ВСТУПЛЕНИЕ.

Во время работы на РЖД, в рамках построения системы нейросетевого прогнозирования пассажиропотоков, я разработал и внедрил новый алгоритм прогнозирования случайного процесса - количества проданных билетов (увы, по независящим от меня причинам, проект был остановлен на стадии тестовых испытаний, РЖД вообще отказалась от прогнозирования).

Суть проблемы такова:

На РЖД продаются билеты на поезда, которые очень плохо прогнозировались, в связи с большим количеством шумов. Все первые попытки провести прогноз временными рядами, статистической обработкой и нейросетями на основе самых разных входных полей, в итоге при попытке посуточного прогноза на конкретный тип поезда давали корреляцию не выше 20-25%.

При этом, сама прогнозируемая случайная величина имеет очень жёсткий ограничитель сверху – она почти не может быть больше количества предложенных мест. Точнее, если поезд скорый и с коротким маршрутом, почти без остановок (например СПб – Москва), то неравенство строгое, число пассажиров не больше числа мест. А если поезд дальний, и на место вышедшего пассажира может сесть новый, то ограничитель действует уже нестрого. В случае с пассажирооборотом неравенство всегда строгое.

При этом, цель стояла не в прогнозе именно пассажиров, при некоторых внешних показателях, а именно возможного спроса на билеты. То есть, задача стояла так – надо предсказать, сколько мест можно предложить в продажу (какую сделать составность поезда), чтобы почти все они были выкуплены.

При этом, в реальности данные были таковы, что в разное время года, при самых разных количествах предложенных в продажу мест, по всем направлениям доля поездов, в которых число билетов очень сильно приближалось, или даже превышало ограничитель, составляла порядка 30-60%. То есть, все потенциально максимальные возможные количества пассажиров были сильно срезаны недостатком предложения мест. И именно поэтому качество прогноза всех первых моделей было очень низким.

Для решения этой задачи мною был предложен и реализован алгоритм аппроксимации случайной величины, с учётом знания размера ограничителя. То есть, была построена нейросеть специального типа, которая настраивается в ходе случайных блужданий, на вход получает все те же параметры, что и все прежние нейросети, включая и количество мест в поезде. Отличие от обычной нейросети – размер ограничителя так же специальным образом подаётся и на выход нейросети. Вследствие этого, первые же примеры использования этого алгоритма дали корреляцию прогноза с реальными наблюдениями сразу в 40%, а после полной отладки комплекса на некоторых поездах она достигла 50%. Это позволило сделать на данной модели очень хорошую систему прогнозирования и выдачи рекомендаций.

ТЕХНИЧЕСКАЯ ПОСТАНОВКА ЗАДАЧИ.

Итак, надо доказать, что мой алгоритм не хуже, а в некоторых случаях и существенно лучше разных иных алгоритмов, но это надо именно доказать экспериментально. Для этой цели я воспользовался математическим моделированием.

ДАНО: Программа генерирует несколько последовательностей, с разными параметрами, каждая из которых имеет несколько входных полей (X1-X(k)), и прогнозируемый выходной спрос (SPROS = sum(a(i)\*X(I))+W ), при этом так же случайно формируется ограничитель (OGR), и итоговое значение формируется с учётом ограничителя и краевого эффекта ограничения (Y={SPROS если spros<ogr ; OGR+W в противном случае} ). Дополнительное требование - параметры случайных величин берутся такими, чтобы при прогнозировании линейной комбинацией X(i) без подачи на вход размера ограничителя, корреляция была бы порядка 50%. Таким способом я считаю, что на вход подаётся вся информативная часть реальных данных, а так же множество шумов, не имеющих отношения к делу. В идеале вполне возможно и желательно, чтобы (SPROS = sum(a(i)\*X(I))+W = a1\*X1+W), а все прочие параметры чисто для усложнения процесса настройки.

НАДО: Исследовать мой алгоритм, а так же разные иные, с целью выяснения, какие из них в данном случае лучше или хуже, и по каким параметрам.

РЕШЕНИЕ 1 ЧАСТЬ.

Поиск в интернете дал следующие алгоритмы для прогнозирования случайной величины с ограничителем (в каждом случае кратко описан итог рассмотрения):

