САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ - ПРОЦЕССОВ УПРАВЛЕНИЯ
Кафедра математической теории игр и статистических решений

Смирнов Дмитрий Сергеевич

Выпускная квалификационная работа бакалавра

Статистический анализ эффективности контекстной рекламы

Направление 010400

Прикладная математика и информатика

Заведующий кафедрой,
доктор физ.-мат. наук,
профессор Петросян Л. А.

Научный руководитель,
кандидат физ.-мат. наук,
доцент Громова Е. В.

Рецензент,
кандидат экономических наук,
ассистент кафедры операционного менеджмента
ВШМ Гладкова М. А.

Санкт-Петербург
2015

Содержание

1. Введение

1.1. Актуальность работы

В наше время практически каждая успешная компания имеет представительство в интернете. С каждым годом все больше и больше российских компаний приходят в интернет в поисках новых каналов продаж товаров и услуг. По прогнозам [1] в следующие 5 лет объем рынка электронной коммерции в России будет увеличиваться на $15-20\%$ ежегодно и в 2020 году составит более 2180 миллиардов рублей.

Наряду с развитием рынка электронной коммерции развивается и рынок интернет-рекламы. Согласно исследованиям [2] на сегодняшний день оборот рынка интернет-рекламы имеет самый высокий темп роста в сравнении с другими видами рекламы: в СМИ, на телевидение, на радио.

Вместе с ростом рынка интернет-рекламы увеличиваются и рекламные бюджеты. От эффективного расходования рекламных средств зависит объем продаж и, соответственно, прибыль компании. Актуальность данной работы определяется необходимостью рационального распределения бюджета, выделяемого на рекламу в интернете.

1.2. Предмет исследования

Среди различных видов интернет-рекламы выделяется *контекстная реклама*, которая по всем важным показателям: оборот рынка, число рекламодателей и т. д. значительно превосходит остальные виды интернет-рекламы [2].

Контекстная реклама — это вид интернет-рекламы, при котором рекламные объявления показываются в соответствии с содержимым интернет-страницы. Контекстная реклама действует избирательно и отображается посетителям интернет-страницы, сфера интересов которых потенциально совпадает или пересекается с тематикой рекламируемого товара или услуги, что повышает вероятность их отклика на рекламу.

В зависимости от формата рекламного объявления контекстная реклама разделяется на текстовую, баннерную и видеорекламу, а в зависимости от места размещения — на поисковую и тематическую. В данной работе под контекстной рекламой будем понимать *текстовую поисковую рекламу.*

Текстовая поисковая реклама — это реклама, показывающаяся на страницах результатов поиска популярных поисковых систем (Google, Yandex, Mail) после введения поискового запроса. В данной работе будет рассматриваться контекстная реклама, показывающаяся на страницах поиска поисковой системы Яндекс. Система размещения и управления рассматриваемой контекстной рекламы называется *Яндекс.Директ*.

На страницах результатов поиска в поисковой системе Яндекс находятся два рекламных блока (см. рис 1). Первый блок, называемый *спецразмещением*, содержит до трех рекламных объявлений и находится выше результатов поиска. Второй блок, называемый *гарантией*, находится ниже результатов поиска и содержит до четырех рекламных объявлений.

Обязательными атрибутами рекламного объявления являются заголовок,

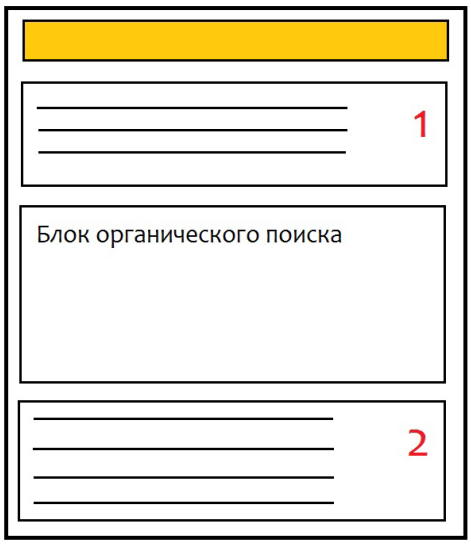

Рис. 1: Схема расположения рекламных блоков на странице поиска

описание и ссылка на сайт (см. рис. 2). При переходе по рекламному объявлению пользователь попадает на сайт, где ему предлагается товар или услуга, описываемая в объявлении.

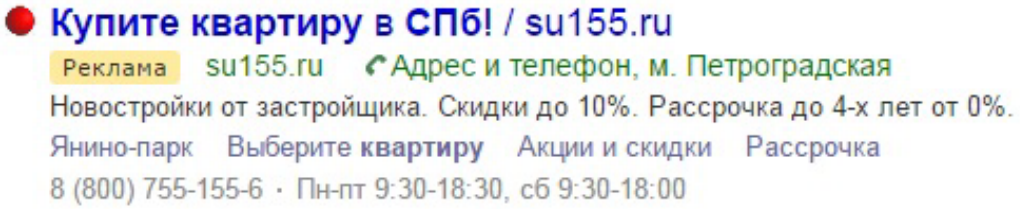

Рис. 2: Пример рекламного объявления

Каждое рекламное объявление в системе Яндекс.Директ размещается по нескольким поисковым запросам. Обозначим их $z_1, .., z_n$. Для каждого поискового запроса z_i рекламодатель назначает *ставку* p_i — цену за один переход по рекламному объявлению. Рекламодатель платит системе Яндекс.Директ за каждый переход по рекламному объявлению.

Ранжирование рекламных объявлений на странице поиска поисковой системы Яндекс происходит по аукционному принципу [3]. Объявления внутри рекламного блока ранжируются по величине ставки: чем больше ставка, тем

выше показывается рекламное объявление. Отбор в рекламный блок производится в порядке убывания величины

$$\frac{X_i}{Y_i} \cdot p_i,$$

где X_i — число переходов по рекламному объявлению, показывающемуся по запросу z_i, Y_i — число показов рекламного объявления по запросу z_i, p_i — величина ставки, соответствующая запросу z_i. Величина $\frac{X_i}{Y_i}$ называется *ctr* (от английского click through rate). В данной работе будем обозначать $ctr_i = \frac{X_i}{Y_i}$ и $ctr = \frac{\sum_{i=1}^{n} X_i}{\sum_{i=1}^{n} Y_i}$.

Таким образом, при прочих равных условиях рекламное объявление, показывающееся по запросу z_i и имеющее значение ctr_i больше, чем у других рекламных объявлений, размещается выше на странице результатов поиска и, соответственно, имеет большее число переходов. Поэтому в качестве метрики эффективности в последующих главах работы будем использовать величину ctr.

В главе 1 рассматривается метод A/B-тестирования двух рекламных объявлений с применением методов математической статистики. В работе предлагается алгоритм анализа данных, на основе результатов которого делается выбор более «успешного» рекламного объявления. Для реализации алгоритма анализа данных написан программный код на языке C++ в среде Visual Studio 2013.

В главе 2 рассматривается ситуация, при которой по одному поисковому запросу z_i сайт находится как в рекламном блоке спецразмещения, так и в блоке органического поиска. Ставится задача определить, как влияет наличие сайта по запросу z_i в блоке органического поиска на величину ctr_i рекламного объявления, показывающегося в спецразмещении по запросу z_i. Для решения задачи были рассмотрены рекламные кампании в системе Яндекс.Директ, со-

браны данные и проведен анализ, на основе результатов которого делается вывод.

В главе 3 исследуется проблема тестирования интернет-страниц. Задача тестирования формализуется в виде математической модели «задачи о многоруком бандите». Рассматриваются известные алгоритмы решения задачи. В качестве демонстрации применимости алгоритмов предложена программная симуляция показа страниц. Разработан программный код на языке Python.

1.3. Обзор литературы

Статьи, посвященные A/B-тестированию, наиболее широко представлены в сети интернет, например [5, 4]. В подавляющем большинстве найденных материалов не уделяется должного внимания анализу данных, полученных после проведения тестирования. Данный анализ играет ключевую роль и должен производится в строгом соответствии с методами математической статистики.

Анализ данных в первой и второй главе построен на применении известных статистических критериев, таких как критерий Пирсона, Фишера–Снедекора, Стьюдента, Манна–Уитни, которые подробно описаны в [6]. Кроме того, в некоторых случаях использовался критерий Кохрана–Кокса [7], являющийся аналогом критерия Стьюдента для случая выборок с неравными дисперсиями.

В основе третьей главы лежит так называемая задача о многоруком бандите, которая, к сожалению, крайне скудно представлена в русскоязычной литературе, в связи с чем при исследовании и написании работы использовались в основном англоязычные источники. Постановка задачи и описание некоторых подходов к решению описаны в [8]. В [9] вводится функция, называемая *сожалением*, которая характеризует работу алгоритмов для решения задачи о многоруком бандите. Там же доказано, что функция сожаления R_T асимптотически не меньше функции $\log T$. Описание используемых в работе алгоритмов и демонстрация их работы представлены в [10]. В [11] описан алгоритм UCB1 и даны оценки функции сожаления R_T некоторых алгоритмов.

2. Глава 1. A/B-тестирование двух рекламных объявлений

2.1. Постановка задачи

A/B-тестирование (A/B-testing, Split testing) — один из популярных методов маркетингового исследования. Суть метода состоит в том, что аудитория пользователей делится на две равные по числу группы (группа A, группа B). Затем происходит взаимодействие группы A с одной версией ресурса и группы B с другой. В результате этого взаимодействия определяется лучшая в некотором смысле версия ресурса. Для качественного анализа результатов версии ресурса должны отличаться одним элементом. Наиболее широкое использование A/B-тест нашел в тестировании веб-страниц с целью увеличения конверсии.

При A/B-тестировании рекламных объявлений группе A и группе B показываются разные версии одного объявления (например, в одной версии изменено описание или заголовок). В ходе проведения эксперимента ежедневно фиксируется статистика значений ctr двух объявлений (см. таблицу 1). После проведения эксперимента необходимо проанализировать полученные данные и сделать вывод о том, какое объявление «успешней», то есть определить, для какого объявления среднее значение ctr больше.

Цель исследования — формализовать анализ данных, полученных после проведения теста, разработать алгоритм для этого анализа и провести реальный эксперимент A/B-тестирования двух рекламных объявлений.

№	ctr 1-го объявления (X)	ctr 2-го объявления (Y)
1	x_1	y_1
2	x_2	y_2
3	x_3	y_3
...	...	...

