Программа повышения конкурентоспособности

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет имени
первого Президента России Б. Н. Ельцина»

ОБРАБОТКА ВРЕМЕННЫХ РЯДОВ НА ОСНОВЕ МНОГОМЕРНЫХ ДАННЫХ

Методические указания к выполнению
лабораторной работы № 4

Екатеринбург
2019

Содержание

1. Введение

Определение временных рядов подразумевает наличие некоторых отсчетов/наблюдений на фиксированной временной сетке. Но что, если временной отсчет имеет такую продолжительную длину (например, сутки или месяц), что в данной точке может фиксироваться сразу несколько наблюдений фиксированного параметра? Тогда эти данные будут являться многомерными, но при этом отвечать всем требованиям анализа и прогнозирования временных рядов. При этом большинство методов анализа ВР описывается в строгой форме для одномерных зависимостей наблюдений от времени. Есть два способа решения этой проблемы. Во-первых, методы построения регрессионных моделей свободны от необходимости одномерности, но при этом страдают от необходимости однозначности наблюдений от каждого отдельного фактора. Другой подход – сведение многомерных данных к главной характеризующей одномерной последовательности от времени. Именно последний метод описан в данной лабораторной работе.

2. Задание на лабораторную работу

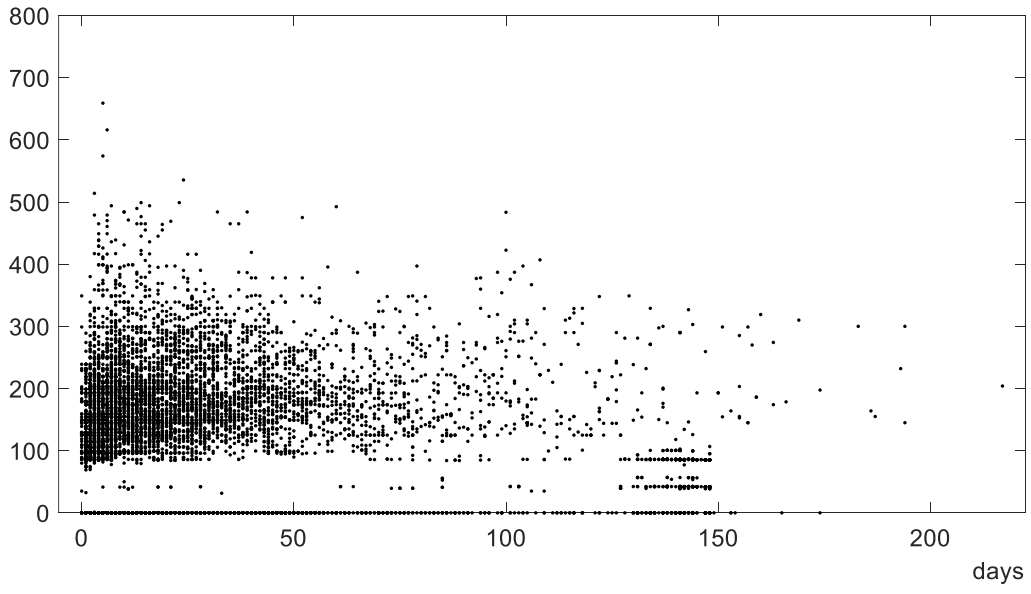
Результатом выполнения лабораторной работы является оформленный отчет в виде *Jupyter*-тетради, в котором должны быть представлены и отражены все нижеперечисленные пункты:

1) Исходные данные содержатся в файле «airline_flights_data_ord.xlsx». Эта таблица содержит цены для некоторого авиаперевозчика на авиабилеты на одно направление в зависимости от множества параметров (различные дополнительные услуги и сроки), отсортированные по дате заказа.

   Главное задание: требуется построить одномерный временной ряд, отражающий зависимость итоговой стоимости билетов (***Ticket cost***) от числа дней между датами вылета и заказа (**{*Flight Date - Sale Data*}** в днях).

   Часто подобную задачу называют усредненным прогнозом цен на авиабилеты от числа предварительных дней между их датами покупки и вылета.