1. Мой алгоритм, всюду называется ‘0.My\_progn’
2. Простая линейная модель – она хоть и плохо, но прогнозирует. К тому же, это был первый вариант прогнозирования, рассматривавшийся на РЖД, именно его точность корреляции составляла всего 15-20%. А значит, именно от неё и надо отталкиваться. Всюду указана как ‘1.lin’
3. Обобщенная линейная модель (GLM) с логистической функцией. По сути, это линейное прогнозирование случайной величины Z=Y/OGR с помощью аппроксимации гипертангенсом. Всюду указана как ‘2.GLM’.
4. Нейронная сеть с активационной функцией sigmoid или tanh на выходном слое. По сути, это тоже прогнозирование случайной величины Z=Y/OGR , но уже с помощью аппроксимации многослойной нейросетью. На моём ПК алгоритм работает очень плохо, жутко медленно, гораздо хуже чем написанные мною 15лет назад алгоритмы на SAS, и на порядок хуже чем на R. Всюду указана как ‘*3.SIGMOID*’.
5. Модель машинного обучения с последующей трансформацией, использующая стандартную модель регрессии (например, случайный лес или градиентный бустинг). Была выбрана программа RandomForest (процедура RandomForestRegressor). Этим методом аппроксимация строится достаточно точно, гораздо точнее чем даёт предложенный мною метод. И при этом, данный метод даёт возможность весьма точно спрогнозировать спрос, поставив на входе ограничитель на максимум. Одно плохо – функция ОЧЕНЬ сильно переобучается, и по этой причине отличная аппроксимация на тренировочном множестве даёт чрезвычайно плохой прогноз на тестовом. Может быть что-то можно поделать, копаясь в настройках, но мне было просто не до этого. Всюду указана как ‘4. RandomForest’.
6. Байесовская регрессия с усеченным нормальным распределением. Данный пример вообще не удалось опробовать, потому что на моём слабом ПК нужные программы просто не захотели запускаться. При этом рассмотрение алгоритма показало, что метод рассчитан на прогнозирование при условии постоянства значения ограничителя – а это не наш случай.
7. Квантильная регрессия. Данный метод прогнозирует вообще не саму случайную величину, а то значение, менее которого она окажется с указанной вероятностью. Данный подход мною тоже рассматривался во время работы на РЖД, но в тех условиях результата не принёс. В данном конкретном случае этот алгоритм из сравнения выпадает.
8. Модель временных рядов с ограничением (если данные имеют временную структуру). По сути, это та же самая линейная аппроксимация, только входными параметрами служат итоги обработки временных рядов, и с постоянным ограничителем, так же как и в случае с Байесовской регрессией.

НАПИСАНИЕ ПРОГРАММЫ

Входные данные были сгенерированы программой на PostgreSQL (файл my\_project.sql). А именно, было сгенерировано 100 последовательностей (с номерами TIP=1:100) по 1000 элементов, все кроме первой - со случайными параметрами, близкими к первому базовому варианту. В каждой последовательности есть 2 входа X1, X2, посчитанный по ним полный спрос SPROS, ограничитель OGR, и значение ограниченной величины Y. Созданные данные записаны в таблицу Dannie, в публичной схеме в postgres. Так же рядом создана таблицы Dannie2, в которую скидываются итоги настройки прогнозов.

Для решения вопроса о качестве прогнозирования разными методами были взяты все перечисленные варианты, по ним взяты в интернете типовые программы, и адаптированы для текущей задачи. Задачу прогнозирования всеми указанными алгоритмами решает программа (my\_program.py). В ней исходные данные читаются процедурой из программы (connect.py, в ней же процедура записи в базу), затем обрабатываются всеми возможными вариантами алгоритмов прогнозирования, и записываются в таблицу результатов Dannie2. При этом, программа пользуется стандартными библиотеками, а так же написанной мною на Python программой предлагаемого мною метода прогнозирования (my\_nastroika.py).

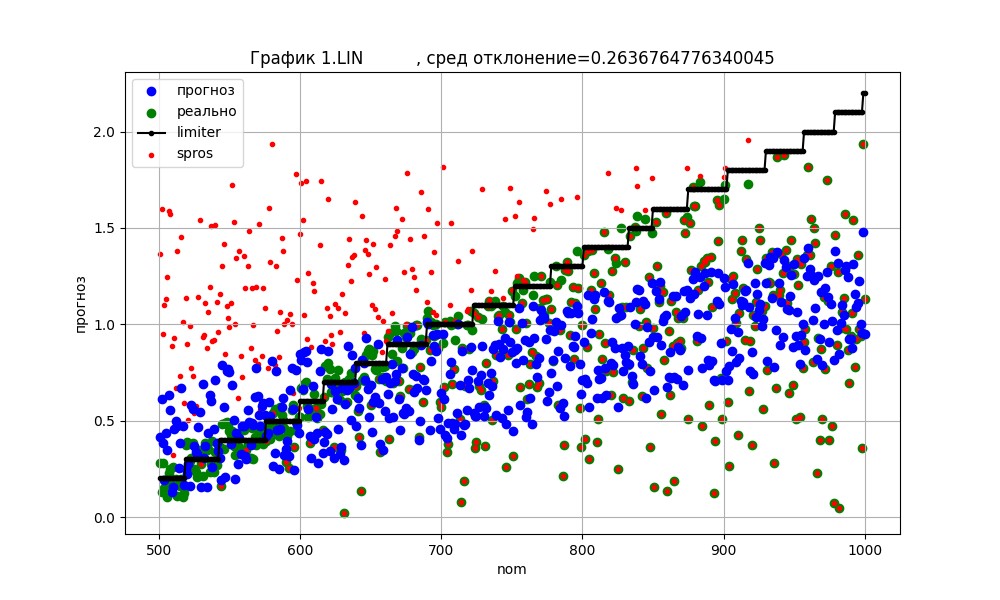
Вторым этапом итоги прогнозирования обрабатываются одинаковой для всех методов программой статобработки (my\_stat.py). (Исходно обработка проводилась в первой программе my\_program.py. но это оказалось в дальнейшем просто неудобно.) С её помощью получены все прилагаемые графики, выдаваемые процедурой из файла (graf.py).

