levels)
```