2) Загрузите данные и постройте точки стоимости билетов от числа дней между датами вылета и заказа (не забудьте их отмасштабировать):

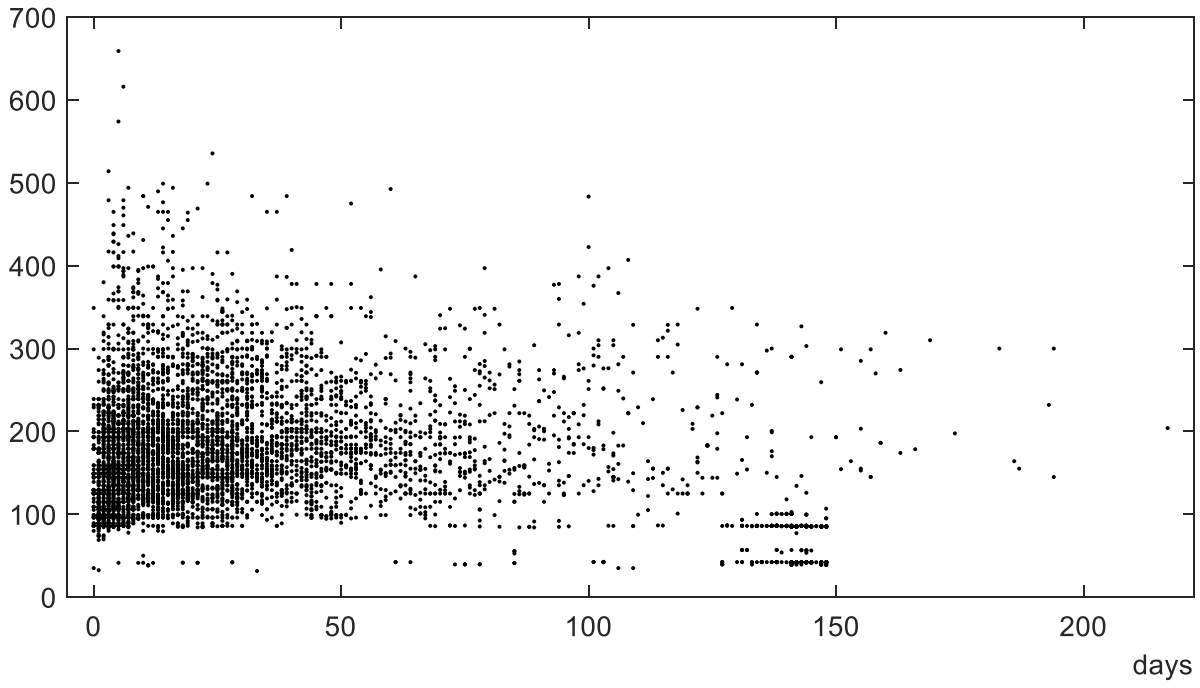


3) Из рисунка легко заметить, что за один временной отсчет (день) имеется множество наблюдений. А еще более заметно – что данные требуют **предобработки**, так как они содержат нули (отмененные заказы, например), Not-a-Number (ошибки системы заказа) и аномальные отсчеты (слишком «подозрительно» дешевые и слишком дорогие билеты, одиночные закупки и т.д.).

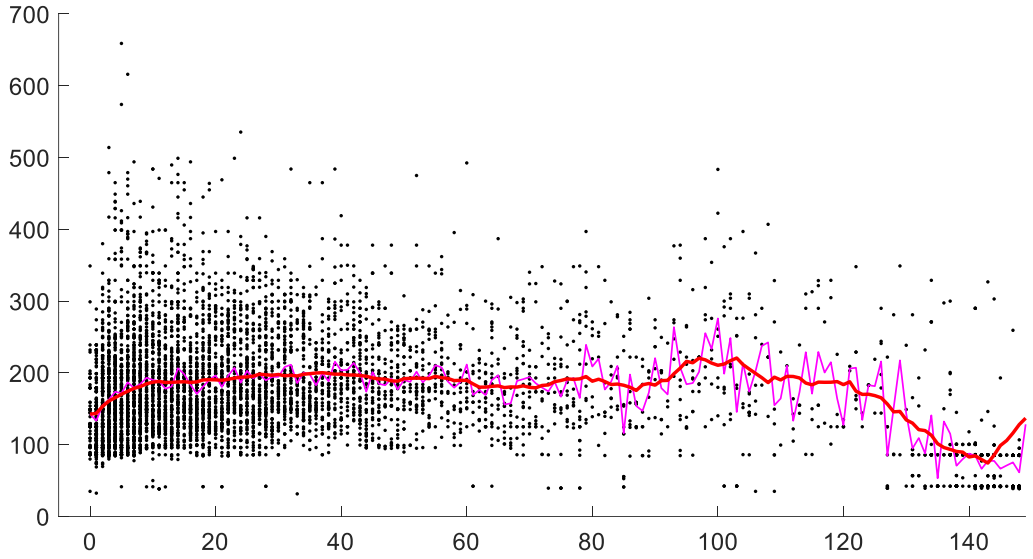
Удалите следующие точки:

- все билеты, для которых столбец *Sum* равен 0 (отмененные заказы);
- все билеты, стоимость которых выше 1000$ (слишком дорого);
- все билеты, стоимость которых ниже 30$ (подозрительно дешево);
- а еще точки с **NaN** и билеты, которые купили очень мало людей в определенный срок (например, когда число дней между датами вылета и заказа **{*Flight Date - Sale Data*}** равно 111 и 124, где зафиксирована всего одна непрезентативная покупка), так как они тоже являются аномалиями для данной многомерной выборки.

Получится примерно следующий график:

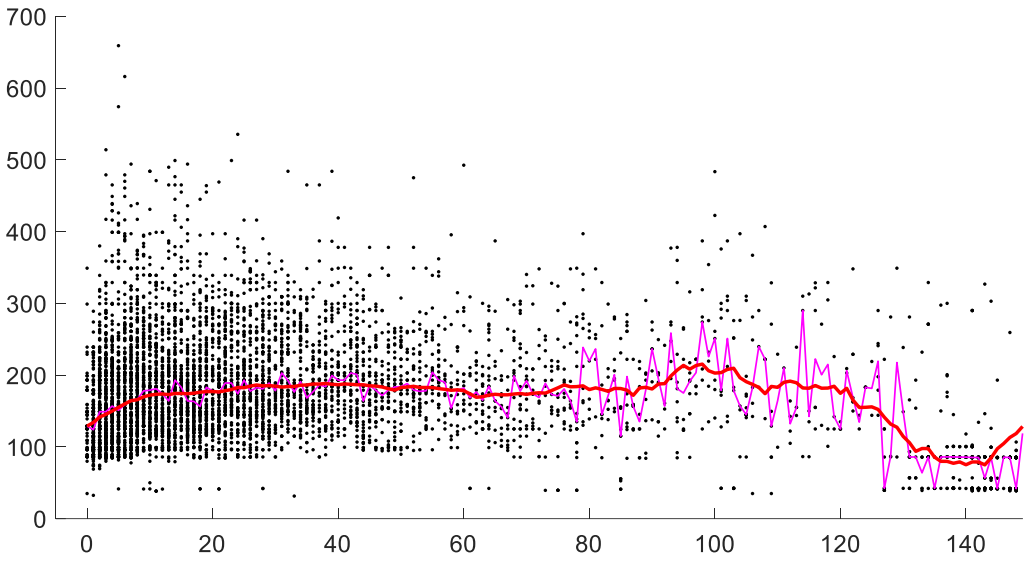


4) Для простоты дальнейших вычислений **удалите** также точки стоимости билетов со сроками покупки **больше 148 дней**, так как после этого срока точки покупок встречаются очень редко и их можно считать выбивающимися из общей зависимости (недостаточными для статистики).
5) Теперь попробуем построить из этого множества точек одномерный временной ряд стоимости билетов от числа предварительных дней до покупки. Первое, что можно сделать – это просто посчитать среднее значение стоимости билетов для каждого дня, то есть посчитать функцию **mean()** от каждого массива стоимости билетов в каждой временной точке. Вот что примерно должно у Вас получиться:



Здесь розовой кривой построена зависимость средней стоимости билетов от дней, красной – сглаженная/усредненная кривая (постройте эту кривую методами скользящего среднего сглаживания или подгоночными кривыми/сплайнами).