Так как моя программа показала более чем достойное поведение, по сравнению со всеми прочими альтернативами, то в гите из неё изъят центральный блок прогнозирования постобработки прогноза с помощью знания об ограничителе, я не хочу им делиться не имея шансов на трудоустройство, отдам только после собеседования и гарантии получения зарплаты. Из основного скажу лишь одно – программа занимается нелинейным прогнозированием, и потому имеет множество локальных минимумов, в любом из которых может зависнуть. Для нейтрализации этого минуса, программа совершает настройку методом случайных блужданий три раза, и на выход выдаёт только один лучший результат. Но невзирая на тройное использование, программа работает очень быстро.

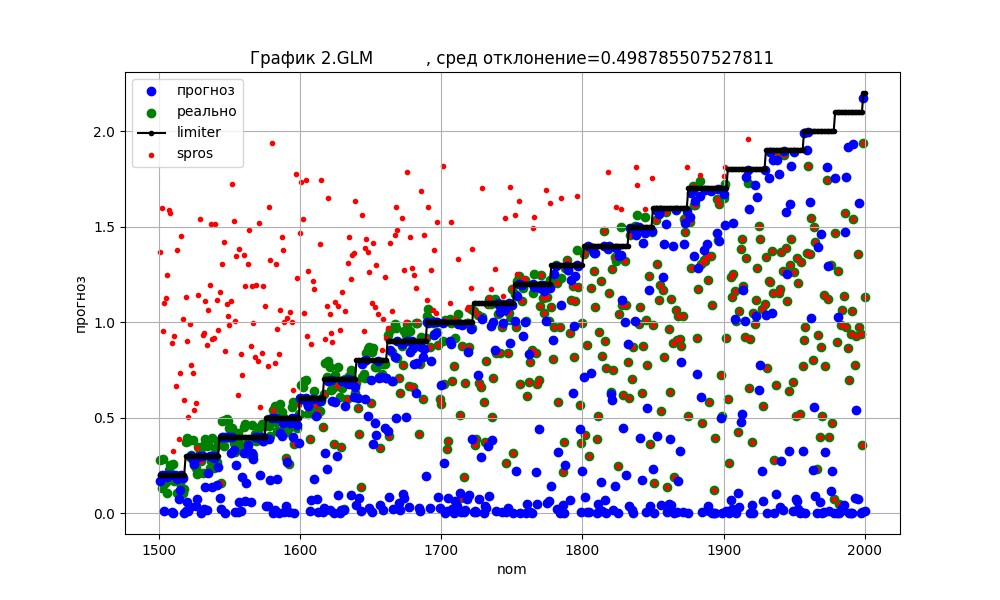
ТИПИЧНЫЕ ГРАФИКИ ПРОГНОЗОВ

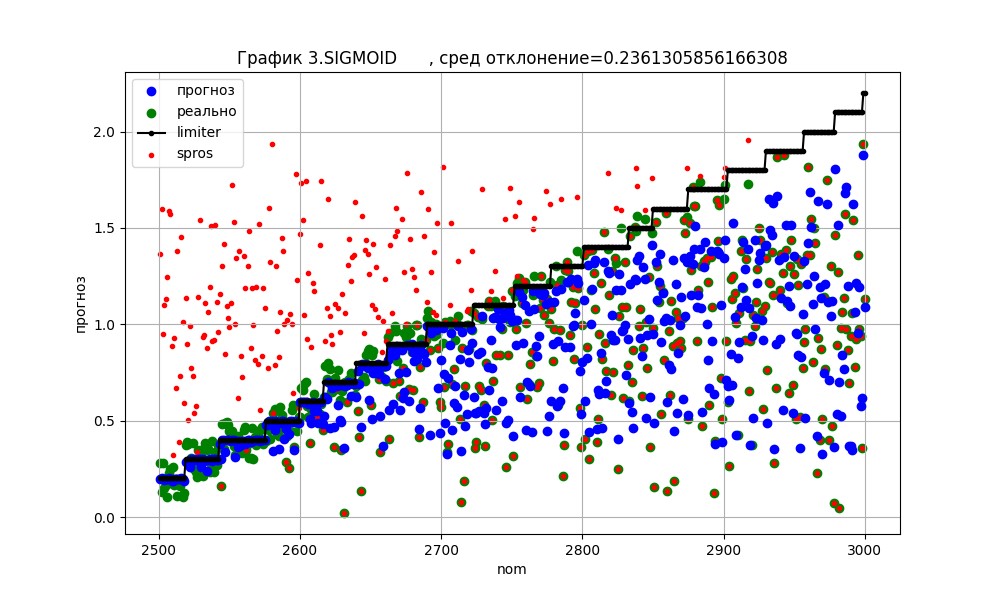
Все графики по всем методам выданы на основе одной и той же сгенерированной серии наблюдений, причём именно по тестовой половине множества строк. В каждой из 100 серий первые 500 наблюдений были отнесены к настроечной, а последующие – к тестовой части. Рандомизация выборки строк на настроечную и тестовую части не производилась, в связи с исходно случайной их генерацией.

