Санкт-Петербургский государственный университет
Кафедра математической теории игр и
статистических решений

Павлов Леонид Сергеевич

Выпускная квалификационная работа

бакалавра

Предсказание удовлетворенности пользователя при поиске в онлайн-картах

Направление 010400

Прикладная математика, фундаментальная информатика
и основы программирования

Научный руководитель,
кандидат физ.-мат. наук,
старший преподаватель
Грауэр Л. В.

Санкт-Петербург

2016

Содержание

Введение

Миллионы пользователей сети интернет ежедневно взаимодействуют с поисковыми системами. Они подают запросы, следуют по ссылкам со страницы результата поиска, перефразируют и переформулируют свои запросы, а так же выполняют другие различные задания. Эти действия могут служить ценным источником для улучшения поисковых систем.

В настоящее время сервисы онлайн карт становятся популярнее среди пользователей благодаря возможности настроить масштаб и сменить локацию карты при поиске желаемого географического объекта. Таким образом на экран поиска выводится дополнительная информация. Однако, поисковые системы выдают результаты на основе текущего масштаба карты, в то время как желаемый объект может находиться за ее пределами. Даже если масштаб карты идентичен у некоторых пользователей, они могут искать различную информацию. В связи с этим поднимается вопросы удовлетворенности поиском и удобства пользования сервисом.

К сожалению, опыт показывает, что пользователи довольно редко высказывают желание дать свой ответ на эти вопросы. Однако, необходимую информацию можно извлечь из логов пользователя в поисковых системах. Благодаря тому, что ос-

новные поисковые движки обрабатывают миллионы запросов в день, необходимые данные доступны в изобилии.

В условиях того, что подбор необходимой прогнозирующей функции вручную достаточно трудоемок, затратен и непрактичен, исследование в области машинного обучения находит свое применение в этой области задач.

Постановка задачи

В данной работе производится исследование и вычисление прогнозирования удовлетворенности пользователя работой поисковой системы и удобства получения результата. Рассматривается подход к решению задачи с помощью регрессионного анализа путем логистической регрессии на основе использования настоящего поведения пользователей, собранного в результате обычных взаимодействий с поисковой системой в рамках эксперимента.

Для решения представленных выше проблем были достигнуты следующие результаты

1. Реализована программа-парсер, позволяющая вычленить необходимую для исследования и прогнозирования информацию из логов действий пользователей;
2. Сформулированы и построены различные модели для предсказания удовлетворенности пользователя и удобства поиска;
3. Проанализированы результаты, полученные в ходе решения данной задачи;

Обзор литературы

В связи с тем, что данная работа связана с прогнозированием путем использования логов пользователей, были разобраны статьи по данной тематике и изучены методы, представленные в этих трудах. Появление данного этапа работы обусловлено аналогичностью подхода с рассматриваемой нами задачей, с целью адаптирования методов для решения поставленной проблемы.

Одной из похожих задач является прогнозирование смены поисковой системы пользователем.

В [1] показано, что прогнозирование возможно с использованием логов пользователей. Из них были извлечены поисковые сессии. Каждая сессия начиналась с запроса на поисковой движок и содержала страницы результата поиска, посещение главных страниц поисковых систем и страницы соединенные гиперссылками со страницей результата поиска. Сессия считалась оконченной, если пользователь был неактивен более 30 минут.

Так же возможно представление каждой сессии как последовательность символов [1]. Для этой цели был составлен специальный алфавит. Также было извлечено время, которое пользователь провел на странице. Оно было разбито на три

типа в зависимости от продолжительности: 'short', 'medium', 'long'. Положим, что смена поисковой системы есть одно из трех поведений внутри сессии: подача запроса другой поисковой системе, переход к главной странице другого поискового движка, запрос названия другого поисковика.

В этой же статье представлен простой метод классификации. Берется параметр n, обозначающий количество предыдущих символов, которых мы берем в расчет при прогнозировании смены поисковой системы. Строится таблица из трех столбцов, содержащая все виденные ранее последовательности символов длины n, количество раз, когда следующим действием была смена системы (положительный результат) и количество раз, когда следующим действием не была смена поисковой системы (отрицательный результат). В таблице происходит поиск последовательности символов, встреченной в сессии, и вычисляется отношение между положительным и отрицательным результатами для данной последовательности. Если отношение больше, чем известный параметр p, то метод выдает положительный прогноз, иначе – отрицательный. Если данная последовательность символов не была найдена, то она добавляется в таблицу и ей присваивается отрицательный прогноз. Таким образом, таблица обновляется в реальном времени, что позволяет

методу адаптироваться к ранее не известным поведениям.

В статье [2] был представлен новый алфавит. Проводится разделение посещенных страниц на два представления: базовый и продвинутый. В базовым представлении страницы различаются только на те, которые были выданы на странице результатов поиска (Search result page; SERP) или же нет. В продвинутом же представлении используется информация о времени, проведенном на странице.

Так же приведены две величины пользовательских усилий и активности (число запросов и число действий) и две величины, показывающие качество взаимодействия (процент запросов, после которых следовало нажатие на ссылку не связанную с результатом поиска (%NoClicks); а так же количество ссылок, на которых пользователь провел более 30 секунд (SatAction), т.к. они вероятнее будут оценены как успешный результат поиска).

Для предсказания смены поисковой системы была построена модель, использующая логистическую регрессию с некоторыми характеристиками [2]. Характеристики запроса присваиваются последнему запросу в сессии, в которой делается прогноз (среднее количество нажатых ссылок из результата поиска, длина запроса в символах). Они вычисляются из самого

запроса и логов. Характеристики сессии вычисляются исходя из наблюдения действий пользователя до момента совершения прогноза. Они включают в себя информацию о продолжительности сессии (количество поданных запросов или посещенных страниц), успешности поиска (число запросов без нажатий на ссылку со страницы результата) и т.д. Характеристики пользователя вычисляются на всех стадиях сессии на основе текущей истории поиска пользователя, полученной на протяжении некоторого времени. Из каждой истории поиска пользователя были извлечены характеристики запросов, частота поиска, средняя продолжительность сессии (в запросах, времени и ссылок) и доля запросов, поданных предпочитаемой поисковой системе.

В статье [3] рассматривается поиск на онлайн картах. Был разработан метод, позволяющий получить информацию, необходимую пользователю, используя неявную информацию из логов. При работе пользователя с картой поиска, поисковая система может выявлять определенные последовательности действий для определения цели пользователя при поиске и улучшения выдаваемых результатов.

Глава 1. Подготовка данных и выбор модели предсказания удовлетворенности

1.1. Исходные данные

Для данной дипломной работы были предоставлены журналы действий пользователей https://harita.yandex.com.tr в виде файлов представления таблиц баз данных в формате .tsv. В каждой строке этой базы содержится время действие в UNIX-формате, идентификатор пользователя, действие пользователя, а также дополнительные параметры соответствующие определенным действиям. Пользователям, согласившимся на участие в эксперименте, было предложено решить различные задачи, связанные с данным поисковым сервисом. По окончании задания предлагалось заполнить опросник по поводу удобства сервиса и удовлетворенностью результатом участниками.

Участники эксперимента, для выполнения поставленных перед ними задач, пользовались поисково–информационной картографической служба Яндекса.

Карта была поделена на четыре области: область строки запроса, область инструментов поиска, область результатов поиска и область онлайн-карты.