Таблица 1: Данные после проведения теста

2.2. Алгоритм анализа данных

Решение поставленной задачи сводится к решению известной задачи проверки однородности двух независимых выборок, которая может быть решена как параметрическими, так и непараметрическими методами. Непараметрические методы не требуют знания вида закона распределения и более просты в расчетах, однако мощность статистических критериев, построенных на их основе, уступает аналогичным параметрическим критериям [7]. Поэтому для проверки однородности выборок решено использовать критерий Стьюдента [6], который относится к параметрическим. Однако для его применения необходимо проверить гипотезу о том, что выборки извлечены из генеральной совокупности, имеющей нормальное распределение. Поэтому в случаях, когда эта гипотеза не подтверждается, следует обращаться к непараметрическим критериям.

Для применения критерия Стьюдента необходимо задать уровень значимости α и выдвинуть нулевую гипотезу $H_0 : E[X] = E[Y]$. Для определения альтернативной гипотезы H_1 следует сравнить средние значения выборок и принять $H_1 : E[X] > E[Y]$, если $\overline{X} > \overline{Y}$ и $H_1 : E[X] < E[Y]$, если $\overline{X} < \overline{Y}$.

Статистика критерия Стьюдента имеет вид

$$\varphi(X_{[n]}, Y_{[m]}) = \frac{\overline{x} - \overline{y}}{\widehat{S} \cdot \sqrt{\frac{1}{n} + \frac{1}{m}}},$$

$$\widehat{S} = \sqrt{\frac{(n-1)\widetilde{s_1}^2 + (m-1)\widetilde{s_2}^2}{n+m-2}},$$

где $\widetilde{s_1}^2, \widetilde{s_2}^2$ — исправленные выборочные дисперсии. Критический интервал зависит от вида альтернативной гипотезы и для односторонних гипотез принимает 2 формы (см. таблицу 2).

В таблице $t_{1-\alpha,n+m-2}$ — квантиль распределения Стьюдента с $(n+m-2)$

H_1	Критический интервал
$E[X] > E[Y]$	$(t_{1-\alpha,n+m-2}, +\infty)$
$E[X] < E[Y]$	$(-\infty, -t_{1-\alpha,n+m-2})$

Таблица 2: Критические интервалы

степенями свободы уровня $(1-\alpha)$.

Как уже частично отмечалось, для применения критерия Стьюдента необходимо, чтобы выполнялись два условия: во-первых, выборки должны быть извлечены из нормальной генеральной совокупности, во-вторых, эти генеральные совокупности должны иметь равные дисперсии.

Для проверки первого условия следует воспользоваться критерием Пирсона [6]. Нулевая гипотеза критерия имеет вид $H_0 : F(\cdot) = F_0(\cdot, \theta)$, альтернативная гипотеза — $H_1 : F(\cdot) \neq F_0(\cdot, \theta)$, где $F_0(\cdot, \theta)$ — функция нормального распределения, определенная с точностью до параметров. Перед проверкой гипотезы необходимо найти оценки неизвестных параметров распределения.

Статистика критерия имеет вид

$$\gamma_{\chi^2}(X_{[n]}) = \sum_{i=1}^{r} \frac{(n_i - np_i^{(0)}(\hat{\theta}))^2}{n \cdot p_i^{(0)}}.$$

Критическая область определяется интервалом $(\chi^2_{1-\alpha,r-l-1}, \infty)$, где α — уровень значимости, r — число интервалов разбиения, l — число оцениваемых параметров, а $\chi^2_{1-\alpha,r-l-1}$ — квантиль уровня $(1-\alpha)$ распределения χ^2 с $(r-l-1)$ степенями свободы.

Выполнение второго условия, необходимого для применения критерия Стьюдента, будем проверять критерием Фишера–Снедекора [6].

Нулевая гипотеза критерия $H_0 : \sigma_1^2 = \sigma_2^2$, альтернативная гипотеза $H_1 : \sigma_1^2 \neq \sigma_2^2$. Статистика критерия имеет вид

$$F(X_{[n]}, Y_{[m]}) = \frac{\widetilde{s_1}^2}{\widetilde{s_2}^2},$$

где $\widetilde{s_1}^2$ и $\widetilde{s_2}^2$ — исправленные выборочные дисперсии первой и второй выборки соответственно. Критическая область для данной статистики задается интервалом

$$[0; F_{\frac{\alpha}{2},n-1,m-1}) \cup (F_{1-\frac{\alpha}{2},n-1,m-1}; +\infty),$$

где α — уровень значимости, $F_{\alpha,n,m}$ — квантиль уровня α распределения Фишера с (n, m) степенями свободы.

Рассмотрим случаи, когда гипотезы о нормальности закона распределения и о равенстве дисперсии не подтверждаются. В первом случае применим непараметрический аналог критерия Стьюдента, который носит название критерия Манна–Уитни [6]. Для использования этого критерия формулируется нулевая гипотеза вида $H_0 : F(\cdot) = G(\cdot)$, где $F(\cdot), G(\cdot)$ — непрерывные функции распределения генеральных совокупностей, из которых извлечены соответственно первая и вторая выборки. В качестве альтернативной гипотезы при $\overline{X} > \overline{Y}$ рассмотрим $F(\cdot) \geq G(\cdot)$ и при $\overline{X} < \overline{Y}$ — $F(\cdot) \leq G(\cdot)$. Статистика критерия имеет вид

$$U(X_{[n]}, Y_{[m]}) = \sum_{i=1}^{n} \sum_{j=1}^{m} I(x_i < y_j),$$

где

$$I(x_i < y_j) = \begin{cases} 1, & \text{если } x_i < y_j, \\ 0, & \text{если } x_i \geq y_j; \end{cases}$$

При $n, m > 8$ статистика критерия аппроксимируется статистикой

$$U^*(X_{[n]}, Y_{[m]}) = \frac{U(X_{[n]}, Y_{[m]}) - \frac{mn}{2}}{\sqrt{\frac{mn(m+n+1)}{12}}},$$

которая в случае справедливости гипотезы H_0 асимптотически подчиняется стандартному нормальному распределению. Критические области в зависимо-

сти от альтернативной гипотезы H_1 даны в таблице 3, в которой u_α — квантиль стандартного нормального распределения уровня α.

H_1	Критический интервал
$F(z) \geq G(z)$	$(u_{1-\alpha}, +\infty)$
$F(z) \leq G(z)$	$(-\infty, -u_{1-\alpha})$

Таблица 3: Критические интервалы

В случае, когда выборки извлечены из нормальных генеральных совокупностей, имеющих неравные дисперсии, применим критерий Кохрана–Кокса [7]. Гипотезы для критерия Кохрана–Кокса формулируются аналогично гипотезам для критерия Стьюдента. Статистика критерия имеет вид

$$t_K = \frac{1}{s}(\bar{x} - \bar{y}),$$

где $s^2 = \frac{\widetilde{s}_1^2}{n} + \frac{\widetilde{s}_2^2}{m}$, а $\widetilde{s}_1^2$ и $\widetilde{s}_2^2$ – исправленные выборочные дисперсии первой и второй выборки соответственно. Критические значения статистики вычисляются по формуле

$$t'_\alpha = \frac{\nu_1 t_\alpha(n-1) + \nu_2 t_\alpha(m-1)}{\nu_1 + \nu_2},$$

где $\nu_1 = \frac{\widetilde{s}_1^2}{n}$, $\nu_2 = \frac{\widetilde{s}_2^2}{m}$, $t_\alpha(f) - \alpha$–квантиль распределения Стьюдента с f степенями свободы. Критические области критерия зависят от вида альтернативной гипотезы и представлены в таблице 4.

H_1	Критический интервал
$E[X] > E[Y]$	$(t'_{1-\alpha,n-1}, +\infty)$
$E[X] < E[Y]$	$(-\infty, -t'_{1-\alpha,m-1})$

Таблица 4: Критические интервалы

Таким образом имеем следующий алгоритм анализа данных, полученных после проведения A/B-тестирования двух рекламных объявлений.

1. Задать уровень значимости α.

2. С помощью критерия Пирсона проверить данные на нормальность. Если выборки извлечены из нормальной генеральной совокупности, перейти к следующему шагу, в противном случае воспользоваться критерием Манна–Уитни.

3. С помощью критерия Фишера–Снедекора проверить равенство дисперсий двух выборок. В случае, если дисперсии равны, перейти к шагу 4, иначе воспользоваться критерием Кохрана–Кокса.

4. Применить критерий Стьюдента. Сделать вывод.

В случаях, когда альтернативная гипотеза о неоднородности двух выборок отклоняется, рекомендуется продолжить тестирование.

Для автоматизации анализа данных, полученных после проведения теста, реализована программа на языке C++ (см. приложение 1). На вход программе подаются две выборки и уровень значимости α. Программа с помощью описанного выше алгоритма определяет объявление, имеющее большее среднее значение ctr.

2.3. Пример анализа данных

После проведения теста, который длился 50 дней, были получены данные (см. приложение 2). На основании анализа данных необходимо выявить объявление, имеющее большее среднее значение ctr. Первую выборку обозначим через X, вторую — через Y. Согласно алгоритму зададим уровень значимости, на котором будем проверять гипотезы. Пусть $\alpha = 0.05$.

С помощью критерия Пирсона проверим гипотезу о том, что данные извлечены из генеральной совокупности, имеющей нормальное распределение.

Выдвинем гипотезу $H_0 : F(\cdot) = F_0(\cdot, \theta)$ при альтернативной $H_1 : F(\cdot) \neq F_0(\cdot, \theta)$, где $F_0(\cdot, \theta)$- функция нормального распределения, определенная с точностью до параметров. Для первой и второй выборки соответственно значения статистики $\gamma_{\chi^2}(X_{[n]})$ равны

$$\gamma_{\chi^2}(X_{[n]}) = 4,637,$$

$$\gamma_{\chi^2}(Y_{[n]}) = 2,069.$$

Для заданного уровня значимости критические области для первой и второй выборки совпадают

$$(5,991, +\infty).$$

Как видно, значения статистики критерия для первой и второй выборки не попадают в критическую область, следовательно, можно сделать вывод о том, что выборки извлечены из генеральной совокупности, имеющей нормальное распределение.