Метод линейного прогнозирования даёт самый простейший график прогноза, с наименьшей точностью. Вот так выглядит самый типичный пример – график реально похож на линию, потому что размер исходной случайной величины сильно коррелирует с размером ограничителя, и программа просто берёт его в качестве базиса прогноза.

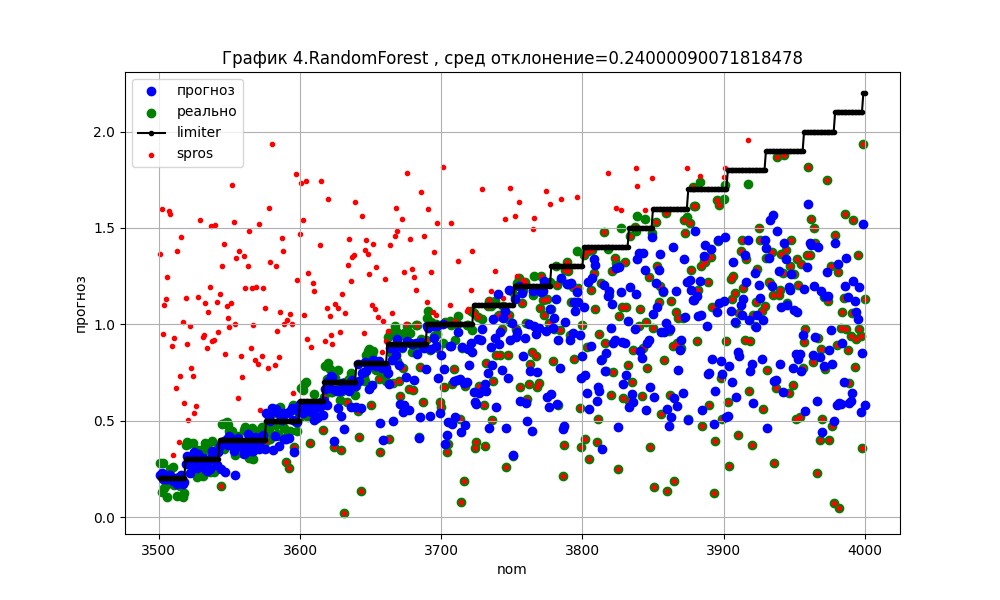


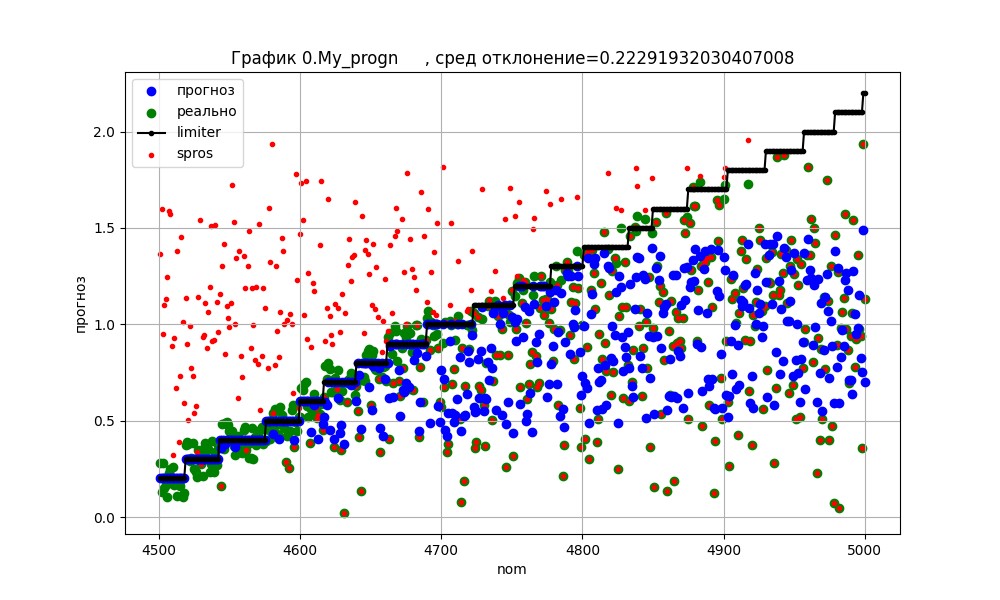
Второй метод – GLМ – не похож ни на что, и даёт очень плохое качество прогнозирования.