6) Для первичной оценки – неплохо, но здесь есть 3 проблемы. Во-первых, стоимость билетов принимает некоторые явные значения из множества, нежели «абстрактные» средние величины, поэтому более верным будет построить график расчета медианы **median()** от времени:



Во-вторых, функция среднего значения есть дискретная форма мат. ожидания только для стационарных и эргодических случайных величин (здесь случайная величина = стоимость билетов). Ни то ни другое условие здесь явно не выполняется, а значит построенные кривые демонстрируют стоимость билетов от числа предварительных дней, как если бы стоимость их была бы равномерно разбросана в каждом временном отсчете (это явно не так).

В-третьих, нас интересует усредненный прогноз цен на авиабилеты от числа предварительных дней между их датами покупки и вылета, а у прогноза еще есть такое понятие, как *доверительный интервал*. То есть, кроме средних цен на билеты нас интересует в каком диапазоне цены разбросаны вообще, а этого простым вычислением среднего уже не посчитать. Вообще-то, есть еще дисперсия или СКВО, но см. проблему №2 выше. **Постройте** подобные оценочные интервалы (среднее ± 1.65***std()**, со сглаживанием) самостоятельно.

7) Для решения всех указанных проблем используется более научная методика. Пусть каждый набор точек (стоимости билетов) в каждом временном отсчете (число дней) есть выборка некоторой **случайной величины**. Если это случайная величина – у нее есть функция распределения (плотность), благодаря которой можно оценить ее мат. ожидание = 50%-перцентиль (средняя стоимость билета) и доверительные интервалы = 5%-перцентиль и 95%-перцентиль (разброс стоимости билетов по дням).

   Значит, самое главное – построить плотность функции распределения для этой неизвестной случайной величины стоимости билета. Для этого в Python есть библиотека **scipy.stats**:

   import scipy.stats as st

   и ее функции **distribution.fit(data)** и **distribution.pdf(x)**.

8) Проблема в том, что переменную **distribution** требуется заранее определить, как одну из известных распределений для **scipy.stats** из вот этого множества:

   DISTRIBUTIONS=[st.alpha, st.anglit, st.arcsine, st.beta, st.betaprime, st.bradford, st.burr, st.cauchy, st.chi, st.chi2, st.cosine, st.dgamma, st.dweibull, st.erlang, st.expon, st.exponnorm, st.exponweib, st.exponpow, st.f, st.fatiguelife, st.fisk, st.foldcauchy, st.foldnorm, st.frechet_r, st.frechet_l, st.genlogistic, st.genpareto, st.gennorm, st.genexpon, st.genextreme, st.gausshyper, st.gamma, st.gengamma, st.genhalflogistic, st.gilbrat, st.gompertz, st.gumbel_r, st.gumbel_l, st.halfcauchy, st.halflogistic, st.halfnorm, st.halfgennorm, st.hypsecant, st.invgamma, st.invgauss, st.invweibull, st.johnsonsb, st.johnsonsu,

st.ksone, st.kstwobign, st.laplace, st.levy, st.levy_l, st.levy_stable, st.logistic, st.loggamma, st.loglaplace, st.lognorm, st.lomax, st.maxwell, st.mielke, st.nakagami, st.ncx2, st.ncf, st.nct, st.norm, st.pareto, st.pearson3, st.powerlaw, st.powerlognorm, st.powernorm, st.rdist, st.reciprocal, st.rayleigh, st.rice, st.recipinvgauss, st.semicircular, st.t, st.triang, st.truncexpon, st.truncnorm, st.tukeylambda, st.uniform, st.vonmises, st.vonmises_line, st.wald, st.weibull_min, st.weibull_max, st.wrapcauchy]

Неплохой такой список для выбора…

А главная проблема в том, что каждое распределение имеет собственный набор параметров, заранее нам неизвестных.

9) Но сначала попробуем взять самое известное распределение – *нормальное*, и для него все оценить:

distribution= st.norm

10) Его параметрами являются мат. ожидание и сигма (корень из дисперсии), которые и вернет функция **fit()**:

params = distribution.fit(data)