Из предоставленных логов была извлечена информация о

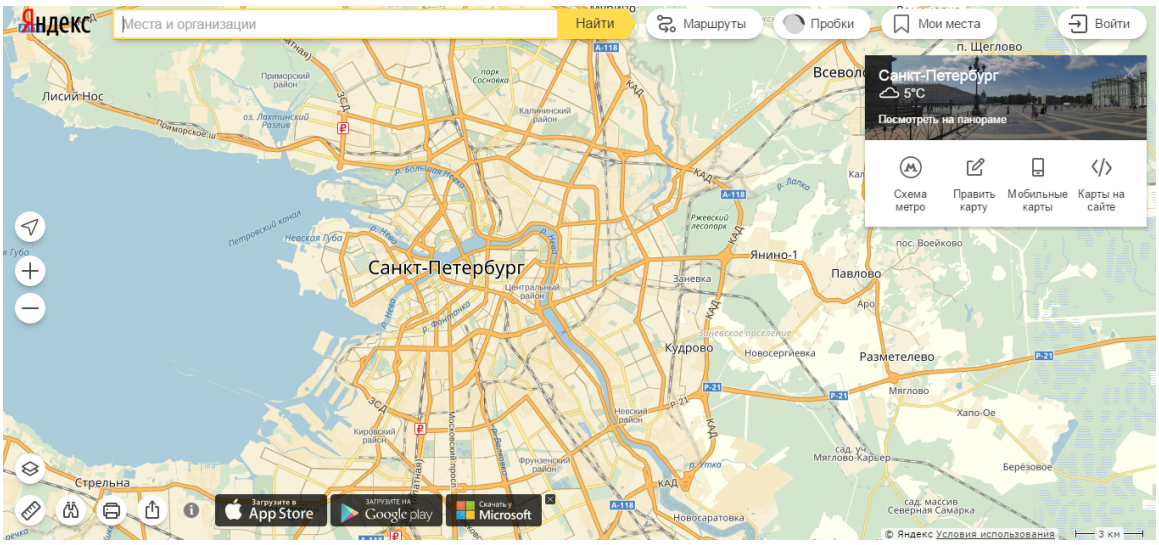

Рис. 1: Главная страница сервиса maps.yandex.ru

координатах нажатий клавиши мыши. Для удобства анализа информации в данной работе был разработан алфавит действий пользователя, где каждому из них ставился в соответствие символ. Были рассмотрены следующие действия

1. task_start (начало выполнения задания, "T");
2. location_change (смена локации, "L");
3. input (ввод текста в поисковую строку, "I");
4. mouse_click (нажатие мыши, "C");
5. mouse_move (движение курсора, "M");
6. task_finish (конец выполнения задания; для положительного результата поиска использовался символ "Y"для отрицательного - "N").

На основе действий были введены признаки (свободные переменные): среднее количество символов в запросе, макси-

мальное количество символов в запросе в одном задании, минимальное количество символов в запросе в одном задании, время между запросами, количество запросов, количество запросов без нажатий мыши, количество запросов с нажатием, общее количество нажатий, количество смен локаций, количество движений мыши, время выполнения задания.

Факторы, связанные с областью нажатия

Помимо данных факторов из логов пользователей, были добавлены новые факторы, связанные с областью нажатия.

При поиске информации на онлайн-карте, пользователь совершает большое число кликов. Нажатия клавиши мыши на одну область карты могут нести в себе совершенно другую информацию в отличие от кликов на другую область. К примеру, нажатие на строку запроса подразумевает, что пользователь начинает или продолжает поисковую сессию, в то время как нажатие на ссылку результата поиска обозначает о нахождении пользователем достаточно удовлетворительном результате.

Рассмотрим интерфейс онлайн-сервиса Яндекс.Карты. В левом верхнем углу находится строка запроса, на правом крае карты формируется выпадающая страница результатов поиска, инструментарий карты расположен в левом нижнем углу

и посередине левого края. Нажатия клавиши мыши по другим областям будем считать кликами по карте.

Логи пользователей содержат координаты кликов, следовательно выполнимо разбиение интерфейса на предложенные выше области и осуществление маркировки каждого нажатия для определенной области.

На основе вышеизложенного на карту были введены дополнительные признаки: количество нажатий на область строки запроса, количество нажатий на область инструментов поиска, количество нажатий на область результатов поиска, количество нажатий на карту. Итого, нами рассмотрено пятнадцать факторов.

В связи с наличием явной обратной связи с участниками эксперимента, посредством проведенного опроса, имеем зависимые переменные: удовлетворенность поиском и удобство пользования сервисом при выполнении задания.

Положим, что все пары (идентификатор задания, идентификатор пользователя) являются наблюдениями. Таких наблюдений в журнале 486. Из них в 437 наблюдениях пользователи были удовлетворены, 37 – частично удовлетворены, 12 – не удовлетворены. Для удобства анализа к не удовлетворенным будем относить и частично удовлетворенных. В 424 наблюдениях

пользователям сервис был удобен, 62 – не удобен.

1.2. Выбор подхода

Одним из распространенных подходов для решения задачи предсказания, является построение регрессионной модели, которые мы рассмотрели в рамках обзора литературы.

Логистическая регрессия

Математической идеей, лежащей в основе логистической регрессии, является логит (logit) – натуральный логарифм вероятностного отношения.

В большинстве случаев логистическая регрессия хорошо подходит для описания и проверки гипотез об отношении между зависимой переменной и одной или более независимых переменных.

Простейшую логистическую модель можно описать следующим образом

$$\mathsf{P}(Y = 1) = \frac{1}{1 + \exp^{-(\alpha+\beta*x)}}$$

где π – вероятность свершения события, β – коэффициент регрессии, α -нулевой коэффициент. [4]

Глава 2. Алгоритм прогнозирования

Для считывания данных была реализована программа парсинга на языке c++.

2.1. Программа для парсинга

На вход подаются три файла, содержащие логи. На выходе программа создает четыре файла: fctr.txt - файл, содержащий матрицу признаков всех наблюдений и вектор зависимых переменных, log.txt содержит последовательность действий пользователя, представленную в виде алфавита, а так же время запроса, непосредственно сам запрос и время завершения задания, answr.txt содержит текст задания предлагаемого на выполнение участнику эксперимента, plot.txt содержит координаты всех нажатий клавиши мыши на карту и номер соответствующего им задания.

Парсинг реализован путем построчного считывания. Первые три столбца в таблице для всех одинаковы: время в UNIX-формате, идентификатор пользователя и действие пользователя. Эти три атрибута заносятся в массив tag. В связи с тем, что набор всех действий в логах известен, проводится проверка на каждое действие, в зависимости от него в log.txt записыва-

ется значение, соответствующее считанному действию в алфавите. Для действий location_change, mouse_click, mouse_move, input увеличивается счетчик. Из действия task_start извлекается идентификатор задания и его содержание. Из действия mouse_click считываются координаты нажатия на карту. Действие input в журнале чувствительно к изменению одного символа в поисковой строке, оно не всегда соответствует каждому запросу, посланного пользователем поисковой системе. В программе парсинга был успешно реализован обход этой проблемы через действия input и mouse_move.

Таким образом происходит извлечение заданных нами необходимых факторов для построения логистической регрессии. Полученные данные записываются в файлы выхода.

2.2. Построение модели регрессии

Для построения модели логистической регрессии используется программа, разработанная на языке R. На вход подается матрица факторов размерностью 486x11 и вектор ответов 486x2, содержащихся в файле fctr.txt и координаты нажатий на карту из файла plot.txt. Производится их считывание и инициализация переменных.

На основе информации из файла plot.txt выясняются раз-

меры карты, выделяются определенные выше области карты и на них наносятся точки нажатий.

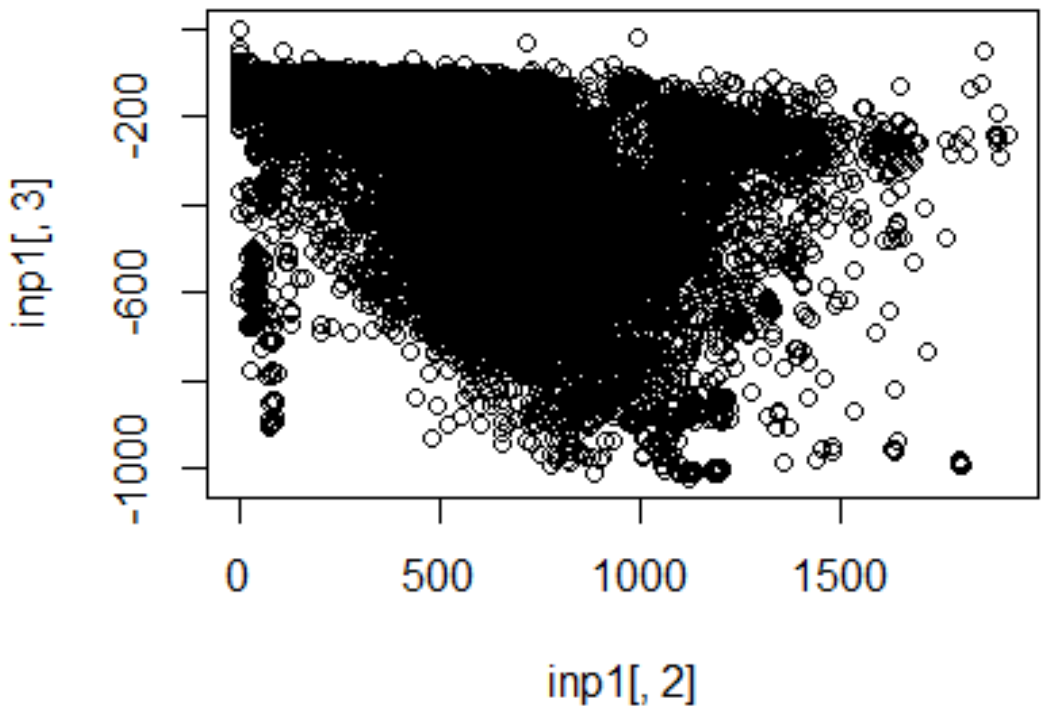

Рис. 2: Карта нажатий пользователей

Структура матрицы факторов представляет собой набор из 486 наблюдений по строкам и 15 признаков по столбцам в данном порядке: среднее количество символов в запросе (avg_smbls), максимальное количество символов в запросе в одном задании (max_smbls), минимальное количество символов в запросе в одном задании (min_smbls), время между запросами (btwn_tme), количество запросов (q_cnt), количество запросов