Третий метод SIGMOID. Даёт весьма правдоподобный прогноз

Четвёртый метод RandomForest, внешне по графику почти не отличим, но с существенно лучшим качеством прогноза



Ну и последний график по предложенному мною методу – внешне тоже почти не отличается 

СРАВНИТЕЛЬНЫЕ ИТОГИ ПРОГНОЗИРОВАНИЯ

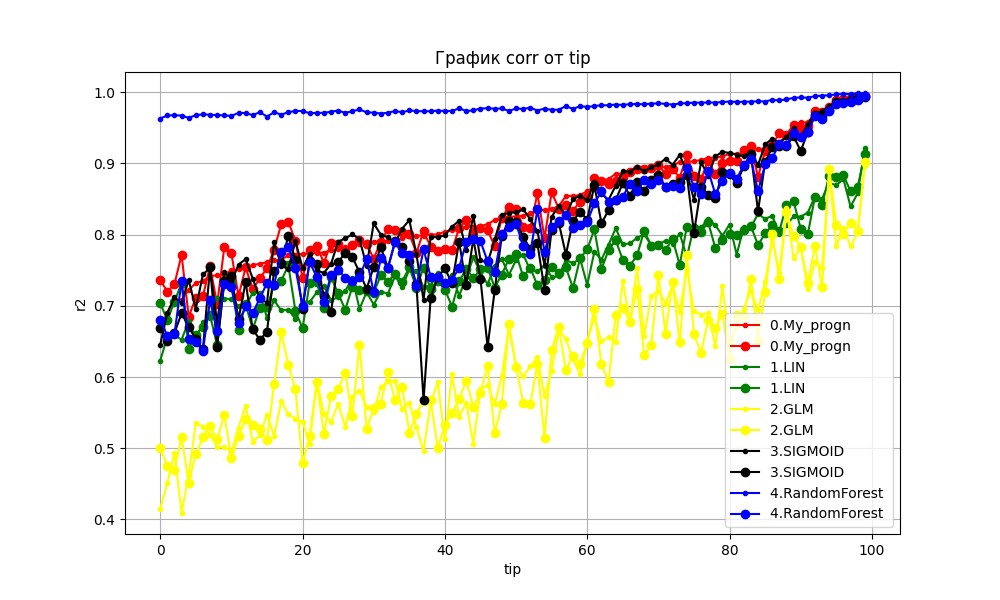
После прочтения результатов прогнозирования они были обработаны следующим образом:

1. Все номера серий были перенумерованы в порядке возрастания значения корреляции прогноза исходной ограниченной случайной величины на настроечном множестве данных моим алгоритмом. По этой причине все графики имеют тенденцию к улучшению качества слева-направо.
2. По каждой серии наблюдений, по каждому методу прогнозирования, по отдельности для настроечного и тестового множеств были посчитаны значения корреляции, среднеквадратичного отклонения, а также показателя r2. При этом, по методу RandomForest и моему алгоритму были посчитаны так же прогнозы на полный спрос при отсутствии ограничителя.
3. Итоги выведены в виде 3 пар графиков, по качеству прогноза самой ограниченной случайной величины, а также по качеству прогноза спроса при отсутствии ограничителя.

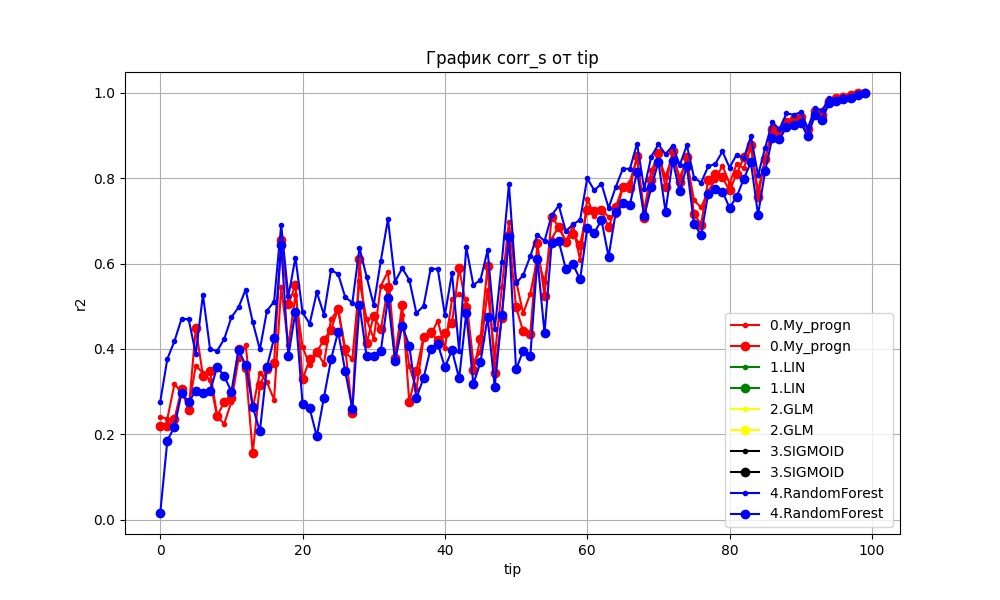
В результате получены следующие графки:

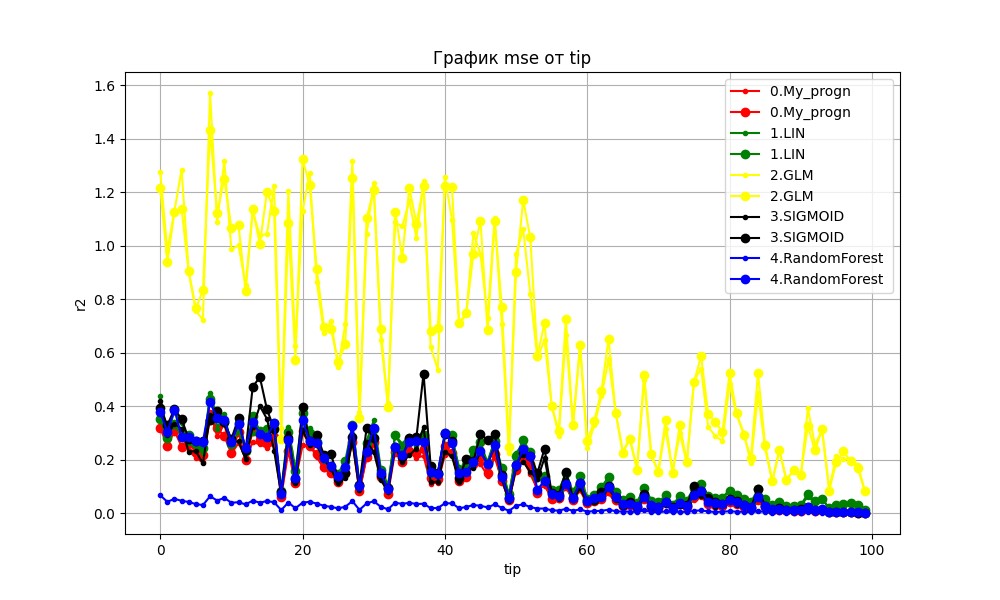
1. Корреляция. Отлично видно, что метод RandomForest (синий график мелкими точками) даёт наивысшую корреляцию, сильно обгоняя все прочие. Однако же, синий график крупными точками (корреляция на тестовом множестве) даёт не просто гораздо более плохой результат, но даже и хуже предложенного мною метода (красный график). Примерно такую же точность, только немного хуже даёт график сигмоиды (чёрный) и ещё хуже даёт линейный прогноз (зелёный). Самый плохой прогноз даёт метод GLM.

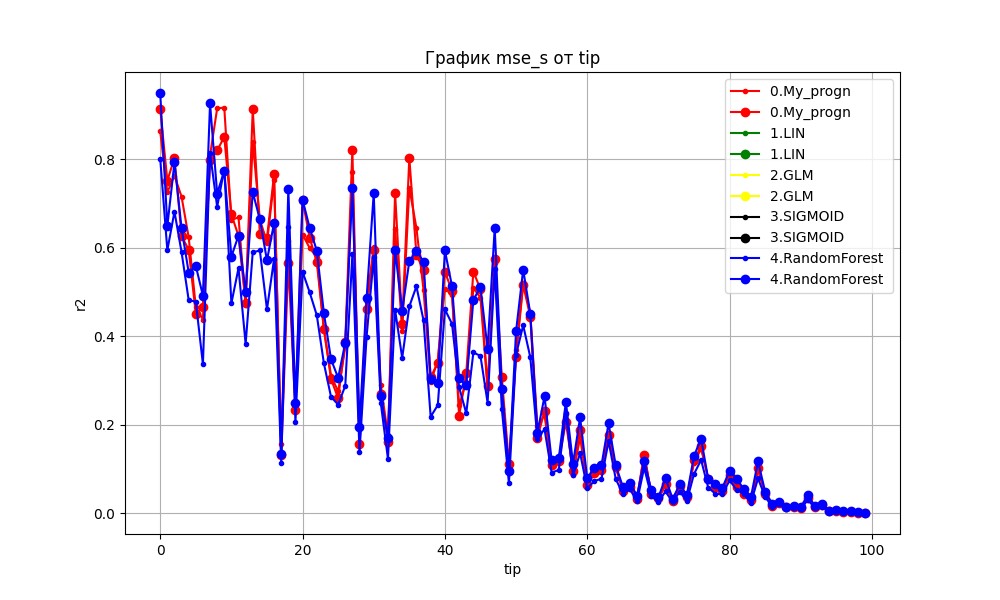
На всех рисунках графики с мелкими точками относятся к итогам по настроечным данным, а с крупными – по тестовым данным.