Теперь с помощью критерия Фишера–Снедекора проверим гипотезу о равенстве дисперсий генеральных совокупностей, из которых извлечены рассматриваемые выборки. Выдвинем гипотезу $H_0 : \sigma_1^2 = \sigma_2^2$ при альтернативной гипотезе $H_1 : \sigma_1^2 \neq \sigma_2^2$. Статистика критерия равна

$$F(X_{[n]}, Y_{[m]}) = 1,226.$$

Критический интервал имеет вид

$$[0; 0,567)U(1,762; +\infty).$$

Снова значение статистики не попадает в критический интервал, значит, можно сделать вывод о том, что дисперсии генеральных совокупностей равны.

Теперь, после успешной проверки гипотез о виде закона распределения и равенстве дисперсий, применим критерий Стьюдента и окончательно решим

задачу. Выдвинем нулевую гипотезу $H_0 : E[X] = E[Y]$. Для формулирования альтернативной гипотезы найдем средние значения двух выборок $\overline{X} = 0,076$, $\overline{Y} = 0,098$. Так как $\overline{Y} > \overline{X}$, то альтернативная гипотеза имеет вид $E[X] < E[Y]$. Значение статистики критерия равно

$$F(X_{[n]}, Y_{[m]}) = -3,823.$$

На заданном уровне значимости критический интервал имеет вид

$$(-\infty, -1,6606).$$

Значение статистики критерия попадает в критический интервал, следовательно, принимаем альтернативную гипотезу. Это значит, что среднее значение второй выборки больше среднего значения первой, следовательно, второе рекламное объявление имеет большее среднее значение ctr. Таким образом, для рекламной кампании следует выбрать вторую версию объявления, как наиболее «успешную».

3. Глава 2. Анализ структуры рекламного блока

3.1. Постановка задачи

Определить, как влияет наличие сайта по запросу z_i в блоке органического поиска на ctr_i рекламного объявления, показывающегося в спецразмещении по запросу z_i.

3.2. Решение задачи

Для получения данных были рассмотрены реальные рекламные кампании. В каждом рекламном объявлении выбирались значения ctr_i тех запросов z_i, по которым сайт занимает одну из первых трех позиций в блоке органического поиска. Таким значениям ctr_i сопоставлялось среднее значение (в рамках одного объявления) ctr запросов, по которым сайт не занимает одну из первых трех позиций поиска и количество показов X_i по каждому запросу z_i больше 10 (см. таблицу 5). Таблица данных в приложении 3.

№	$ctr(pos = 1, 2, 3)$	$\overline{ctr}$
Объявление №1	x_1	y_1
Объявление №2	x_2	y_2
Объявление №3	x_3	y_3
...	...	...

Таблица 5: Вид полученных данных

Таким образом получили две выборки объемом $n = 85$. Первую выборку ctr запросов, для которых сайт находится в поиске на одной из первых трех позиций, обозначим через X, вторую выборку — через Y.

Для решения поставленной задачи определим, какая выборка извлечена из генеральной совокупности, имеющей большее значение математического

ожидания.

Для решения этой задачи вновь воспользуемся параметрическими критериями. Сначала проверим гипотезу о нормальном законе распределения генеральных совокупностей, из которых извлечены выборки.

Для этого воспользуемся критерием Пирсона. Рассмотрим первую выборку и выдвинем нулевую гипотезу $H_0 : F(\cdot) = F_0(\cdot, \theta)$, при альтернативной гипотезе $H_1 : F(\cdot) \neq F_0(\cdot, \theta)$, где $F_0(\cdot, \theta)$ — функция нормального распределения, определенная с точностью до параметров математического ожидания и среднеквадратичного отклонения.

Предварительно оценив данные параметры по методу максимального правдоподобия [6], найдем значение статистики критерия Пирсона:

$$\gamma_{\chi^2}(X_{[n]}) = 5,2404.$$

Вычислим критические области для разных уровней значимости (см. таблицу 6):

α	Критическая область
$0,05$	$(7,8147, +\infty)$
$0,01$	$(11,3449, +\infty)$
$0,001$	$(16,2662, +\infty)$

Таблица 6: Критические интервалы

При всех рассмотренных уровнях значимости значение статистики $\gamma_{\chi^2}(X_{[n]})$ не попадает в критическую область, а значит, можно утверждать, что первая выборка извлечена из нормальной генеральной совокупности.

Аналогично проверим вторую выборку. Вычислим значение статистики критерия

$$\gamma_{\chi^2}(Y_{[n]}) = 3,8393$$

Критические области для различных уровней значимости представлены в таблице 7.

α	Критическая область
$0,05$	$(3,8415,+\infty)$
$0,01$	$(6,6348,+\infty)$
$0,001$	$(10,8276,+\infty)$

Таблица 7: Критические интервалы

Вновь значение статистики $\gamma_{\chi^2}(Y_{[n]})$ не попадает в критическую область, следовательно, нет оснований отклонить нулевую гипотезу, а значит, можно считать, что и вторая выборка извлечена из нормальной генеральной совокупности.

Наиболее мощным критерием, подходящим для решения поставленной задачи, является критерий Стьюдента. Однако для его применения необходимо проверить гипотезу о равенстве дисперсий двух нормально распределенных генеральных совокупностей. Для проверки данной гипотезы воспользуемся критерием Фишера–Снедекора. Выдвинем нулевую гипотезу $H_0 : \sigma_1^2 = \sigma_2^2$ при альтернативной $H_1 : \sigma_1^2 \neq \sigma_2^2$. Статистика критерия имеет вид

$$F(X_{[n]}, Y_{[m]}) = \frac{\widetilde{s_1}^2}{\widetilde{s_2}^2} = 3,53,$$

где $\widetilde{s_1}^2$ и $\widetilde{s_2}^2$ — исправленные выборочные дисперсии первой и второй выборки соответственно.

При уровне значимости $\alpha = 0,05$ критический интервал примет вид

$$[0; 0,61) \cup (1,63; +\infty).$$

Видно, что значение статистики попадает в критический интервал, следовательно, нулевая гипотеза H_0 отвергается, а значит, можно сделать вывод о том,

что дисперсии двух исследуемых случайных величин не равны. Таким образом, применение к выборкам критерия Фишера–Снедекора позволяет сделать вывод о невозможности применения критерия Стьюдента.

Задача сравнения средних двух нормально распределенных совокупностей при неизвестных и неравных (по выборочным оценкам) дисперсиях носит название задачи Беренса – Фишера [7]. Точное решение данной задачи на настоящий момент не получено. На практике обычно используются различные приближения. Как и в первой главе воспользуемся критерием Кохрана–Кокса.

Выдвинем гипотезу $H_0 : EX = EY$ при альтернативной гипотезе $H_1 : EX > EY$. Найдем значение статистики критерия

$$t_K = 4,342.$$

Рассмотрим различные уровни значимости и вычислим критические интервалы (см. таблицу 8).

α	Критическая область
$0,05$	$[1,645, +\infty)$
$0,01$	$[2,326, +\infty)$
$0,001$	$[3,09, +\infty)$

Таблица 8: Критические интервалы

Таким образом, при рассмотренных уровнях значимости значение статистики попадает в критическую область, следовательно, справедлива альтернативная гипотеза, а, значит, можно утверждать, что, для запросов z_i, по которым сайт находится на одной из первых трех позиций в поиске, ctr_i рекламных объявлений больше, чем средний ctr объявлений, показывающихся по запросам, для которых сайт не находится на одном из первых трех мест поиска.

В ходе исследования были проанализированы данные, полученные из реальных рекламных кампаний. В результате анализа выяснилось, что в случае,

когда объявление показывается по запросу, для которого сайт находится на одном из первых трех мест в поиске, в среднем оно имеет большее значение ctr_i, чем значение ctr этого объявления, показывающегося по запросам, по которым сайт не занимает одно из первых трех мест. Это значит, что рекламодателю выгодно размещать объявления по запросам, по которым сайт высоко ранжируется в поиске. Полученный результат можно использовать при настройке и оптимизации рекламных кампаний в Яндекс.Директ.

4. Глава 3. Тестирование интернет-страниц на основе задачи о многоруком бандите

4.1. Постановка задачи

Тема, затронутая в настоящей главе, лишь косвенно связана с контекстной рекламой, однако важна с точки зрения её эффективности. Поскольку в конечном итоге показателем эффективности рекламы является количество новых клиентов, то важно, чтобы страница, на которую пользователь пришел с контекстной рекламы, с максимальным успехом «превращала» его в клиента, другими словами, имела высокую *конверсию*.

Конверсия — это отношение числа визитов пользователей, в которых были выполнены целевые действия, к общему числу визитов. Целевые действия определяются исходя из коммерческой составляющей страницы. Например, если на странице продается товар, то естественным будет определить целевое действие, как нажатие кнопки «купить».

Тестирование интернет-страниц проводится с целью увеличить конверсию. Перед тестированием страницы задается определенная цель (например, клик по некоторой кнопке), показатели достижения которой отслеживаются. Пользователям поочередно показываются разные версии одной страницы (например, в одной версии может быть изменена цветовая гамма, расположение элементов и т.д). После проведения теста сравниваются показатели достижения целей и выявляется страница с большей конверсией.

Подход к тестированию, описанный выше, довольно популярен, однако имеет существенный недостаток. Во время теста в равных пропорциях показываются страницы с низкой и высокой конверсией. Показ страниц с низкой конверсией несет убыток организатору теста. Компания, кроме выявления

страницы с наибольшей конверсией, заинтересована в том, чтобы во время проведения теста достичь наибольшую суммарную конверсию. Именно эту задачу решает подход к тестированию, основанный на решении задачи о многоруком бандите.

Задача о многоруком бандите [10] – частный случай дилеммы исследования-использования (exploration-exploitation [8]), которая может быть описана следующим образом. Имеется система взаимодействия игрока с окружающей средой. Игрок имеет набор действий для этого взаимодействия. После каждого действия игрок получает некоторый выигрыш. Перед очередным взаимодействием с окружающей средой игрок имеет выбор: выполнить уже известное действие и получить выигрыш близкий к ожидаемому или выполнить новое действие и получить новую информацию. Фаза, в которой игрок выбирает хорошо знакомое действие, называется *фазой использования*. Если игрок выбирает действие, которое ему мало знакомо или не знакомо вовсе, говорят, что игрок находится в *фазе исследования*. Цель игрока — достичь максимальный суммарный выигрыш.

Задача о многоруком бандите впервые рассмотрена Гербертом Роббинсом (Herbert Ellis Robbins) в 1952 году. И с тех пор широко используется для моделирования задач вида исследования-использования. Математическая модель задачи строится следующим образом.