без нажатий мыши (mc0_cnt), количество запросов с нажатием (mc1_cnt), общее количество нажатий (mc_cnt), количество смен локаций (lc_cnt), количество движений мыши (mm_cnt), время выполнения задания(tsk_tme), количество нажатий на область строки запроса (query_bar), количество нажатий на область инструментов поиска (tools_bar), количество нажатий на область результатов поиска (result_bar), количество нажатий на карту (map_click) . Вектор ответов собственно содержит маркеры для данных 486 наблюдений.

Производится разбиение общей выборки на обучающее и тестовое подмножества с процентным соотношением 75/25 успешных наблюдений к неуспешным в каждом из подмножеств и с соотношением 75/25 количества наблюдений обучающей выборки к количеству наблюдений тестовой выборки. В результате в обучающем подмножестве имеется 364 наблюдения, в тестовом – 122 наблюдения.

Далее происходит построение нескольких моделей предсказывания. Первая модель использует все 15 считываемых факторов, вторая модель использует четыре фактора, связанных с областью нажатия, и всевозможные их произведения. Для построения следующих трех моделей используется stepwise регрессия на 15 факторах, которая выбирает факторы с помо-

щью информационного критерия Акаике, с прямым, обратным и двухсторонним подходами соответственно. Шестая и седьмая модель так же построены с помощью прямого и двухстороннего подходов stepwise регрессии на четырех факторах, связанных с областью нажатия.

Так же был проведен тест Вальда для оценки значимости всех коэффициентов модели предсказания. Вычисленное значение статистики хи–квадрат $X^2 = 24.4$, с тринадцатью степенями свободы и $P - value = 0.028$ указывает, что коэффициенты являются значимыми.

2.3. Предсказание удовлетворенности

Таблица 1: Таблица весов факторов для различных моделей алгоритмов предсказания удовлетворенности при поиске

	1	2	3	4	5	6	7
(Intrcpt)	−1.8559	−3.121	−3.5421	1.5962	−3.5421	−2.7278	−2.7278
avg_s	0.0445	0	0.0481	0.0462	0.0481	0	0
max_s	0.0045	0	0	0	0	0	0
min_s	0.1642	0	0	0	0	0	0
btn_t	0.0890	0	0	0	0	0	0
q_c	0.1600	0	0	0.1370	0	0	0
mc0_c	−0.1446	0	0	−0.1272	0	0	0
mc1_c	0	0	0.1466	0	0.1466	0	0
mc_c	−0.0068	0	0	0	0	0	0
lc_c	−0.0077	0	0	0	0	0	0
mm_c	1.4240	0	0	0.9547	0	0	0
tsk_t	−0.9631	0	0	−0.8959	0	0	0
q_bar	0.0041	0.0671	0	0	0	0.0512	0.0512
t_bar	0.0866	−1.540	0	0	0	0	0
r_bar	0.0094	0.0221	0	0	0	0	0
map_c	0	0.0103	0	0	0	0	0
q_bar : t_bar	0	0.1505	0	0	0	0	0
q_bar : r_bar	0	0.0021	0	0	0	0	0
t_bar : r_bar	0	−3.395	0	0	0	0	0
q_bar : map_c	0	0.0002	0	0	0	0	0

Таблица 2: Продолжение таблицы 1

	1	2	3	4	5	6	7
t_bar : map_c	0	0.0737	0	0	0	0	0
r_bar : map_c	0	−0.0003	0	0	0	0	0
q_bar : t_bar : r_bar	0	0.0828	0	0	0	0	0
q_bar : t_bar : map_c	0	−0.0062	0	0	0	0	0
q_bar : r_bar : map_c	0	0	0	0	0	0	0
t_bar : r_bar : map_c	0	0.0414	0	0	0	0	0
q_bar : t_bar : r_bar : map_c	0	−0.0011	0	0	0	0	0

С помощью прямого подхода stepwise регрессии (модель 3) было выбрано два фактора из пятнадцати: количество запросов с хотя бы одним нажатием на карту и среднее количество символов в запросе; используя обратный подход (модель 4), было выбрано пять факторов: среднее количество символов в запросе, количество запросов, количество запросов без нажатий на карту, количество движений мыши и время выполнения задания. Для модели 5 был выбран двухсторонний подход и, по результатам его работы были выбраны те же факторы, что и в модели 3. Модель 6 и модель 7, модель 3 и модель 5 получились идентичными.

Для первой модели значимыми факторами являются среднее количество символов в запросе, количество запросов и количество движений мыши, для второй, шестой и седьмой – нажатие в область строки запроса, для третьей – количество запросов с хотя бы одним нажатием на карту, для четвертой – количество запросов, для пятой – количество запросов с хотя бы одним нажатием на карту.

Далее происходит построение таблиц сопряженности для каждой модели. Таким образом определим точность, чувствительность и специфичность каждого алгоритма.

Чтобы оценить качество классификации, производится

ROC-анализ на тестовой выборке. Строится ROC–кривая и вычисляется показатель AUC (площадь под ROC–кривой). На рис.3 и рис 4. представлены их сравнительные графики. Синим цветом выделена ROC–кривая первой модели, ROC–кривые для третьей и пятой моделей совпадают, они выделены на графике черным цветом; зеленая кривая представляет четвертую модель. Коричневым цветом выделена кривая 6 и 7 моделей, фиолетовым – для 2 модели.

Все модели, исключая модель 2, имеют высокие значения точности и чувствительности, а значит часто выдают истинный результат при положительном исходе. Низкая специфичность всех моделей обусловлена малым количеством наблюдений с отрицательным исходом. Выбираем наилучшую модель, исходя из наибольшего значения AUC. Из табл.3 видно, что

Таблица 3: Оценки качества моделей предсказания удовлетворенности

	1	2	3	4	5	6	7
Точность	0.9091	0.7521	0.9091	0.9091	0.9091	0.9008	0.9008
Чувствительность	0.9908	0.82569	0.9908	0.9908	0.9908	0.9908	0.9908
Специфичность	0.1667	0.08333	0.1667	0.1667	0.1667	0.0833	0.0833
AUC	0.6964	0.4545	0.6693	0.6598	0.6693	0.792	0.792

наилучший результат продемонстрировали модели 6 и 7, использующие stepwise регрессию только для факторов, связанных с областью нажатия, включив в алгоритм только нажатия на область запроса. Так же неплохой результат показала модель 1 со значением $AUC = 0.6964$, что является приемлемым результатом. Остальные модели показали небольшое значение AUC, что значит о неудовлетворительном качестве алгоритма. Наихудший результат показала модель 2, $AUC = 0.4545$, хуже, чем алгоритм случайного присваивания маркировок, имеющий $AUC = 0.5$. Данный вывод не позволяет применять эту модель для решения задач на практике.