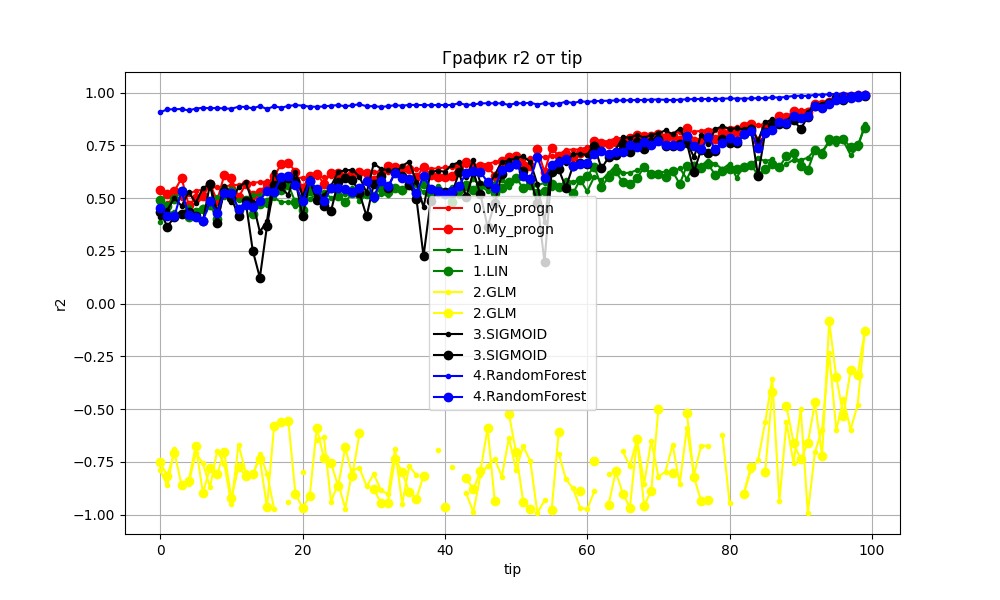
1. 

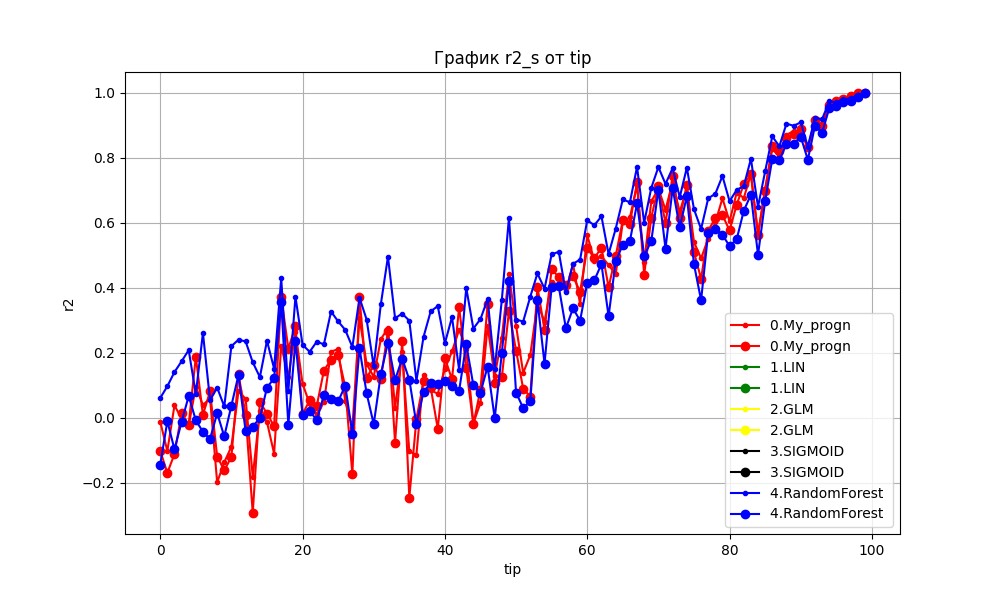
И за ним график корреляции прогноза по спросу



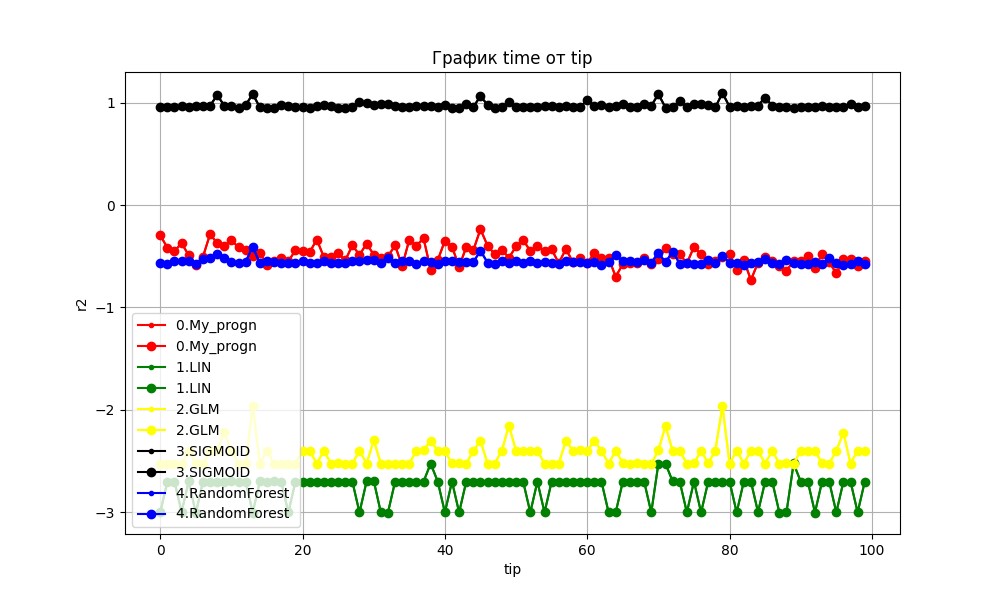
1. Затем аналогичные графики по среднеквадратичному отклонению для прогноза самой ограниченной случайной величины, и для спроса.
2. 