Рассмотрим K игровых автоматов. С каждым автоматом связаны случайные величины $X_{i,n}$, где $1 \leq i \leq K$, $n \geq 1$, то есть i — индекс автомата. При каждой игре на автомате случайная величина генерирует выигрыш. Например, играя на автомате i игрок будет получать выигрыши $X_{i,1}, X_{i,2}, ...$ — реализации независимых случайных величин, имеющих общий неизвестный закон распределения с неизвестным математическим ожиданием μ_i. Независимость распределений имеет место и для выигрышей разных автоматов, то есть

$X_{i,s}$ и $X_{j,t}$ независимы (и как правило имеют разные законы распределения) для любых $1 \leq i < j \leq K$ и $s, t \geq 1$. Цель игрока — достичь максимальный выигрыш за время игры.

Под стратегией (policy, allocation strategy) A будем понимать алгоритм выбора следующего автомата для игры на основе последовательности предыдущих игр и полученных выигрышей. Основной характеристикой сравнения стратегий является *функция потерь.*

Пусть $T_i(n)$ — это число игр, сыгранных на автомате i с использованием стратегии A в течение первых n игр. Тогда функция потерь стратегии A после n сыгранных игр определяется как

$$R_T = \mu^* n - \mu_j \sum_{j=1}^{K} E[T_j(n)],$$

где $\mu^* = \max_{1 \leq i \leq K} \mu_i$.

Если бы распределения выигрышей были известны, то стратегия игры была бы тривиальна: всегда играть на автомате, который имеет максимальное математическое ожидание выигрыша.

Доказано [9], что для функции потерь справедлива оценка $R_T = \Omega(\log T)$, где T — количество сыгранных игр. Алгоритмы, которые достигают этой оценки, называются *оптимальными*. В дальнейшем будут использоваться следующие обозначения: $\widehat{\mu}_i(t)$ — средняя награда автомата i после t игр, $p_i(t)$ — вероятность игры на автомате i во время t.

4.2. Описание алгоритмов

4.2.1. ε–жадный алгоритм (ε–greedy)

Алгоритм ε–greedy [10] относится к так называемым *жадным* алгоритмам, которые отличаются тем, что на каждом этапе оптимизационной задачи выбирают локально оптимальное решение, допуская, что конечное решение также окажется оптимальным.

В силу простоты реализации ε–жадный алгоритм широко используется для решения задачи о многоруком бандите. Суть алгоритма состоит в том, что в каждый момент времени $t = 1, 2, \ldots$ алгоритм выбирает автомат с наибольшим средним выигрышем с вероятностью $1 - \varepsilon$ и с вероятностью ε выбирает случайный автомат среди всех. Формально, пусть после t игр средние выигрыши автоматов равны $\widehat{\mu}_1(t), \ldots, \widehat{\mu}_K(t)$, тогда вероятность выбора автомата с номером i в следующий момент времени вычисляется по формуле

$$p_i(t+1) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{K}, & \text{если } i = \arg\max_{j=1,\ldots,K} \mu_j(t); \\ \frac{\varepsilon}{K}, & \text{в противном случае.} \end{cases}$$

На каждом шаге с вероятностью $1 - \varepsilon$ алгоритм находится в фазе использования и с вероятностью ε — в фазе исследования. Таким образом, параметр ε отвечает за частоту появления фазы исследования. При $\varepsilon = 1$ получим стратегию, при которой номер автомата для следующей игры выбирается случайно, а при $\varepsilon = 0$ получим так называемый «чистый» жадный алгоритм, который никогда не будет иметь фазу исследования.

ε–жадный алгоритм обладает существенным недостатком: вероятность попасть в фазу исследования в любой момент времени одинакова, тогда как более логичным и естественным является уменьшение этой вероятности с течением времени. Этот момент учтен в модификации данного алгоритма, кото-

рый называется ε_n–greedy.

4.2.2. Модифицированный ε–жадный алгоритм (ε_n–greedy)

Алгоритм ε_n–greedy [11, 12] отличается от предыдущего алгоритма тем, что на каждом шаге вероятность ε вычисляется по формуле

$$\varepsilon_n = \min\left(1, \frac{cK}{d^2 n}\right),$$

где K — число автоматов, c и d — параметры, причем $0 < d < \min \Delta_i$, $\Delta_i = \mu^* - \mu_i$.

В [11] доказано, что данный алгоритм является оптимальным. Два приведенных выше алгоритма обладают существенными недостатками. Первый из которых заключается в том, что в фазе исследования номер автомата для следующей игры выбирается случайно, то есть независимо от полученных выигрышей. Второй недостаток заключается в том, что вероятность попадания в фазу исследования никак не зависит от полученных выигрышей, тогда как было бы полезно увеличивать вероятность попадания в фазу исследования в случае наличия автоматов с близкими выигрышами. От обоих недостатков свободен следующий алгоритм.

4.2.3. Алгоритм Softmax

Алгоритм описан в [10]. Его идея состоит в том, чтобы играть на автомате с вероятностью пропорциональной среднему выигрышу. Автоматы с наибольшим средним выигрышем будут играться чаще. Вычисление вероятностей основано на распределении Больцмана. Пусть после t игр имеем средние выигрыши $\widehat{\mu}_1(t), .., \widehat{\mu}_K(t)$ тогда вероятность того, что в момент времени $t+1$ будет сыгран автомат с номером i равна

$$p_i(t+1) = \frac{e^{\widehat{\mu}_i(t)/\tau}}{\sum_{j=1}^{K} e^{\widehat{\mu}_i(t)/\tau}}, i = 1, \ldots, K.$$

Параметр τ называется *температурным коэффициентом*. При $\tau \to +\infty$ алгоритм ведет себя случайным образом, при $\tau \to 0$ алгоритм становится жадным.

4.2.4. Алгоритм UCB1

UCB (Upper Confidence Bounds) — семейство алгоритмов, которое описано в [11] как простая и элегантная реализация идеи оптимизма в условиях неопределенности, предложенной в [9]. Рассмотрим самый простой в реализации и популярный алгоритм UCB1.

На каждом шаге алгоритма вычисляется величина

$$\max_{i=1,\ldots,K} \Big(\overline{x}_i + \sqrt{\frac{2\ln n}{n_i}}\Big),$$

где $\overline{x}_i$ — средний выигрыш автомата i, n_i — количество сыгранных игр на автомате i, n — количество всех сыгранных игр. Алгоритм выбирает тот автомат, для которого эта величина максимальна. Таким образом, видно, что алгоритм выбирает автомат для следующей игры основываясь не только на знании среднего выигрыша, но также использует знания о количестве сыгранных игр каждым автоматом. Именно этот факт лег в основу построения данного алгоритма. Рассмотренный алгоритм отличается от ранее описанных главным образом тем, что не имеет параметров, следовательно, нет необходимости в их подборе. Другим отличием алгоритма является его полная детерменированность, то есть отсутствие случайности. В [11] показано, что для функции сожаления R_T данного алгоритма справедлива оценка

$$R_T \leq 8 \sum_{i:\mu_i<\mu^*} \Big(\frac{\ln n}{\Delta_i}\Big) + \Big(1 + \frac{\pi^2}{3}\Big)\Big(\sum_{j=1}^{K} \Delta_j\Big),$$

где как и ранее $\Delta_i = \mu^* - \mu_i$.

4.2.5. Алгоритм погони (Pursuit)

Алгоритм описан в [10]. На каждом шаге вычисляются вероятности выбора каждого автомата для следующей игры по формуле

$$p_i(t+1) = \begin{cases} p_i(t) + \beta(1 - p_i(t)), & \text{если } i = \arg\max_{j=1,\ldots,K} \mu_j(t); \\ p_i(t) + \beta(0 - p_i(t)), & \text{в противном случае.} \end{cases}$$

На первом шаге считаем, что $p_i(0) = \frac{1}{k}$. Параметр β называется *скоростью обучения*.

4.3. Проведение эксперимента

4.3.1. Модель Бернулли многорукого бандита

Модель Бернулли задачи о многоруком бандите отличается тем, что случайные величины, характеризующие выигрыш в игре на автоматах, имеют распределение Бернулли с неизвестными вероятностями $p_1, \ldots, p_k$. Таким образом при игре на автомате с номером i игрок получает выигрыш 1 с вероятностью p_i и выигрыш 0 с вероятностью $1 - p_i$. Данный частный случай задачи о многоруком бандите отлично подходит для моделирования процесса тестирования интернет-страниц. Будем считать, что показ страницы — это игра на автомате Бернулли, а выигрыш 1 — есть достижение конверсионной цели.

4.3.2. Описание эксперимента

В качестве эксперимента, иллюстрирующего работу описанных выше алгоритмов, предложена программная реализация симуляции тестирования интернет-страниц. На вход программе подается число тестируемых страниц K, вектор вероятностей $p = [p_1, \ldots, p_k]$, где компонента p_i — есть вероятность достижения конверсионной цели на странице с номером i. Кроме того, задается число показов страниц N. Программная реализация выполнена на языке Python. В качестве результатов работы программы выводится величина конверсии, достигнутой каждым алгоритмом, и номер страницы с наибольшей конверсией. Под конверсией в данной случае понимается отношение числа достигнутых целей к общему числу показов страниц N.

Поскольку 4 из 5 рассмотренных алгоритмов зависят от параметров, которые влияют на их работу (см. пример на рис 3), то не представляется возможным подобрать параметры алгоритмов, которые гарантировали бы достижение наибольшей конверсии на всех векторах $p = [p_1, \ldots, p_k]$. С этим же связана

проблема невозможности выявления алгоритма, который бы работал на всех векторах $p = [p_1, \ldots, p_k]$ наилучшим образом.

4.3.3. Результаты и выводы

Рассмотрим несколько случаев входных данных, на которых продемонстрируем работу алгоритмов и сравним полученные результаты. Сначала рассмотрим случай тестирования двух страниц $K = 2$. Пусть $p = [0,1; 0,3]$. Параметры алгоритмов были подобраны в ходе экспериментов. Результаты работы алгоритмов представлены в таблице 9 и на рисунке 4. Все алгоритмы, за исключением UCB1, имеют элемент случайности, поэтому для них в качестве результатов берется среднее значение для 100 запусков.