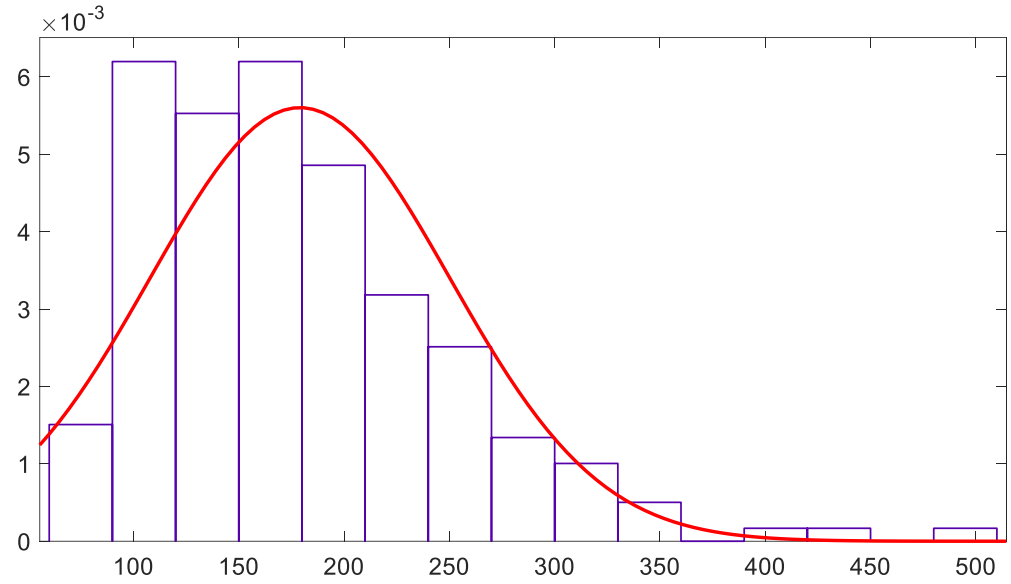
11) Для начала попробуйте сопоставить гистограмму стоимости билетов, например, для разности дат = 7 дней, с плотностью нормального распределения с найденными параметрами:

y, x = np.histogram(data, bins=…)

pdf_fitted = dist.pdf(x, *param[:-2], loc=param[-2], scale=param[-1])

и постройте гистограмму и полученную кривую на одном рисунке.

Должно получиться что-то похожее на рисунок ниже:

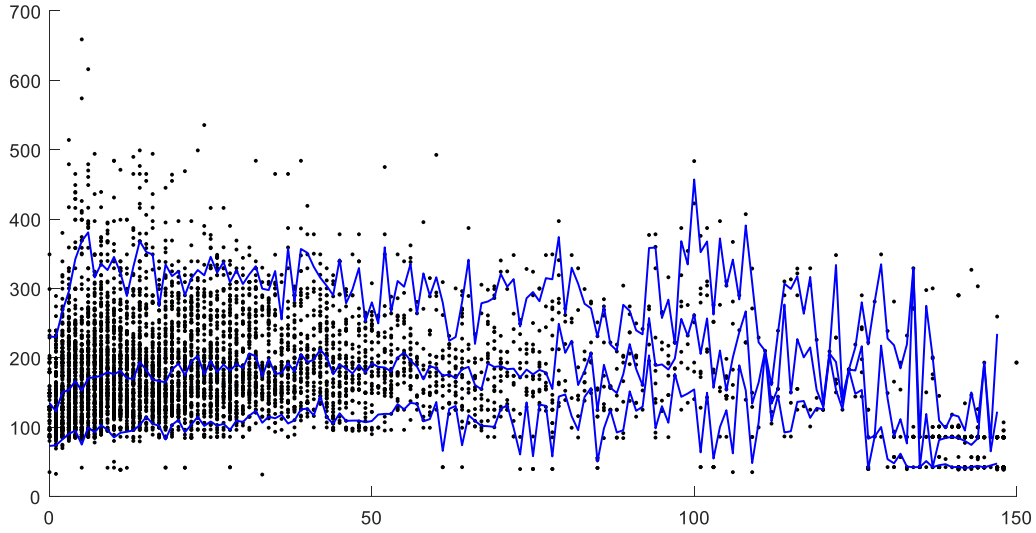


12) Аналогичным образом найдите функции плотности нормального распределения для всех временных точек (дней), но без построения гистограмм/кривых. Возможно, не все точки в принципе возможно подогнать под нормальное распределение, и тогда Python будет выдавать предупреждения (*warning*), но пока что их можно игнорировать.