И 

1. И последняя пара графиков по значению R2 – тут на графиках видны разрывы, потому что вычисляемое значение иногда становится отрицательным (вплоть до -100, что теряет всякий смысл), и потому эти значения заменены на None
2. 



И самый последний график – затраченное на вычисления время. По вертикали указан десятичный логарифм затраченного времени.



ВЫВОДЫ

1. Предлагаемый мною алгоритм, будучи написан «на коленке», без всякой оптимизации процесса вычислений, показал скорость настройки на уровне метода RandomForest. Быстрее него оказались только линейные по сути алгоритмы – линейный и GLM. Алгоритм реальной нейросети (SIGMOID) оказался на полтора порядка медленнее.

2. При этом он показал гораздо меньшую переобучаемость, что снижает требования к программистам, которым придётся донастраивать алгоритм для снижения переобучения и увеличения качества прогноза.

3. При этом, предложенный алгоритм предложил безусловно ЛУЧШЕЕ качество настройки в терминах среднеквадратичного отклонения и корреляции. То есть, если по каждой их 100 серии наблюдений на тестовом множестве взять среднеквадратичную ошибку по каждому методу, отсортировать по размеру этой ошибки, выбрать первые 3 места, и посчитать статистику по всем 100 вариантам, то получается такой итог распределения тройки лидеров:

sr=0.My\_progn 3.SIGMOID 4.RandomForest zn==33

sr=0.My\_progn 4.RandomForest 3.SIGMOID zn==32

sr=0.My\_progn 1.LIN 4.RandomForest zn==6

sr=0.My\_progn 4.RandomForest 1.LIN zn==24

sr=0.My\_progn 3.SIGMOID 1.LIN zn==1

sr=3.SIGMOID 0.My\_progn 4.RandomForest zn==1

sr=0.My\_progn 1.LIN 3.SIGMOID zn==3

То есть, результаты переводятся в такую таблицу:

|  |  |  |  |
| --- | --- | --- | --- |
| алгоритм | 1 место | 2 место | 3 место |
| Мой алгоритм | 99 | 1 |  |
| SIGMOID | 1 | 34 | 35 |
| RandomForest |  | 56 | 40 |
| линейный |  | 9 | 25 |

Аналогичное ранжирование по корреляции даёт следующее распределение результатов:

sr=0.My\_progn 3.SIGMOID 4.RandomForest zn==50

sr=0.My\_progn 4.RandomForest 3.SIGMOID zn==28

sr=0.My\_progn 1.LIN 4.RandomForest zn==2

sr=0.My\_progn 4.RandomForest 1.LIN zn==14

sr=0.My\_progn 3.SIGMOID 1.LIN zn==2

sr=0.My\_progn 1.LIN 3.SIGMOID zn==4

Или в виде сводной таблицы получается:

|  |  |  |  |
| --- | --- | --- | --- |
| алгоритм | 1 место | 2 место | 3 место |
| Мой алгоритм | 100 |  |  |
| SIGMOID |  | 52 | 32 |
| RandomForest |  | 42 | 52 |
| линейный |  | 6 | 16 |

ПОСЛЕСЛОВИЕ.

Предлагаемый мною алгоритм рассчитан только для прогнозирования случайной величины с изменяемым во времени ограничителем. В случае разных процентных соотношений превышения спроса над предложением, его превосходство над другими алгоритмами может быть очень разным, от огромного как получилось в этом тестовом примере, вплоть до ничтожного. Мне и самому было бы интересно проверить, насколько сильно он превосходит прочие описанные алгоритмы на реальных данных РЖД, но их у меня просто нет. Если для принятия на работу потребуется проверить качество моего алгоритма на каких-либо реальных данных – готов это сделать.

Все указанные программы доступны для просмотра на моём Гите по адресу

<https://github.com/AlexanderElsufiev/webinar_els/tree/master/my_project>

Единственно, программа настройки нейросети моим алгоритмом там указана с изъятием ключевых строк кода вычисления ошибки прогноза.

Примеры других моих программ, оставшихся с прежней работы, доступны в соседней папке.

<https://github.com/AlexanderElsufiev/webinar_els/tree/master/my_code_in_several_languages>