Алгоритм \ Показы	1000	5000	10000	20000	50000
ε-greedy(0,01)	0,2946	0,2991	0,2983	0,2988	0,299
ε_n-greedy(0,001)	0,2934	0,2983	0,2993	0,2937	0,2996
Softmax(0,01)	0,3	0,2981	0,2991	0,3004	0,2999
Pursuit(0.05)	0,295	0,2941	0,2975	0,2999	0,3001
UCB1	0,2714	0,2893	0,2937	0,2962	0,2999

Таблица 9: Достигнутая конверсия: случай тестирования двух страниц

Из таблицы 9 и графика 4 видно, что результаты работы алгоритмов приблизительно совпадают. Следует отметить, что алгоритм Softmax в большинстве случаев достигал наибольшей конверсии среди всех алгоритмов. Кроме того, видно, что алгоритм UCB1 имеет медленный рост достигаемой конверсии, однако при большом числе показов страниц ($N > 10000$) не уступает другим алгоритмам.

В таблице 10 и на рисунке 5 приведены результаты работы алгоритмов для тестирования 5 страниц. Вектор вероятностей имеет вид $p =$

$[0,1; 0,2; 0,3; 0,4; 0,5]$.

Алгоритм \ Показы	1000	5000	10000	20000	50000
ε-greedy(0,01)	0,4645	0,483	0,4806	0,49	0,4925
ε_n-greedy(0,1)	0,4473	0,485	0,4861	0,4944	0,4975
Softmax(0,01)	0,4718	0,4794	0,472	0,4847	0,4882
UCB1	0,4214	0,4633	0,476	0,4865	0,4927
Pursuit(0,01)	0,4516	0,4844	0,486	0,4957	0,4938

Таблица 10: Достигнутая конверсия: случай тестирования пяти страниц

Из таблицы 10 видно, что для 1000 показов наибольшую конверсию достиг алгоритм Softmax, при $N > 1000$ наилучшим образом сработали Pursuit и ε_n-greedy.

Теперь рассмотрим случай $K = 10$ с вектором вероятностей

$$p = [0,1; 0,15; 0,2; 0,25; 0,3; 0,35; 0,4; 0,45; 0,5; 0,6].$$

Результаты работы алгоритмов приведены в таблице 11 и на рисунке 6. В данном случае наибольшую конверсию достигали алгоритмы ε_n-greedy и Pursuit.

Алгоритм \ Показы	1000	5000	10000	20000	50000
ε-greedy(0,05)	0,5483	0,5714	0,5746	0,5802	0,5847
ε_n-greedy(0,05)	0,5307	0,5753	0,583	0,5938	0,5903
Softmax(0,01)	0,5441	0,5563	0,579	0,5781	0,5719
UCB1	0,4464	0,5268	0,554	0,5729	0,5864
Pursuit(0,01)	0,5382	0,5817	0,5807	0,5838	0,5874