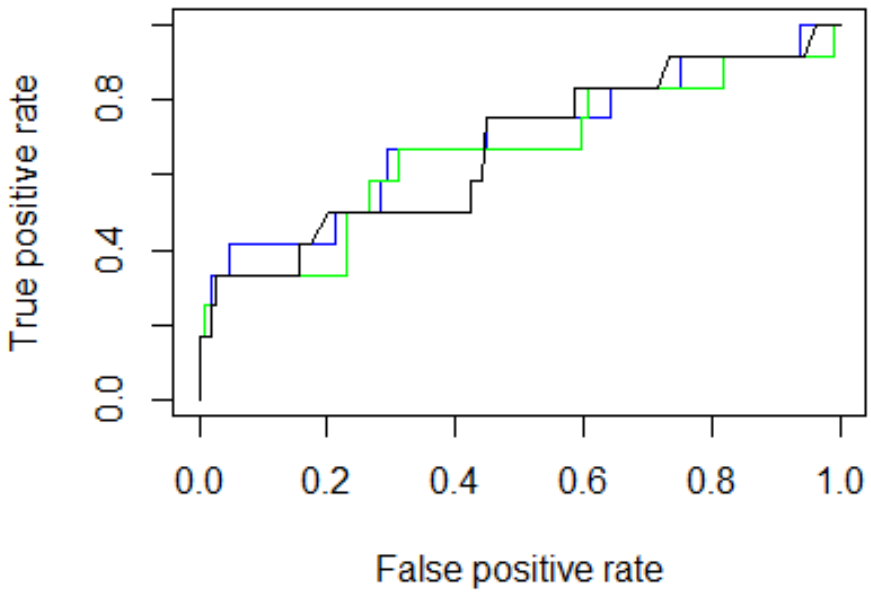

Рис. 3: ROC-кривые

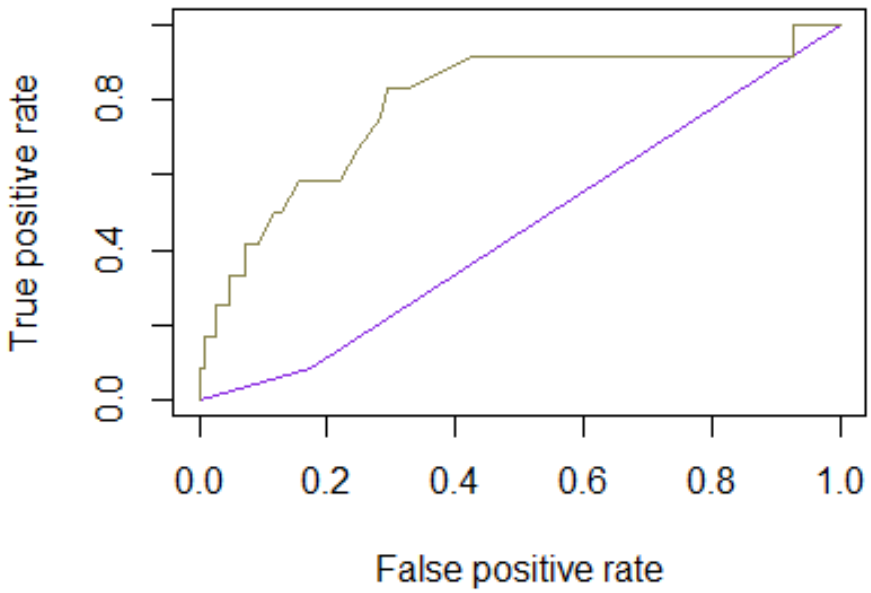

Рис. 4: ROC-кривые

2.4. Предсказание удобства поискового сервиса

Таблица 4: Таблица весов факторов для различных моделей алгоритмов предсказания удобства поискового сервиса

	1	2	3	4	5	6	7
(Intrcpt)	−3.6166	−2.330	−3.1727	−2.6338	−3.1727	−2.6338	−2.6338
avg_s	0.0284	0	0.0333	0.0462	0.0333	0	0
max_s	0.0023	0	0	0	0	0	0
min_s	0.1322	0	0	0	0	0	0
btn_t	0.3109	0	0	0	0	0	0
q_c	0.1041	0	0	0.14594	0	0	0
mc0_c	−0.1592	0	0	−0.1950	0	0	0
mc1_c	0	0	0.1353	0.1354	0.1353	0	0
mc_c	−0.0243	0	0	−0.0151	0	0	0
lc_c	0.0017	0	0	0	0	0	0
mm_c	1.3230	0	0	0.7289	0	0	0
tsk_t	−0.9281	0	0	−0.8959	0	0	0
q_bar	0.0351	0.0394	0	0	0.0476	0.0476	0.0476
t_bar	0.1038	−1.564	0	0.10241	0	0	0
r_bar	0.0012	−0.0108	0	0	0	0	0
map_c	0	−0.036	0	0	0	0	0
q_bar : t_bar	0	0.294	0	0	0	0	0
q_bar : r_bar	0	−0.0004	0	0	0	0	0
t_bar : r_bar	0	−1.305	0	0	0	0	0
q_bar : map_c	0	0.01569	0	0	0	0	0

Таблица 5: Продолжение таблицы 4

	1	2	3	4	5	6	7
t_bar : map_c	0	0.1316	0	0	0	0	0
r_bar : map_c	0	0.0001	0	0	0	0	0
q_bar : t_bar : r_bar	0	−0.0567	0	0	0	0	0
q_bar : t_bar : map_c	0	−0.0120	0	0	0	0	0
q_bar : r_bar : map_c	0	−0.0006	0	0	0	0	0
t_bar : r_bar : map_c	0	0.0105	0	0	0	0	0
q_bar : t_bar : r_bar : map_c	0	0.0051	0	0	0	0	0

Как и для предыдущей задачи, были построены те же самые модели, после выполнения прямого, обратного и двустороннего stepwise подходов выбрались аналогичные факторы. Исключение составляет лишь предсказываемая переменная, где в ее качестве был выбран фактор успеха поиска. Идентичными оказались модели 6 и 7, а так же модели 3 и 5.

Цвета графиков ROC–кривых оставили неизменными. Синим цветом выделена ROC–кривая первой модели, ROC–кривые для третьей и пятой моделей совпадают, они выделены на графике черным цветом; зеленая кривая представляет четвертую модель. Коричневым цветом выделена кривая 6 и 7 моделей, фиолетовым – для 2 модели. Их сравнение представлено на рис. 5 и рис.6

Все модели снова показали высокую точность и чувствительность, при этом модели 6 и 7 показали значение чувствительности максимально приближенное к единице, что говорит о возможности использования этих моделей, при требовании прогнозировать только успешные наблюдения. Однако значение специфичности у данных алгоритмов ниже, чем у остальных – 0.0625. Следовательно, для выявления неудачных результатов ээта модель не пригодна. Наилучшим решением по параметру AUC оказались модели 3 и 5 $AUC = 0.8107$, что является

Таблица 6: Оценки качества моделей предсказания удобства поискового сервиса

	1	2	3	4	5	6	7
Точность	0.8689	0.8443	0.8852	0.877	0.8852	0.877	0.877
Чувствительность	0.9811	0.9906	0.9811	0.9906	0.9528	0.995	0.995
Специфичность	0.1250	0.1875	0.1875	0.1875	0.1875	0.0625	0.0625
AUC	0.7547	0.63207	0.8107	0.7824	0.8107	0.6877	0.6877

отличным результатом для построенного алгоритма прогнозирования. Эти модели, stepwise регрессии с прямым и двухсторонним подходом, показали лучшие результаты по сравнению с другими алгоритмами.