13) Чтобы рассчитать значение перцентилей для найденных распределений в *Scipy* есть функция **ppf()**:

```
# 5%-перцентиль (нижний доверительный интервал)
pp05 = dist.ppf(0.05, *param[:-2], loc=param[-2], scale=param[-1])
# 50%-перцентиль (средний прогноз)
pp50 = dist.ppf(0.50, *param[:-2], loc=param[-2], scale=param[-1])
# 95%-перцентиль (верхний доверительный интервал)
pp95 = dist.ppf(0.95, *param[:-2], loc=param[-2], scale=param[-1])
```

14) Найдите 5%, 50% и 95%-перцентили для нормального распределения для всех заданных временных точек и постройте полученные значения на исходном графике соотношения цен на билеты. Получится отдаленно что-то подобное:



15) Сгладьте полученные кривые.

16) При построении нормального распределения Вы наверняка столкнулись с большим числом ошибок/предупреждений *Python* из-за того, что не ко всякому множеству точек возможно построить нормальное распределение. И в самом деле – если вернуться к рисунку с гистограммой выше, легко заметить, что распределение цен билетов отнюдь не нормальное. Какое распределение тогда нужно взять? Равномерное? Гамма, Бета, Пирсона или другие? Одномодальное или двух-модальное?

17) Есть два способа решения этой проблемы. Во-первых, для каждого временного отсчета перебирать построение распределений из некоторого множества выбранных названий распределений и выбрать то, которое вообще можно посчитать (для начала) и которое наиболее близко к построенной гистограмме (по СКВО ошибки). Но это требует долгих расчетов, перебора и хороших навыков программирования.

18) Второй способ гораздо более эффективен, надежен и грамотен. Все перечисленные распределения выше являются ***параметрическими***, то есть такими, у которых есть параметры: имя и характерные коэффициенты для формулы плотности распределения.

Поэтому для них мы и определяли нужные параметры и подставляли в функцию распределения. Но есть и ***непараметрические методы*** описания случайных величин, где функция плотности распределения есть набор элементарных функций (***ядро*** = ***kernel***) с фиксированными параметрами, наподобие гистограммы, где ядром служило равномерное распределение (прямоугольники).

19) Возьмем самый простой случай, когда ядро имеет Гауссовское нормальное распределение. Для этого есть готовая функция:

```
from scipy.stats import gaussian_kde
```

20) Тогда оценка плотности распределения для некоторого массива ***x*** случайных чисел строится как:

```
kde = gaussian_kde(x.ravel())        # строим оценку
plt.plot(kde.pdf(np.linspace(x.min(),x.max(),x.size)))
# функция распределения в границах случайной величины
```

Заметьте, что выбор параметров здесь вообще отсутствует, отчего задача существенно упрощается.

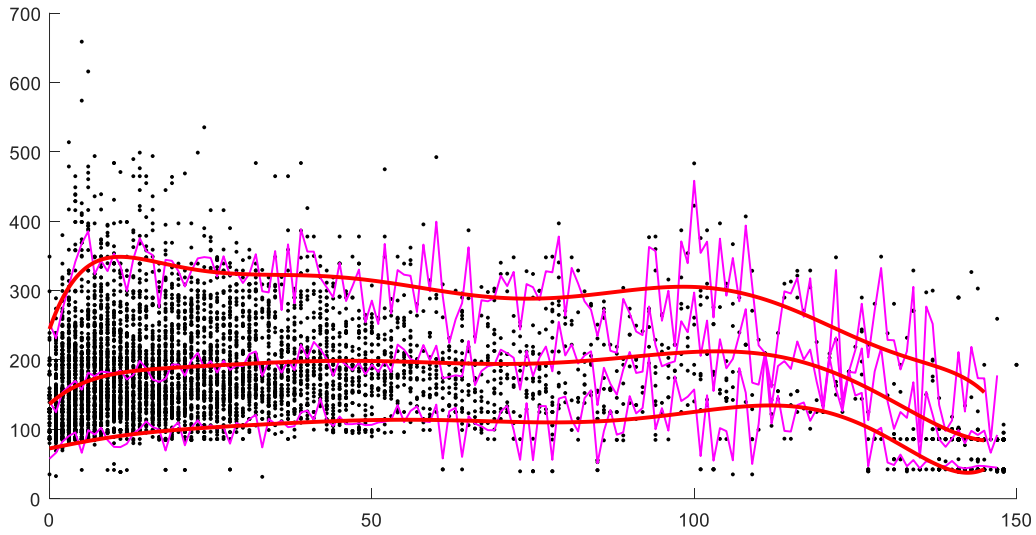
21) Осталось оценить нужные нам перцентили, но, к сожалению, в *Python* нет готовых функций для этих классов. Однако, знания по теории вероятности помогут нам написать эту функцию:

```
def kde_perc(ikde, low, high, n, perc=0.50):
    for i in np.linspace(low, high, n)[1:]:    # от нижней до верхней границы
        F=ikde.integrate_box_1d(low,i)         # интеграл плотности
        if F>perc:                             # чуть выше границы суммы
            return i                           # есть искомое значение
```