Таблица 11: Достигнутая конверсия: случай тестирования десяти страниц

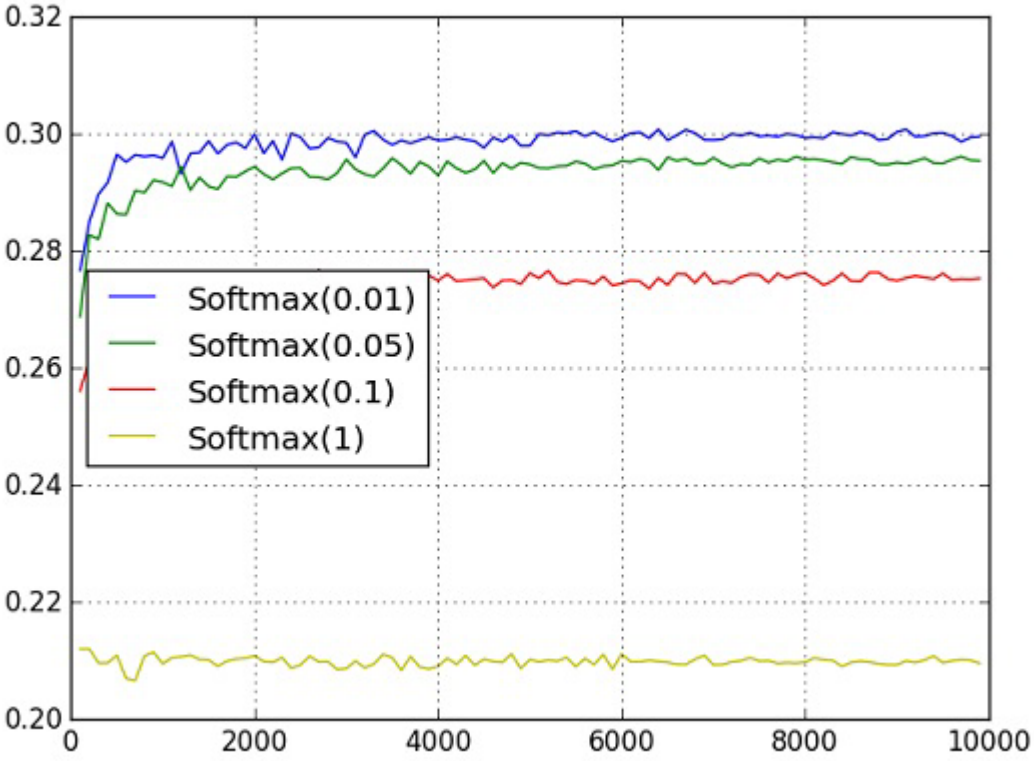

Рис. 3: Зависимость работы алгоритма Softmax от параметров

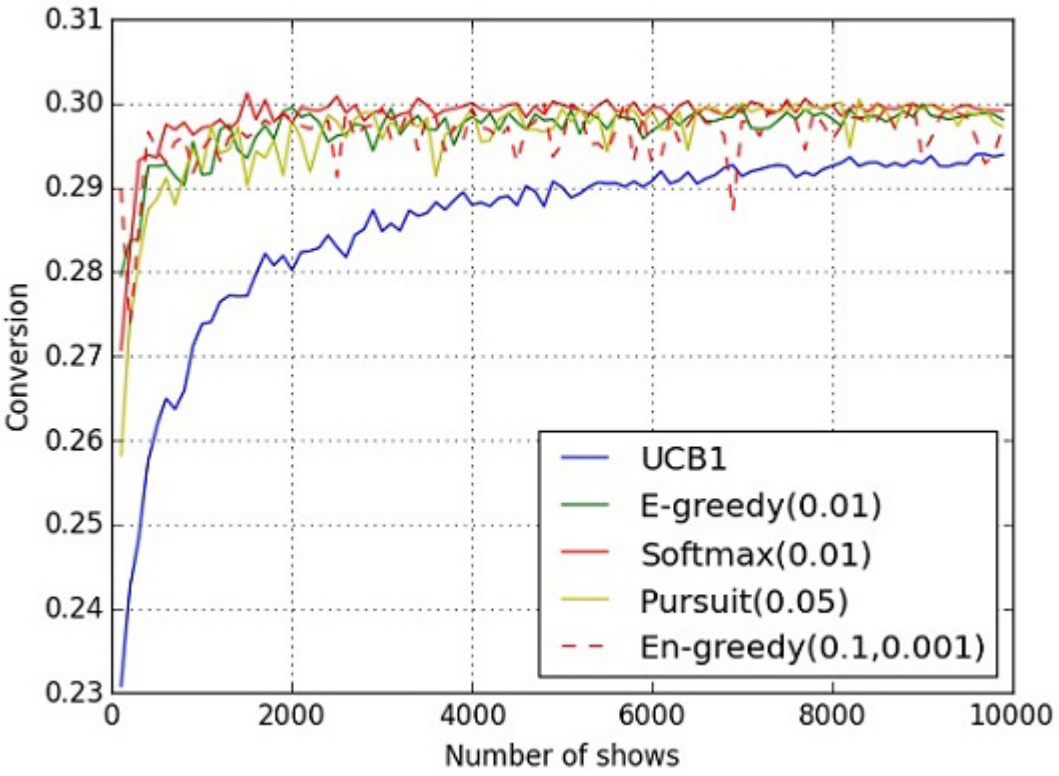

Рис. 4: Результат работы программы для двух тестируемых страниц

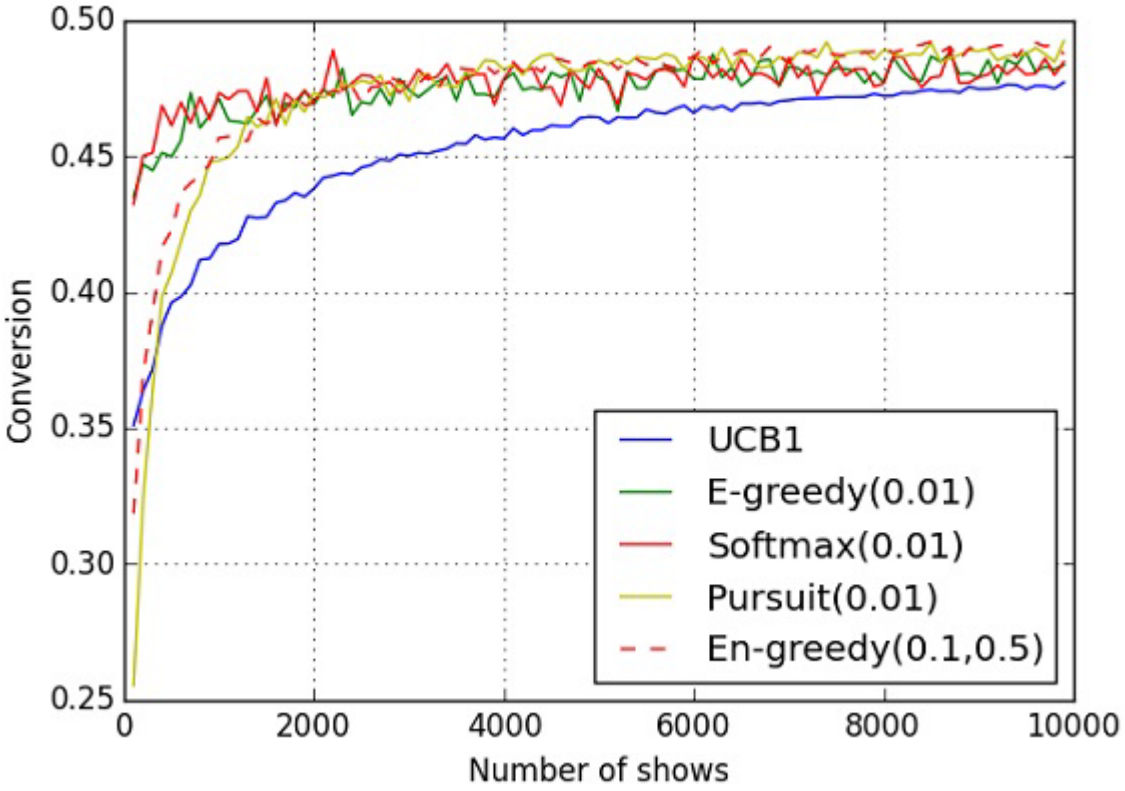

Рис. 5: Результат работы программы для 5 тестируемых страниц

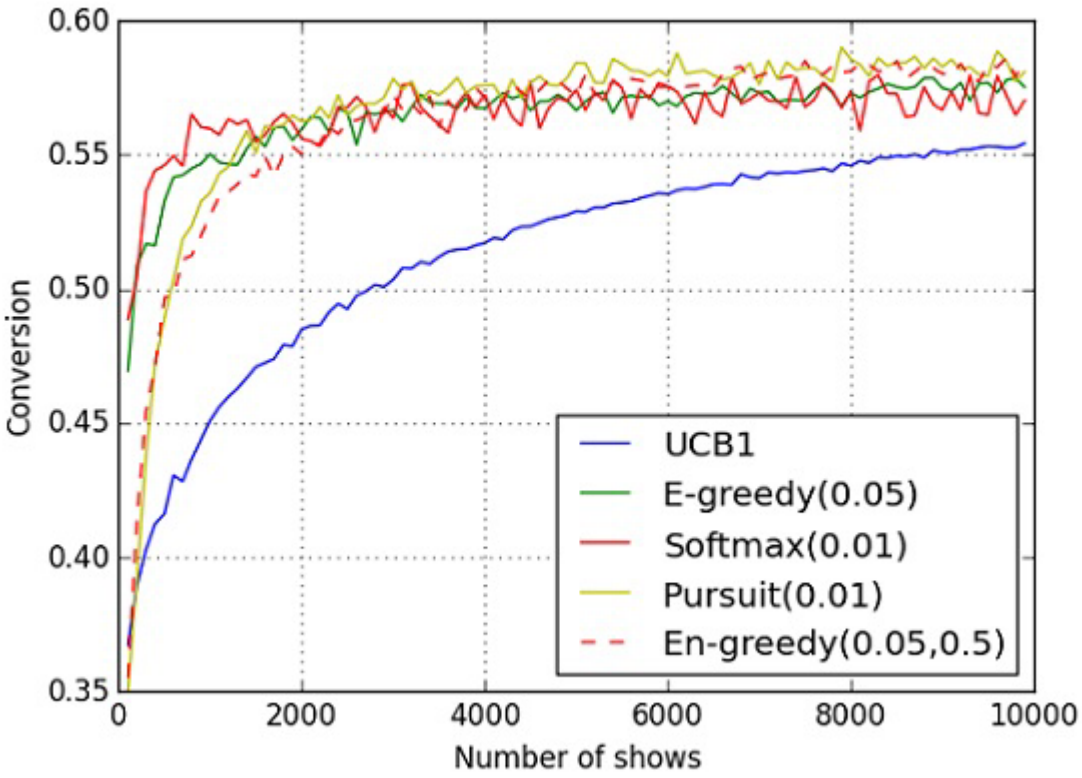

Рис. 6: Результат работы программы для 10 тестируемых страниц

Рассмотрим случай, когда конверсии страниц имеют близкие значения, что наиболее часто встречается на практике. Пусть $K = 10$ и вектор конверсионных вероятностей имеет вид

$$p = [0,4; 0,42; 0,45; 0,47; 0,5; 0,52; 0,55; 0,57; 0,58; 0; 6].$$

Результат работы алгоритмов представлен в таблице 12 и на рисунке 7.

Алгоритм \ Показы	1000	5000	10000	20000	50000
ε-greedy(0,05)	0,5658	0,5766	0,5818	0,5844	0,5886
ε_n-greedy(0,05)	0,5599	0,5784	0,5814	0,586	0,59
Softmax(0,01)	0,5659	0,5796	0,5814	0,5828	0,5857
UCB1	0,5293	0,5451	0,5544	0,5635	0,576
Pursuit(0,01)	0,5578	0,5827	0,5842	0,5839	0,5893

Таблица 12: Сравнение работы алгоритмов тестирования 10 страниц

В рассматриваемом случае важно то, с каким успехом алгоритмы находят страницу с наибольшей конверсией. В таблице 13 представлены результаты работы алгоритмов при 100 запусках. В ячейках — число запусков, в которых алгоритм обнаружил страницу с наибольшей конверсией.

Алгоритм \ Показы	1000	5000	10000	20000	50000
ε-greedy(0,05)	38	51	53	68	78
ε_n-greedy(0,05)	38	44	58	63	66
Softmax(0,01)	42	43	42	57	57
UCB1	48	82	90	100	100
Pursuit(0,01)	47	50	56	63	55

Таблица 13: Определение страницы с наибольшей конверсией

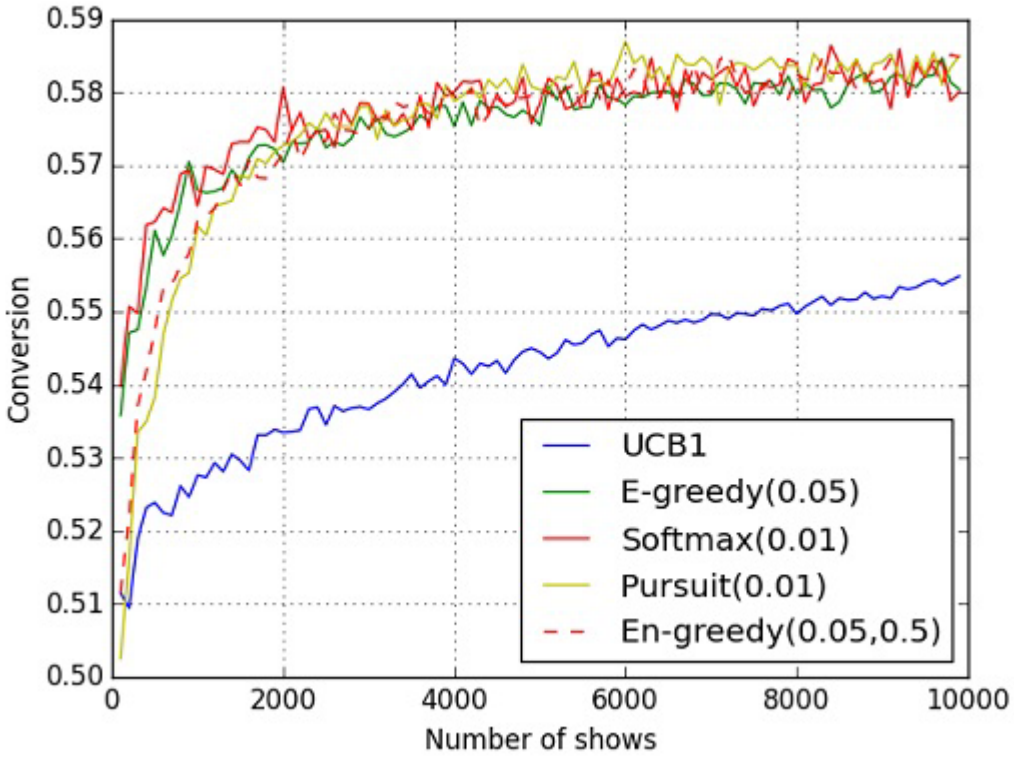

Рис. 7: Результат работы программы для 10 тестируемых страниц

В данной главе был предложен подход к тестированию интернет-страниц с использованием алгоритмов для решения задачи о многоруком бандите. В качестве проверки применимости алгоритмов предложена программная симуляция процесса тестирования. Результаты работы алгоритмов на некоторых входных данных представлены в работе.

Для случая малого числа страниц $K \leq 10$, который рассматривался в работе и который чаще всего и встречается на практике, в ходе многочисленных тестов установлено, что все алгоритмы дают приблизительно равные результаты.

Камнем преткновения к применению алгоритмов является наличие у 4 из них параметров, которые необходимо подбирать. Общего правила для их подбора, к сожалению, нет.

На практике подбор параметра следует осуществлять исходя из его смыслового значения в определенном алгоритме. Кроме того, имея некоторые статистические данные, характеризующие конверсию, или оценки конверсии для каждой страницы, можно экспериментальным путем подобрать параметры.

С точки зрения практического применения среди описанных алгоритмов выгодно отличается алгоритм UCB1, не имеющий параметров. При малом числе показов $N < 5000$ по значению достигнутой конверсии алгоритм уступает другим рассмотренным алгоритмам, однако при увеличении N демонстрирует высокий показатель конверсии, сравнимый с другими алгоритмами. Кроме того, благодаря детерменированности алгоритм гарантирует высокую конверсию от запуска к запуску, чего нельзя утверждать о других алгоритмах.

5. Заключение

В связи с бурным ростом интернет аудитории развивается и рынок интернет-рекламы. На протяжении последних 4-х лет объем рынка интернет-рекламы имеет самый быстрый рост, опережая другие виды рекламы. Основным драйвером развития рекламы в интернете является контекстная реклама, на которую приходится львиная доля рекламного бюджета. Каждый рекламодатель в рамках фиксированного рекламного бюджета стремится получить как можно больше новых клиентов. Именно задача эффективного расходования бюджета контекстной рекламы легла в основу данной работы.

Во введении работы описывается предмет исследования и вводится показатель эффективности контекстной рекламы.

Первая глава работы посвящена A/B-тестированию двух рекламных объявлений, целью которого является выявление объявления с наибольшим показателем эффективности. На основе известных статистических критериев предложен алгоритм анализа данных, полученных после проведения тестирования. Разработан код на языке C++, который полностью автоматизирует процесс анализа данных.

Во второй главе ставится задача определить, как влияет наличие сайта по запросу z_i в блоке органического поиска на показатель эффективности рекламного объявления, размещенного по запросу z_i в блоке спецразмещения. Задача была решена с использованием статистических критериев. Выяснилось, что наличие сайта в блоке органического поиска положительно сказывается на показатель эффективности контекстной рекламы, следовательно, рекламодателю выгодно размещать объявления по запросам, по которым сайт высоко ранжируется в поиске.