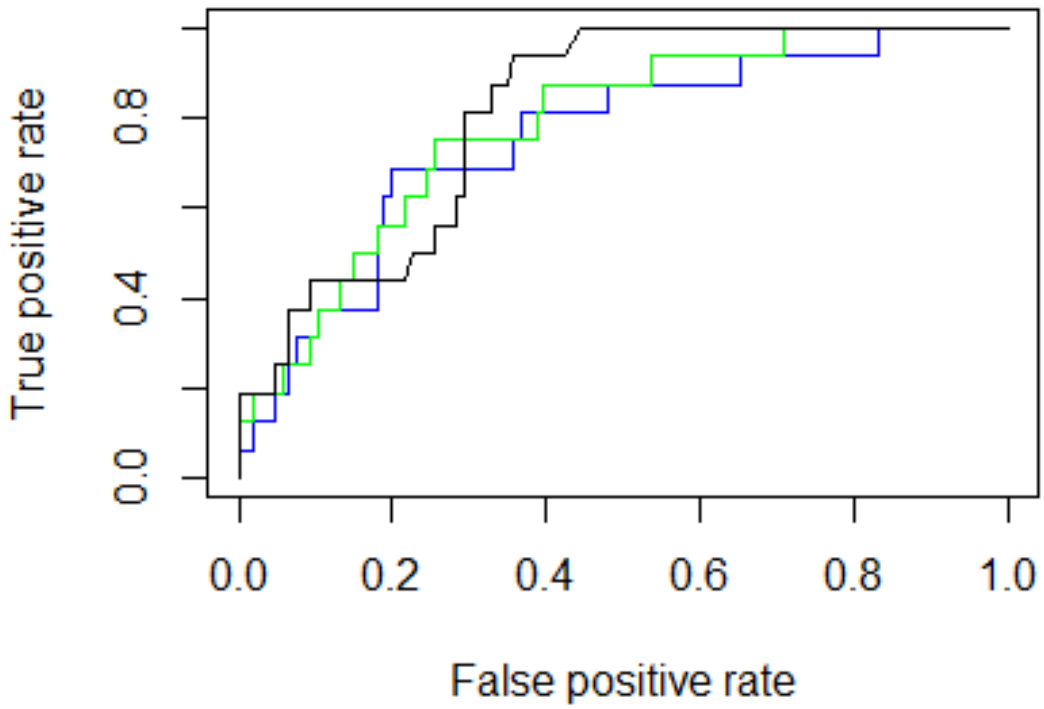

Рис. 5: ROC-кривые

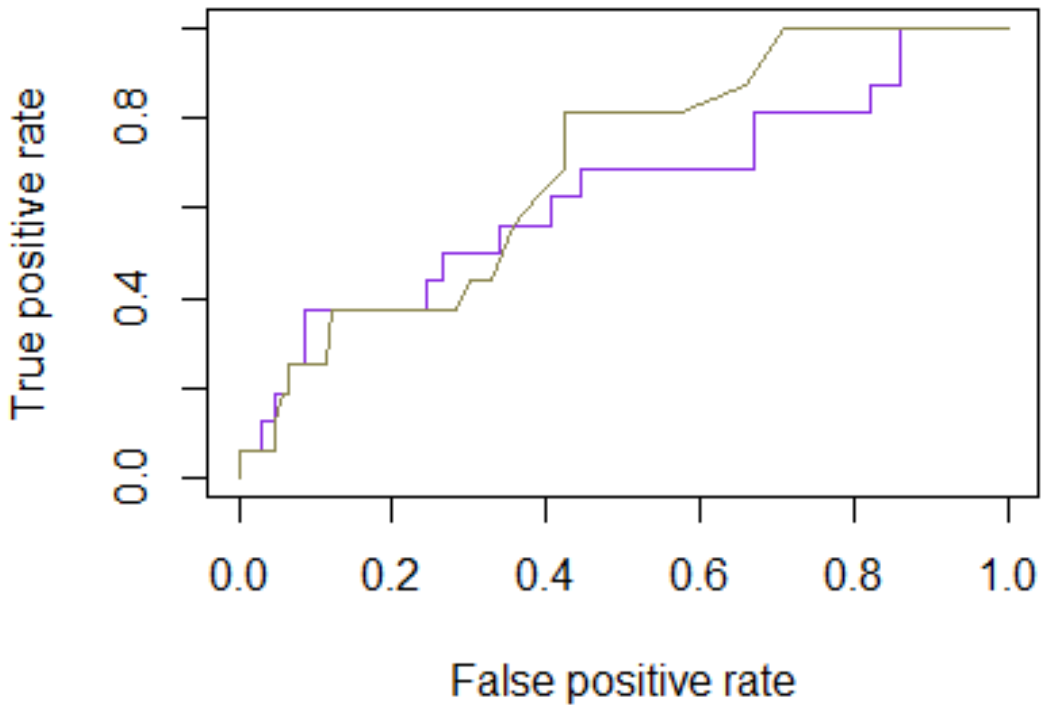

Рис. 6: ROC-кривые

Заключение

В данной работе рассмотрены модели предсказания удовлетворенности пользователя и успешности при поиске в онлайн-картах при помощи логистической регрессии.Разработана программа парсинга логов пользователей на языке c++. Были предложены различные факторы на основе логов пользователей и интерфейса карты, был произведен выбор значимых факторов. Произведена оценка качества построенных моделей. Для этих целей создана программа на языке R. В рамках предоставленных данных модель с учетом фактора нажатия в область строки запроса показала наилучший результат для предсказания удовлетворенности пользователя при поиске. Для задачи предсказания удобства пользования сервисом поиска наилучшей стала модель с учетом двух факторов: среднего количества символов в запросе и количества запросов с последующим хотя бы одним нажатием клавиши мыши.

Список литературы

1. Allison P. Heath, Ryen W. White. Defection detection: predicting search engine switching // In Proceedings of the 17th international conference on World Wide Web (WWW '08). 2008.

2. Ryen W. White, Susan T. Dumais. Characterizing and predicting search engine switching behavior // In Proceedings of the 18th ACM conference on Information and knowledge management (CIKM '09). 2009.

3. Hiramoto R., Sumiya K. Web information retrieval based on user operation on digital maps // GIS '06 Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems, 2006 P.99-106

4. Chao-Ying J. Peng, K. L. Lee, Gary M. Ingersoll. An introduction to logistic regression analysis and reporting // EBSCO Publishing. 2002.