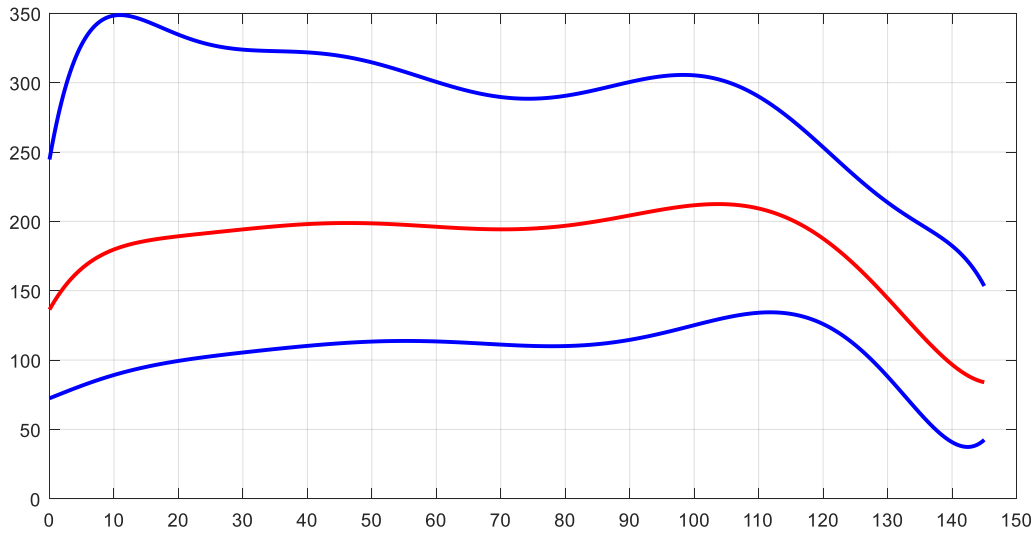
22) Теперь значение перцентилей всегда можно будет посчитать как:

```
kde_perc(kde,x.min(),x.max(),x.size,0.05) # 05%-перцентиль
kde_perc(kde,x.min(),x.max(),x.size,0.50) # 50%-перцентиль
kde_perc(kde,x.min(),x.max(),x.size,0.95) # 95%-перцентиль
```

23) Теперь, если найти непараметрические распределения стоимости билетов для каждой временной точки и затем для них нужные перцентили, то получится график, примерно, как на рисунке ниже:



24) Тогда итоговый усредненный прогноз стоимости авиабилетов от числа предшествующих дней между покупкой и вылетом, вместе с доверительными интервалами, будет выглядеть как:



25) Таким путем можно проанализировать и цены на авиабилеты: чем раньше их заказывать, тем они дешевле (правый край), но есть и дешевые билеты прямо перед вылетом («горящие» билеты, левый край). От 1 до 3 месяцев перед вылетом цены почти в среднем не меняются, за квартал до даты цены немного растут. Самый высокий разброс цен в первые две недели. Все эти характерные особенности легко увидеть из полученного одномерного временного ряда, нежели чем из многомерного анализа исходных таблиц. Наглядность этого решения и является его основным преимуществом.

26) Завершите свой отчет сравнением различных итоговых получившихся кривых и сравнением временных затрат на их расчеты.

3. Требования к оформлению отчета

Отчет в Jupyter-тетради должен обязательно содержать: номер лабораторной работы, ФИО студента, номер варианта (либо студенческий номер), номер группы, результаты выполнения работы с комментариями студента (комментарии пишутся после #) и изображениями.