Третья глава работы посвящена тестированию интернет-страниц с це-

лью увеличения конверсии. Конверсия страницы отражает то, с каким успехом посетитель интернет-страницы становится клиентом. Данная глава лишь косвенно связана с контекстной рекламой, однако важна с точки зрения её эффективности, поскольку конечной целью любой рекламной кампании являются новые клиенты. В основе предложенного подхода к тестированию лежит задача о многоруком бандите, которая является частным случаем дилеммы исследования-использования. Такой подход к тестированию позволяет не только выявить страницу с наибольшей конверсией, но и достичь высокую конверсию во время проведения теста. Были рассмотрены 5 алгоритмов решения данной задачи. В качестве иллюстрации работы алгоритмов и их применимости к тестирования интернет-страниц предложена программная симуляция тестирования, реализация которой выполнена на языке Python. В ходе экспериментов подтверждена применимость данных алгоритмов для тестирования интернет-страниц.

Список литературы

[1] Холодкова К.С. Анализ рынка электронной коммерции в России // Современные научные исследования и инновации. 2013. № 10 [Электронный ресурс]. URL:http://web.snauka.ru/issues/2013/10/26760 (дата обращения: 29.03.2015).

[2] Обзор рынка интернет-рекламы в России // URL: http://json.tv/ict_telecom_analytics_view/obzor-rynka-internet-reklamy-v-rossii-itogi-2014-g-20150226053941 (дата обращения: 23.04.2015).

[3] Правила показа объявлений в системе Яндекс Директ // URL: http://legal.yandex.ru/direct_display_rules/ (дата обращения: 29.03.2015).

[4] Жадный алгоритм в A/B-тестировании // URL: http://habrahabr.ru/post/144977/ (дата обращения: 05.04.2015).

[5] Статистическая значимость и ошибки в A/B тестах // URL: http://www.cossa.ru/articles/152/23876/ (дата обращения: 02.03.2015).

[6] Буре В. М., Парилина Е. М. Теория вероятностей и математическая статистика. М.: Лань, 2013. 416 с.

[7] Кобзарь А. И. Прикладная математическая статистика. Для инженеров и научных работников. М.: Физматлит, 2006. 816 с.

[8] Sutton R. S., Barto A. G. Introduction to reinforcement learning. MIT Press, 1998.

[9] Lai T. L., Robbins H. Asymptotically efficient adaptive allocation rules // Advances in applied mathematics. 1985. No 6. P. 4–22.

[10] Kuleshov V., Precup D. Algorithms for the multi-armed bandit problem // Journal of Machine Learning Research. 2000. No 1. P. 1–48.

[11] Auer P., Cesa-Bianchi N., Fischer P. Finite-time Analysis of the Multiarmed Bandit Problem // Machine Learning. 2002. Vol. 47, No 2-3. P. 235–256.

[12] Борисова Т. С. Решение задачи исследования-использования для показа баннеров.

6. Приложения

6.1. Приложение 1. Программный код для главы 1

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <vector>
#include <algorithm>
#include <math.h>
#define M_PI 3.14159265358979323846
using namespace std;

double sred (vector <double> x);
double dispv (vector <double> x, double sredx);
double student (vector <double> x, vector <double> y, double sredx, double sredy,
    double dispx, double dispy);
double pirson (vector <double> x, double sredx, double dispx);
double disp (vector <double> x, double sredx);
double plotnost (double x, double sredx, double dispx);
double normrasp (double x, double sredx, double otklx);
double hi2obr(double uroven);
double student_obr(double uroven, int step_svob);
double fisher_obr(double uroven, int step_svob1, int step_svob2);
double mann_uitni(vector <double> x, vector <double> y);
double kvantil_norm_stand(double uroven);
double kohran_koks(vector <double> x, vector <double> y);
double kohran_koks_kvantil(vector <double> x, vector <double> y, double uroven);

double stepen_svobodi = 0;
int flag_fisher, flag_norm;

int main() {
  system("cls");
  double sreda, sredb, dispva, dispvb, dispra, disprb, phi, temp, alpha, fisher;
  int n,m, i = 0, select_test = 0;
```

```
fstream G, F;
vector <double> a, b;

F.open("A.txt",ios::in); //открываем файлы для чтения
G.open("B.txt",ios::in);

while (!F.eof()) {
  F >> temp;
  a.push_back(temp);
}

while (!G.eof()) {
  G >> temp;
  b.push_back(temp);
}

G.close();
F.close();

sreda = sred(a); //средние
sredb = sred(b);
dispva = dispv(a, sreda); //испр выб дисп
dispvb = dispv(b, sredb);
dispra = disp(a, sreda); //выборочные дисперсии
disprb = disp(b, sredb);

cout << "sred A = " << sreda << endl;
cout << "sred B = " << sredb << endl;
cout << "disp A = " << dispva << endl;
cout << "disp B = " << dispvb << endl;
cout << "dispr A = " << dispra << endl;
cout << "dispr B = " << disprb << endl;

cout << "Select alpha (0.1, 0.05, 0.01 or 0.001) = ";
cin >> alpha;

if (sreda>sredb)
  select_test = 1;
```

```
else
  select_test = 2;
//------------------ Проверка на нормальность ---------------
flag_norm = 1;
double chia = pirson(a, sreda, dispra);
cout << "Pirson A = " << chia << endl;
double granica_pirson_a = hi2obr(alpha);

if (chia > granica_pirson_a) {
  cout << "A have not normal distribution" << endl;
  flag_norm = 0;
}
else
  cout << "A have normal distribution" << endl;

double chib = pirson(b,sredb,disprb);
cout << "Pirson B = " << chib << endl;
double granica_pirson_b = hi2obr(alpha);

if (chib > granica_pirson_b) {
  cout << "B have not normal distribution" << endl;
  flag_norm = 0;
}
else
  cout << "B have normal distribution" << endl;

//--------------------------------- Критерий Манна-Уитни ------------
if (flag_norm == 0) {
  double mann = mann_uitni(a,b);
  double kvant_norm_stand = kvantil_norm_stand(1-alpha);
  if (select_test == 1) {
    if (mann > kvant_norm_stand)
      cout << "F(z) > G(z) yes (Mann-Uitni)" << endl;
    else
      cout << "F(z) > G(z) no (Mann-Uitni)";
  }
  else {
    if (mann < kvant_norm_stand)
```

```
      cout << "F(z) < G(z) yes (Mann-Uitni)" << endl;
    else
      cout << "F(z) < G(z) no (Mann-Uitni)";
  }
}
else {
  flag_fisher = 1;
//-------------- Проверка равенства дисперсий -------------------

  if (dispva > dispvb)
    fisher = dispva/dispvb;
  else
    fisher = dispvb/dispva;

  cout << "Fisher = " << fisher << endl;

  double fish_obr = fisher_obr(alpha, a.size()-1, b.size() - 1);

  if (((fisher >=0) & (fisher < 1.0/fish_obr)) || (fisher > fish_obr)) {
    cout << "variances are not equal" << endl;
    flag_fisher = 0;
  }
  else
    cout << "variances are equal" << endl;
}

if (flag_fisher == 1) {
//если дисперсии равны, используем критерий Стьюдента

  phi = student(a, b, sreda, sredb, dispva, dispvb);
  cout << "Student = " << phi << endl;

  double kvant_stud = student_obr(alpha, a.size()+b.size() - 2);
  if (select_test == 1) {
    if (phi > kvant_stud)
      cout << "EA > EB yes (Student's test)" << endl;
    else
      cout << "EA > EB no (Student's test)" << endl;
```

```
    }
    else {
      if (phi < -kvant_stud)
        cout << "EA < EB yes (Student's test)" << endl;
      else
        cout << "EA < EB no (Student's test)" << endl;
    }
  }
  else {
    //-------------------------- критерий Кохрана-Кокса ------------
    double koh_koks = kohran_koks(a, b);
    cout << "kohran = " << koh_koks << endl;

    double kvant_koh = kohran_koks_kvantil(a, b, alpha);

    if (select_test == 1) {
      if (koh_koks > kvant_koh)
        cout << "EX > EY yes (Kohran-Koks)" << endl;
      else
        cout << "EX > EY no (Kohran-Koks)" << endl;
    }
    else {
      if (koh_koks < -kvant_koh)
        cout << "EX < EY yes (Kohran-Koks)" << endl;
      else
      cout << "EX < EY no (Kohran-Koks)" << endl;
    }
  }
system("pause");
return 0;
}

//функция среднего
double sred (vector <double> x) {
  double summ = 0;
  for (int j = 0; j < x.size(); j++)
    summ = summ + x[j];
  return summ/x.size();
```

```
  }

//исправленная выборочная дисперсия
double dispv (vector <double> x, double sredx) {
  double summ = 0;
  for (int j = 0; j<x.size(); j++)
    summ = summ + (x[j] - sredx) * (x[j] - sredx);
  return summ/(x.size() - 1);
}

//выборочная дисперсия
double disp (vector <double> x, double sredx) {
  double summ = 0;
  for (int j = 0; j<x.size(); j++)
    summ = summ + (x[j] - sredx) * (x[j] - sredx);
  return summ/(x.size());
}

//Критерий Стьюдента
double student (vector <double> x, vector <double> y, double sredx, double sredy,
    double dispx, double dispy) {
  double s = sqrt((x.size()*dispx + y.size()*dispy)/(x.size()+y.size()-2));
  double phi = (sredx - sredy) / (s * sqrt(double((1/double(x.size())) + (1/
     double(y.size())))));
  return phi;
}

// Критерий Пирсона
double pirson (vector <double> x, double sredx, double dispx) {
  stepen_svobodi = 0;

  //поиск максимума и минимума
  double minx, maxx;
  minx = x[0];
  maxx = x[0];
  for (int j = 1; j<x.size(); j++) {
    if  (x[j] < minx)
      minx = x[j];
```

```
}

for (int j = 1; j<x.size();j++) {
  if (x[j] > maxx)
    maxx = x[j];
}

//формула Стерджесса - оптимальное число интервалов.
float r, k = modff((1+3.32*log10(double(x.size()))),&r);
double step = (maxx - minx) / r; //длина шага

//определение частотности
vector <double> freqn(r);
double k1 = minx;
int k2 = 0;
for (int j = 0; j < x.size(); j++) {
      if ((x[j] >= k1) & (x[j] <= (k1 + step)))
        k2 = k2 + 1;
    }
freqn[0] = k2;
k1 = k1 + step;

int i = 1;
while (i < r) {
  k2 = 0;
  for (int j = 0; j < x.size(); j++) {
      if ((x[j] > k1) & (x[j] <= (k1 + step)))
        k2 = k2 + 1;
    }
  freqn[i] = k2;
  i = i + 1;
  k1 = minx + step*i;
}