Приложение

Приложение 1. Реализация парсинга на языке C++

```
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>
#include <stdio.h>
#include <algorithm>
#include <vector>

std::ofstream log_file("log.txt", std::ios::app);      // файл логов
std::ofstream fctr_file("fctr.txt", std::ios::app);    // файл факторов
std::ofstream answr_file("answr.txt", std::ios::app);  // файл ответов
std::ofstream vect_plot("plot.txt", std::ios::app);    // файл точек нажатий на карту

std::ifstream f("C:\\Users\\Leonid\\Downloads\\happiness logs\\happiness logs\\168_filtered.tsv");
std::ifstream f("C:\\Users\\Leonid\\Downloads\\happiness logs\\happiness logs\\169_filtered.tsv");
std::ifstream f("C:\\Users\\Leonid\\Downloads\\happiness logs\\happiness logs\\170_filtered.tsv");


int main(){

int num_task_1 = 1;         // номер задания (для структуры координат кликов)
int click_count = 0;
int mc = 0;                 // количество mouse click в данном запросе
int mc_count = 0;           // общее количество кликов
int mc0_count = 0;          // количество запросов без нажатия на результат
int mc1_count = 0;          // количество запросов с хотя бы одним нажатием на результат
int mm_count = 0;           // общее количество движений мыши
int lc_count = 0;           // общее количество смен локаций
int sugg_count = 0;         // общее количество смен локаций
int query_count = 0;        // количество запросов
int query_len = 0;          // длина запроса
int max_query_len = 0;      // максимальная длина запроса
int min_query_len = 1000;   // минимальная длина запроса
int query_len_avg = 0;      // средняя длина запроса
int flag;                   // значение наблюдаемого показателя (успех или неудача при поиске информации)
int flag_conv;              // значение наблюдаемого показателя (удовлетворенность пользователя)

unsigned long long tag_time;                 // время подачи команды
unsigned long long task_time_start;          // время начала сессии
unsigned long long task_time = 0;            // продолжительность сессии
unsigned long long query_time = 0;           // время подачи команды input
unsigned long long tru_query_btwn_time = 0;  // время между запросами
unsigned long long query_btwn_time = 0;      // время, затраченное на написание запроса
unsigned long long query_pend_time = 0;      //время подачи запроса
unsigned long long query_start_time = 0;     // время начала написания запроса
unsigned long long click_X = 0;
unsigned long long click_Y = 0;
unsigned long long max_click_X = 0;
unsigned long long max_click_Y = 0;

unsigned long long atoull(std::string);      // функция преобразования переменной типа string в тип unsigned long long

bool start_bool = true;    // флаг начала сессии

std::string tag[3];        // массив первых трех аттрибутов: время, id пользователя, действие
std::string s;             // считываемая строка
std::string user_id;
std::string task_id;
std::string tag_query;     // слово, заданное пользователем в поисковую систему
```

```
std::string tag_pend = ""; // слово, заданное пользователем в поисковую систему (проверка)
std::string tag_answ;
std::string str_click_X;
std::string str_click_Y;

std::string quest_0 =  "Did you manage to complete the task?";
std::string quest_1 =  "Was it convenient to use this service to plan the route?";
std::string quest_2 =  "Was it convenient to use this service for performing the task?";
std::string hit =      "{\"questionId\":1,\"answer\"";
std::string hit_yes =  "{\"questionId\":1,\"answer\":{\"text\":\"Yes\",\"badge\":\"positive\"}";
std::string hit_part = "{\"questionId\":1,\"answer\":{\"text\":\"Partly\",\"badge\":\"neutral\"}";
std::string hit_no =   "{\"questionId\":1,\"answer\":{\"text\":\"No\",\"badge\":\"negative\"}";

std::string hit_yes_conv =      "{\"questionId\":4,\"answer\":{\"text\":\"Very convenient\",\"badge\":\"none\"}";
std::string hit_part_yes_conv = "\"questionId\":4,\"answer\":{\"text\":\"Somewhat convenient\",\"badge\":\"none\"}";
std::string hit_part_no_conv =  "\"questionId\":4,\"answer\":{\"text\":\"Somewhat inconvenient\",\"badge\":\"none\"}";
std::string hit_no_conv =       "{\"questionId\":4,\"answer\":{\"text\":\"Very inconvenient\",\"badge\":\"negative\"}";

    // координаты клика
struct loc_mc{
int num_task; // номер задания
unsigned long long struct_click_X; // координата X
unsigned long long struct_click_Y; // координата Y
};

std::vector <loc_mc> v;
loc_mc click;

while(!f.eof()){

std::getline(f, s);     // построчное считывание
for(int i=0; i<3; ++i){ // заполняем массив аттрибутов

size_t n = s.find('\t');
tag[i] = s.substr(0, n);
s = s.substr(n+1, s.length()-1);

}
tag_time = atoull(tag[0].c_str()); //переводим время действия в unsigned long long
user_id = tag[1];
if(tag[2] == "task_start"){

start_bool = true;
log_file << "T";
task_time_start = tag_time;
query_time = task_time_start;
size_t nn = s.find("\"description\":\"");
s = s.substr(nn+15, s.length()-15);
size_t nn1 = s.find("\"");
tag_answ = s.substr(0, nn1);
answr_file << tag_answ << '\n';
size_t n1 = s.find("task_id=");  // номер сессии
task_id = s.substr(n1, s.length()-1);

}
if(tag[2] == "location_change"){

log_file << "L";
lc_count++;

}
if(tag[2] == "mouse_move"){

if((tag_query != tag_pend) && (tag_pend != "")){

tag_query = tag_pend;
query_len = strlen(tag_query.c_str());  // длина запроса
if (query_len < min_query_len) min_query_len = query_len;
if (query_len > max_query_len) max_query_len = query_len;
```

```
query_len_avg = query_len_avg + query_len;
query_pend_time = query_time;
query_start_time = 0;
log_file << '\n' << tag_query << '\t' << query_pend_time << '\n';
query_count++;
if (mc == 0) mc0_count++;
else mc1_count++;
mc = 0;

}
log_file << "M";
mm_count++;

}
if(tag[2] == "mouse_click"){


v.push_back(click);                         // инициализируем клик

// считываем координату X
size_t n_mcX = s.find("screenX=");
s = s.substr(n_mcX + 8, s.length() - 8);
size_t n_mcX_end = s.find('\t');
str_click_X = s.substr(0, n_mcX_end);
click_X = atoull(str_click_X.c_str());     // перевод из string в unsigned long long

// считываем координату Y
size_t n_mcY = s.find("screenY=");
s = s.substr(n_mcY + 8, s.length() - 8);
size_t n_mcY_end = s.find('\t');
str_click_Y = s.substr(0, n_mcY_end);
click_Y = atoull(str_click_Y.c_str());

// убираем выбросы
if ((click_X > 10000) || (click_Y > 10000)) {
click_X = 0;
click_Y = 0;
}
v.at(click_count).struct_click_X = click_X;
v.at(click_count).struct_click_Y = click_Y;
v.at(click_count).num_task = num_task_1;
mc++;
mc_count++;
click_count++;
log_file << "C";

}
if(tag[2] == "input"){

log_file << "I";
size_t n1 = s.find('\t');   // считываем, что в аттрибуте query=
s = s.substr(n1+1, s.length()-1);
n1 = s.find('=');
size_t n2 = s.find('\t');
tag_pend = s.substr(n1+1, n2-n1-1);
query_time = tag_time;
if (query_start_time == 0){ // время начала отправки запроса

query_start_time = tag_time;
if(start_bool == true)
start_bool = false;
else{
query_btwn_time = query_btwn_time + query_start_time - query_pend_time;
}


}

}
```

```
if(tag[2] == "task_finish") {

task_time = tag_time - task_time_start;
tru_query_btwn_time = task_time - query_btwn_time;
log_file << "F" << '\n' << task_time << '\t' << user_id << '\t' << task_id << '\n' ;

// вывод факторов
if (query_count == 0) fctr_file << 0 << '\t' << 0 << '\t' << 0 << '\t' << task_time << '\t';
else
fctr_file << query_len_avg/query_count << '\t' << max_query_len << '\t' << min_query_len << '\t' << tru_query_btwn_time << '\t';
fctr_file  << query_count << '\t' << mc0_count << '\t' << mc1_count << '\t' << mc_count << '\t'
   << lc_count << '\t'  << mm_count << '\t' << task_time << '\t';
num_task_1++;
query_pend_time = 0;
query_btwn_time = 0;
query_count = 0;
query_len_avg = 0;
tag_query = "";
tag_pend = "";
sugg_count = 0;
mc0_count = 0;
mc1_count = 0;
mc_count = 0;
mc = 0;
lc_count = 0;
mm_count = 0;
// маркировка успеха поиска (заменены 0 и 1 между собой)
if(s.find(hit_yes) != std::string::npos) {

flag = 0;

}
else{

flag = 1;

}

// маркировка удовлетворенности пользователя (заменены 0 и 1 между собой)
if((s.find(hit_yes_conv) != std::string::npos) || (s.find(hit_part_yes_conv) != std::string::npos)) flag_conv = 0;
else flag_conv = 1;
fctr_file << flag << '\t' << flag_conv << '\n';

}

}

// вывод карты
for (std::vector<loc_mc>::iterator it=v.begin();it!=v.end();it++){
vect_plot << it -> num_task << '\t' << it-> struct_click_X << '\t' << it-> struct_click_Y << '\n';
}
std::system("pause");
return 0;

}

// перевод из string в unsigned long long
unsigned long long atoull(std::string str){
unsigned long long result = 0;
for (int i = 0; i < str.length(); i++ ) {

result = (result*10) + (str[i] - '0');

}
return result;

}
```

Приложение 2. Реализация моделей логистической регрессии для прогнозирования на языке R

```
library(caret)
  library(ROCR)
  library(pROC)
  library(reshape)
  library(plyr)
  library(nortest)
  library(ggplot2)
  library(caTools)
  library(e1071)
  library(aod)

  # инициализация файла с координатами кликов
  inp1 <- read.table("plot.txt")
  inp1[5913:11787,1] <- inp1 [5913:11787,1] + 137
  inp1[11788:20234,1] <- inp1 [11788:20234,1] + 302

  # максимальные и минимальные значения координат на карте
  max_click_X <- max(inp1[,2])
  max_click_Y <- max(inp1[,3])
  min_click_X <- min(inp1[,2])
  min_click_Y <- min(inp1[,3])

  # разбиение карты на области

  # границы области строки запроса
  query_bar_Y <- -210
  query_bar_X <- 750

  # границы области инструментов карты
  tools_up_Y <- -500
  tools_down_Y <- -700
  tools_X <- 100

  # границы области результатов поиска
  result_click_X <- 1000
  result_click_Y <- 700

  # коррекция координат
  inp1[,3] <- -inp1[,3]
  plot(inp1[,2],inp1[,3]) # вывод графика нажатий на карту

  # инициализация факторов
  inp <- read.table("fctr.txt")
  inp <- rename(inp,
                c(V1 = 'avg_smbls',      # Среднее кол-во символов в запросе
                  V2 = 'max_smbls',      # Макс кол-во символов
                  V3 = 'min_smbls',      # Мин кол-во символов
                  V4 = 'btwn_tme',       # Время между запросами
                  V5 = 'q_cnt',          # Кол-во запросов
                  V6 = 'mc0_cnt',        # Кол-во без нажатий
                  V7 = 'mc1_cnt',        # Кол-во с нажатием
                  V8 = 'mc_cnt',         # Кол-во нажатий
                  V9 = 'lc_cnt',         # кол-во смен локаций
                  V10 = 'mm_cnt',        # кол-во движений мыши
                  V11 = 'tsk_tme',       # время выполнения задания
                  V12 = 'labels_suc',    # маркировка успеха
                  V13 = 'labels_cvn')    # Маркировка удовлетворенности
                )

  # логарифмирование временных факторов и кол-ва движений мыши
  inp[,4] = log(inp[,4])
```

```
inp[,10] = log(inp[,10])
inp[,11] = log(inp[,11])

query_bar <- 0    # кол-во нажатий в область строки запроса
tools_bar <- 0    # кол-во нажатий в область инструментов
result_bar <- 0   # кол-во нажатий в область результатов поиска
map_click <- 0    # кол-во нажатий в область карты


# инициализцаия матрицы с нажатиями по карте
point_click <- matrix(data = NA, nrow = 486, ncol = 4, byrow = T)
for(i in 1:486){
  for(j in 1:20234){
    if (inp1[j,1] == i) {
      if ((inp1[j,2] <= query_bar_X) & (inp1[j,3] >= query_bar_Y)) query_bar <- query_bar + 1
      else
        if ((inp1[j,2] <= tools_X) & (inp1[j,3] >= tools_down_Y) & (inp1[j,3] <= tools_up_Y)) tools_bar <- tools_bar + 1
        else
          if ((inp1[j,2] >= result_click_X) & (inp1[j,3] <= result_click_Y)) result_bar <- result_bar + 1
          else
            map_click <- map_click + 1
    }
  }
  point_click[i,1] <- query_bar
  point_click[i,2] <- tools_bar
  point_click[i,3] <- result_bar
  point_click[i,4] <- map_click
  query_bar <- 0
  tools_bar <- 0
  result_bar <- 0
  map_click <- 0
}

plot(inp1[,2],inp1[,3])
#View(point_click)

####
# для успешности посика

# построение окончательной таблицы факторов
fctr <- transform(inp[,1:11],point_click)
fctr$query_bar <- point_click[,1]
fctr$tools_bar <- point_click[,2]
fctr$result_bar <- point_click[,3]
fctr$map_click <- point_click[,4]
fctr$labels <- inp[,12]
View(fctr)


# Разбиение на обучающую и тестовую выборки
split <- sample.split(fctr$labels, SplitRatio = 0.75)
dataTrain <- subset(fctr, split == TRUE)
dataTest <- subset(fctr, split == FALSE)

plot(fctr)
# Процесс обучения на всех факторах
logit <- glm(labels ~., data = dataTrain, family = 'binomial'(link='logit'))
summary(logit)

logit_mult <- glm(labels~query_bar*tools_bar*result_bar*map_click,
                  data = dataTrain, family = 'binomial'(link='logit'))
summary(logit_mult)



# step forward, backward & bothways для всех факторов
nothing <- glm(labels ~ 1, data = dataTrain, family=binomial)
summary(nothing)
```

```
forwards = step(nothing,
                scope=list(lower=formula(nothing),upper=formula(logit)), direction="forward")
formula(forwards)

backwards = step(logit)
formula(backwards)

bothways = step(nothing, list(lower=formula(nothing),upper=formula(logit)),
                direction="both",trace=0)
formula(bothways)

# для перемноженных

forwards_mult = step(nothing,
                scope=list(lower=formula(nothing),upper=formula(logit_mult)), direction="forward")
formula(forwards_mult)

bothways_mult = step(nothing, list(lower=formula(nothing),upper=formula(logit_mult)),
                direction="both",trace=0)
formula(bothways_mult)

# процесс предсказывания
logit_fwd <- glm(formula(forwards), data = dataTrain, family = 'binomial'(link='logit'))
summary(logit_fwd)

logit_bwd <- glm(formula(backwards), data = dataTrain, family = 'binomial'(link='logit'))
summary(logit_bwd)

logit_both <- glm(formula(bothways), data = dataTrain, family = 'binomial'(link='logit'))
summary(logit_both)


# процесс предсказывания для перемноженных
logit_fwd_mult <- glm(formula(forwards_mult), data = dataTrain, family = 'binomial'(link='logit'))
summary(logit_fwd_mult)

logit_both_mult <- glm(formula(bothways_mult), data = dataTrain, family = 'binomial'(link='logit'))
summary(logit_both_mult)


#baselines
baseline_1 <- matrix(data = 1, nrow = 1, ncol = 120, byrow = T)
baseline_rand <- matrix(data = 0:1, nrow = 1, ncol = 120, byrow = T)

# Процесс предсказывания
predictResult <- predict(logit,newdata=dataTest,type='response')
predictResult <- ifelse(predictResult > 0.5,1,0)

predictResult_fwd <- predict(logit_fwd,newdata=dataTest,type='response')
predictResult_fwd <- ifelse(predictResult_fwd > 0.5,1,0)

predictResult_bwd <- predict(logit_bwd,newdata=dataTest,type='response')
predictResult_bwd <- ifelse(predictResult_bwd > 0.5,1,0)

predictResult_both <- predict(logit_both,newdata=dataTest,type='response')
predictResult_both <- ifelse(predictResult_both > 0.5,1,0)



predictResult_mult <- predict(logit_mult,newdata=dataTest,type='response')
predictResult_mult <- ifelse(predictResult_mult > 0.5,1,0)

predictResult_fwd_mult <- predict(logit_fwd_mult,newdata=dataTest,type='response')
predictResult_fwd_mult <- ifelse(predictResult_fwd_mult > 0.5,1,0)

predictResult_both_mult <- predict(logit_both_mult,newdata=dataTest,type='response')
predictResult_both_mult <- ifelse(predictResult_both_mult > 0.5,1,0)

# Точность, чувствительность и специфичность
```

```
confusionMatrix(data = predictResult, reference = dataTest$labels)
confusionMatrix(data = predictResult_fwd, reference = dataTest$labels)
confusionMatrix(data = predictResult_bwd, reference = dataTest$labels)
confusionMatrix(data = predictResult_both, reference = dataTest$labels)

confusionMatrix(data = predictResult_mult, reference = dataTest$labels)
confusionMatrix(data = predictResult_fwd_mult, reference = dataTest$labels)
confusionMatrix(data = predictResult_both_mult, reference = dataTest$labels)

# wald-test

wald.test(b = coef(logit), Sigma = vcov(logit), L = 1)


# ROC-кривая
p <- predict(logit, newdata=dataTest, type="response")
pr <- prediction(p, dataTest$labels)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf, col='blue') # colorize=TRUE, print.cutoffs.at=seq(0,1,0.1), text.adj=c(-0.2,1.7)

p_fwd <- predict(logit_fwd, newdata=dataTest, type="response")
pr_fwd <- prediction(p_fwd, dataTest$labels)
prf_fwd <- performance(pr_fwd, measure = "tpr", x.measure = "fpr")
plot(prf_fwd, add=TRUE, col='red')

p_bwd <- predict(logit_bwd, newdata=dataTest, type="response")
pr_bwd <- prediction(p_bwd, dataTest$labels)
prf_bwd <- performance(pr_bwd, measure = "tpr", x.measure = "fpr")
plot(prf_bwd, add=TRUE, col='green')

p_both <- predict(logit_both, newdata=dataTest, type="response")
pr_both <- prediction(p_both, dataTest$labels)
prf_both <- performance(pr_both, measure = "tpr", x.measure = "fpr")
plot(prf_both, add=TRUE, col='black')



p_mult <- predict(logit_mult, newdata=dataTest, type="response")
pr_mult <- prediction(p_mult, dataTest$labels)
prf_mult <- performance(pr_mult, measure = "tpr", x.measure = "fpr")
plot(prf_mult, col='blueviolet') # colorize=TRUE, print.cutoffs.at=seq(0,1,0.1), text.adj=c(-0.2,1.7)

p_fwd_mult <- predict(logit_fwd_mult, newdata=dataTest, type="response")
pr_fwd_mult <- prediction(p_fwd_mult, dataTest$labels)
prf_fwd_mult <- performance(pr_fwd_mult, measure = "tpr", x.measure = "fpr")
plot(prf_fwd_mult, add=TRUE, col='pink')

p_both_mult <- predict(logit_both_mult, newdata=dataTest, type="response")
pr_both_mult <- prediction(p_both_mult, dataTest$labels)
prf_both_mult <- performance(pr_both_mult, measure = "tpr", x.measure = "fpr")
plot(prf_both_mult, add=TRUE, col='khaki4')

auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc

auc_fwd <- performance(pr_fwd, measure = "auc")
auc_fwd <- auc_fwd@y.values[[1]]
auc_fwd

auc_bwd <- performance(pr_bwd, measure = "auc")
auc_bwd <- auc_bwd@y.values[[1]]
auc_bwd

auc_both <- performance(pr_both, measure = "auc")
auc_both <- auc_both@y.values[[1]]
auc_both
```

```
auc_mult <- performance(pr_mult, measure = "auc")
auc_mult <- auc_mult@y.values[[1]]
auc_mult

auc_fwd_mult <- performance(pr_fwd_mult, measure = "auc")
auc_fwd_mult <- auc_fwd_mult@y.values[[1]]
auc_fwd_mult

auc_both_mult <- performance(pr_both_mult, measure = "auc")
auc_both_mult <- auc_both_mult@y.values[[1]]
auc_both_mult

######
# для удовлетворенности


# построение окончательной таблицы факторов
fctr <- transform(inp[,1:11],point_click)
fctr$query_bar <- point_click[,1]
fctr$tools_bar <- point_click[,2]
fctr$result_bar <- point_click[,3]
fctr$map_click <- point_click[,4]
fctr$labels <- inp[,13]
View(fctr)

# Разбиение на обучающую и тестовую выборки
split <- sample.split(fctr$labels, SplitRatio = 0.75)
dataTrain <- subset(fctr, split == TRUE)
dataTest <- subset(fctr, split == FALSE)

plot(fctr)
# Процесс обучения на всех факторах
logit <- glm(labels ~., data = dataTrain, family = 'binomial'(link='logit'))
summary(logit)

logit_mult <- glm(labels~query_bar*tools_bar*result_bar*map_click,
                  data = dataTrain, family = 'binomial'(link='logit'))
summary(logit_mult)

#Wald-test

wald.test(b = coef(logit), Sigma = vcov(logit), Terms = 1:14)


# step forward, backward & bothways для всех факторов
nothing <- glm(labels ~ 1, data = dataTrain, family=binomial)
summary(nothing)

forwards = step(nothing,
                scope=list(lower=formula(nothing),upper=formula(logit)), direction="forward")
formula(forwards)

backwards = step(logit)
formula(backwards)

bothways = step(nothing, list(lower=formula(nothing),upper=formula(logit)),
                direction="both",trace=0)
formula(bothways)

# для перемноженных

forwards_mult = step(nothing,
                     scope=list(lower=formula(nothing),upper=formula(logit_mult)), direction="forward")
formula(forwards_mult)

bothways_mult = step(nothing, list(lower=formula(nothing),upper=formula(logit_mult)),
                     direction="both",trace=0)
formula(bothways_mult)
```

```
# процесс предсказывания
logit_fwd <- glm(formula(forwards), data = dataTrain, family = 'binomial'(link='logit'))
summary(logit_fwd)

logit_bwd <- glm(formula(backwards), data = dataTrain, family = 'binomial'(link='logit'))
summary(logit_bwd)

logit_both <- glm(formula(bothways), data = dataTrain, family = 'binomial'(link='logit'))
summary(logit_both)


# процесс предсказывания для перемноженных
logit_fwd_mult <- glm(formula(forwards_mult), data = dataTrain, family = 'binomial'(link='logit'))
summary(logit_fwd_mult)

logit_both_mult <- glm(formula(bothways_mult), data = dataTrain, family = 'binomial'(link='logit'))
summary(logit_both_mult)


#baselines
baseline_1 <- matrix(data = 1, nrow = 1, ncol = 120, byrow = T)
baseline_rand <- matrix(data = 0:1, nrow = 1, ncol = 120, byrow = T)

# Процесс предсказывания
predictResult <- predict(logit,newdata=dataTest,type='response')
predictResult <- ifelse(predictResult > 0.5,1,0)

predictResult_fwd <- predict(logit_fwd,newdata=dataTest,type='response')
predictResult_fwd <- ifelse(predictResult_fwd > 0.5,1,0)

predictResult_bwd <- predict(logit_bwd,newdata=dataTest,type='response')
predictResult_bwd <- ifelse(predictResult_bwd > 0.5,1,0)

predictResult_both <- predict(logit_both,newdata=dataTest,type='response')
predictResult_both <- ifelse(predictResult_both > 0.5,1,0)



predictResult_mult <- predict(logit_mult,newdata=dataTest,type='response')
predictResult_mult <- ifelse(predictResult_mult > 0.5,1,0)

predictResult_fwd_mult <- predict(logit_fwd_mult,newdata=dataTest,type='response')
predictResult_fwd_mult <- ifelse(predictResult_fwd_mult > 0.5,1,0)

predictResult_both_mult <- predict(logit_both_mult,newdata=dataTest,type='response')
predictResult_both_mult <- ifelse(predictResult_both_mult > 0.5,1,0)

# Точность, чувствительность и специфичность
confusionMatrix(data = predictResult, reference = dataTest$labels)
confusionMatrix(data = predictResult_fwd, reference = dataTest$labels)
confusionMatrix(data = predictResult_bwd, reference = dataTest$labels)
confusionMatrix(data = predictResult_both, reference = dataTest$labels)

confusionMatrix(data = predictResult_mult, reference = dataTest$labels)
confusionMatrix(data = predictResult_fwd_mult, reference = dataTest$labels)
confusionMatrix(data = predictResult_both_mult, reference = dataTest$labels)


# ROC-кривая
p <- predict(logit, newdata=dataTest, type="response")
pr <- prediction(p, dataTest$labels)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf, col='blue') # colorize=TRUE, print.cutoffs.at=seq(0,1,0.1), text.adj=c(-0.2,1.7)

p_fwd <- predict(logit_fwd, newdata=dataTest, type="response")
pr_fwd <- prediction(p_fwd, dataTest$labels)
prf_fwd <- performance(pr_fwd, measure = "tpr", x.measure = "fpr")
plot(prf_fwd, add=TRUE, col='red')
```

```
p_bwd <- predict(logit_bwd, newdata=dataTest, type="response")
pr_bwd <- prediction(p_bwd, dataTest$labels)
prf_bwd <- performance(pr_bwd, measure = "tpr", x.measure = "fpr")
plot(prf_bwd, add=TRUE, col='green')

p_both <- predict(logit_both, newdata=dataTest, type="response")
pr_both <- prediction(p_both, dataTest$labels)
prf_both <- performance(pr_both, measure = "tpr", x.measure = "fpr")
plot(prf_both, add=TRUE, col='black')



p_mult <- predict(logit_mult, newdata=dataTest, type="response")
pr_mult <- prediction(p_mult, dataTest$labels)
prf_mult <- performance(pr_mult, measure = "tpr", x.measure = "fpr")
plot(prf_mult, col='blueviolet') # colorize=TRUE, print.cutoffs.at=seq(0,1,0.1), text.adj=c(-0.2,1.7)

p_fwd_mult <- predict(logit_fwd_mult, newdata=dataTest, type="response")
pr_fwd_mult <- prediction(p_fwd_mult, dataTest$labels)
prf_fwd_mult <- performance(pr_fwd_mult, measure = "tpr", x.measure = "fpr")
plot(prf_fwd_mult, add=TRUE, col='pink')

p_both_mult <- predict(logit_both_mult, newdata=dataTest, type="response")
pr_both_mult <- prediction(p_both_mult, dataTest$labels)
prf_both_mult <- performance(pr_both_mult, measure = "tpr", x.measure = "fpr")
plot(prf_both_mult, add=TRUE, col='khaki4')

auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc

auc_fwd <- performance(pr_fwd, measure = "auc")
auc_fwd <- auc_fwd@y.values[[1]]
auc_fwd

auc_bwd <- performance(pr_bwd, measure = "auc")
auc_bwd <- auc_bwd@y.values[[1]]
auc_bwd

auc_both <- performance(pr_both, measure = "auc")
auc_both <- auc_both@y.values[[1]]
auc_both


auc_mult <- performance(pr_mult, measure = "auc")
auc_mult <- auc_mult@y.values[[1]]
auc_mult

auc_fwd_mult <- performance(pr_fwd_mult, measure = "auc")
auc_fwd_mult <- auc_fwd_mult@y.values[[1]]
auc_fwd_mult

auc_both_mult <- performance(pr_both_mult, measure = "auc")
auc_both_mult <- auc_both_mult@y.values[[1]]
auc_both_mult
```