//объединяем интервалы с частотой менее 5
int fl=0, k3=0, min = 5;
vector <double> mnozh(freqn.size()), p(freqn.size()) ;
```

```
for (int j = 0; j < mnozh.size(); j++)
  mnozh[j] = 1;

int count = 1;
i=0;
for(; (i<freqn.size())&&(fl==0); i++)
  if  (freqn[i]>=min){
    p[k3] = freqn[i];
    k3++;
  } else {
    fl = 1;
    count = 1;
    for (int j=i+1; j<freqn.size(); j++){
      freqn[i] = freqn[i] + freqn[j];
      count = count + 1; //считаем количество прибавлений

      if  (freqn[i]>=min){
        p[k3] = freqn[i];
        mnozh[k3] = count;
        i=j;
        fl=0;
        k3++;
        count = 1;
        break;
      }
    }
  }

if (fl == 1) {
  p[k3-1] =p[k3-1]+ freqn[i-1];
  mnozh[k3-1] = mnozh[k3-1] + count;
  }

stepen_svobodi = k3 - 3;

//построение массива с концами интервалов.
vector <double> interval(k3 + 1);
interval[0] = minx;
```

```
  for (int j = 1; j < k3+1; j++)
    interval[j] = interval[j-1] + mnozh[j-1]*step;

    //вычисление вероятностей
  vector <double> pi(k3);
  pi[0] = normrasp(interval[1], sredx, sqrt(dispx));
  for (int j = 1; j < k3; j++)
    pi[j] = normrasp(interval[j+1],sredx,sqrt(dispx))-normrasp(interval[j],sredx,
        sqrt(dispx));

    //вычисление статистики Пирсона
  double chi = 0;
  for (int j = 0; j < k3; j++)
    chi = chi + (p[j]-x.size()*pi[j])*(p[j]-x.size()*pi[j])/(x.size()*pi[j]);

  return chi;
}

//функция нормального распределения
double normrasp (double x, double sredx, double otklx) {
  double db_norm[401];
  double x1, x1_new, temp=0;
  fstream F;
  F.open("db_normal.txt",ios::in);
  x1 = (x-sredx)/otklx;
  int otr=0;

  if (x1<0){
    x1=fabs(x1);
    otr=1;
  }

  x1=modf(x1*1000,&x1_new);

  for (int j=0; j<x1_new+1; j++){
    F >> temp;
  }
  F.close();
```

```
  if (otr==1)
    return 1 - temp;
  else
    return temp;
}

//критический интервал для хи2
double hi2obr(double uroven) {
  fstream F;
  F.open("hi2obr.txt", ios::in);
  double temp;
  uroven = 1 - uroven;

  for (int j = 0; j < 4*(stepen_svobodi-1); j++ )
    F>>temp;

  if (uroven == 0.9)
    F >> temp;
  if (uroven == 0.95)
    F >> temp >> temp;
  if (uroven == 0.99)
    F >> temp >> temp >> temp;
  if (uroven == 0.999)
    F >> temp >> temp >> temp >> temp;

  F.close();
  return temp;
}

//критический интервал Стьюдента
double student_obr(double uroven, int step_svob) {
  fstream F;
  double temp, kvant_ur = 1 - uroven;
  F.open("student_obr.txt",ios::in);

  if (step_svob <= 60) {
    for (int j = 0; j < 6*(step_svob - 1); j++)
```

```
      F >> temp;
  }
  else {
    for (int j = 0; j < 6*60; j++)
      F >> temp;
  }
  if (kvant_ur == 0.9)
    F >> temp;
  if (kvant_ur == 0.95)
    F >> temp >> temp;
  if (kvant_ur == 0.975)
    F >> temp >> temp >> temp;
  if (kvant_ur == 0.99)
    F >> temp >> temp >> temp >> temp;
  if (kvant_ur == 0.995)
    F >> temp >> temp >> temp >> temp >> temp;
  if (kvant_ur == 0.999)
    F >> temp >> temp >> temp >> temp >> temp >> temp;

  F.close();
  return temp;
}

//критический интервал для Фишера
double fisher_obr(double uroven, int step_svob1, int step_svob2) {
  fstream F;
  double temp, uroven_kvantil;
  uroven_kvantil = 1-uroven/2;

  if (uroven_kvantil == 0.95)
    F.open("fisher_obr_95.txt",ios::in);
  if (uroven_kvantil == 0.975)
    F.open("fisher_obr_975.txt",ios::in);
  if (uroven_kvantil == 0.995)
    F.open("fisher_obr_995.txt",ios::in);
  if (uroven_kvantil == 0.9995)
    F.open("fisher_obr_9995.txt",ios::in);
```

```
  if (step_svob1 <= 50) {
    for (int j = 0; j < (step_svob2-1)*51 + step_svob1; j++)
      F >> temp;
    F.close();
  }
  else {
    for (int j = 0; j < step_svob2*51; j++)
      F >> temp;
    F.close();
  }
  return temp;
}

//Критерий Манна-Уитни
double mann_uitni(vector <double> x, vector <double> y) {
  double count = 0;
  for (int i = 0; i < x.size(); i++) {
    for (int j = 0; j < y.size(); j++)
      if (x[i] < x[j])
        count = count + 1;
  }

  cout << "mann = " << count << endl;
  //аппроксимация нормальным стандартным распределением
  double u = (count - x.size() * y.size()/2) / sqrt(double(x.size() * y.size() *
     (x.size() + y.size() + 1) / 12));
  return u;
}

//Критический интервал для Манна-Уитни
double kvantil_norm_stand(double uroven) {
  fstream F;
  double temp;
  F.open("norm_stand_obr.txt",ios::in);
  int count = 0, count_min = 0;
  double min = 100;

  while (!F.eof()) {
```

```
    F >> temp;
    count++;
    if (abs(uroven - temp) < min) {
      min = abs(uroven - temp);
      count_min = count;
    }
  }
  cout << "count_min" << count_min << endl;
  temp = double(count_min)/100.0;
  F.close();

  return temp;
}

//критерий Кохрана-Кокса
double kohran_koks(vector <double> x, vector <double> y) {
  double sredx = sred(x);
  double sredy = sred(y);
  double sx = dispv(x, sredx);
  double sy = dispv(y, sredy);
  double s = sx*sx/x.size() + sy*sy/y.size();
  double t = 1/sqrt(s)*(sredx - sredy);
  return t;
}

//квантиль критерия Кохрана-Кокса
double kohran_koks_kvantil(vector <double> x, vector <double> y, double uroven) {
  double sredx = sred(x);
  double sredy = sred(y);
  double sx = dispv(x,sredx);
  double sy = dispv(y,sredy);
  double vx = sx*sx/x.size();
  double vy = sy*sy/y.size();
  double t = (vx*student_obr(uroven,x.size() - 1) + vy*student_obr(uroven, y.size
      () - 1))/(vx + vy);
  return t;

}
```

6.2. Приложение 2. Данные после проведения теста

№	ctr 1-го объявления	ctr 2-го объявления
1	0,061	0,1045
2	0,0317	0,103
3	0,0773	0,0735
4	0,1083	0,0763
5	0,106	0,0756
6	0,122	0,1025
7	0,0453	0,1061
8	0,063	0,0491
9	0,1029	0,0862
10	0,0374	0,1009
11	0,0493	0,0795
12	0,0193	0,1014
13	0,0146	0,0595
14	0,0807	0,1683
15	0,0468	0,0426
16	0,0665	0,0991
17	0,053	0,1573
18	0,0579	0,111
19	0,074	0,1086
20	0,059	0,0727
21	0,0602	0,0801
22	0,0589	0,1054
23	0,1103	0,1424
24	0,0674	0,058
25	0,0644	0,1218
26	0,0546	0,0756
27	0,1292	0,0972
28	0,096	0,1122
29	0,1413	0,1211
30	0,0504	0,0729

№	ctr 1-го объявления	ctr 2-го объявления
31	0,1198	0,1347
32	0,0216	0,0731
33	0,0862	0,0702
34	0,0971	0,1436
35	0,1276	0,0811
36	0,0675	0,1125
37	0,0843	0,113
38	0,0903	0,081
39	0,0586	0,1011
40	0,0927	0,1008
41	0,0867	0,075
42	0,1046	0,1267
43	0,0444	0,0781
44	0,0591	0,0938
45	0,069	0,0986
46	0,0708	0,0966
47	0,0603	0,0982
48	0,1358	0,12
49	0,1077	0,1523
50	0,0479	0,0512

Таблица 14: Данные после проведения теста

6.3. Приложение 3. Данные для решения задачи главы 2

№	$ctr(pos = 1, 2, 3)$	$\overline{ctr}$
1	21,1	5,05
2	22,82	10,39
3	11,63	8,81
4	40	9
5	6,06	4,02
6	18,42	10,39
7	9,33	13,61
8	5	16,3
9	30,53	14,01
10	29,45	5,69
11	19,48	10,7
12	31,29	41,72
13	38,93	6,83
14	22,83	6,75
15	46,15	12,95
16	0,45	9,42
17	3,1	8,13
18	0	4,63
19	18,97	7,42
20	9,19	7,42
21	14	20
22	17,65	12,5
23	12,5	12,25
24	28,68	12,26
25	17,99	13,58
26	21,57	7
27	24,1	12,4
28	16,46	16,9
29	8,33	14,65
30	19,93	3,54

№	$ctr(pos = 1, 2, 3)$	$\overline{ctr}$
31	26,32	18,56
32	8,8	10
33	34,29	10,82
34	35,71	7,75
35	13,33	11,5
36	0	9,17
37	11,54	6,87
38	12,5	8,4
39	16,67	6,24
40	29,79	6,15
41	6,25	8,45
42	9,09	2,78
43	0	21,72
44	9,33	13,61
45	5	18,76
46	7,62	7,78
47	10,71	8,12
48	0,72	0,45
49	7,49	11,75
50	0,61	4,46
51	17,39	9,56
52	2,06	8,56
53	22,58	3,08
54	3,7	3,08
55	21,21	3,08
56	12,5	11,43
57	8,23	8,73
58	2,78	11,42
59	0,43	0,68
60	0,84	0,79
61	0,45	0,48
62	8,07	18,78

№	$ctr(pos = 1, 2, 3)$	$\overline{ctr}$
63	15,21	0,87
64	21,13	12,425
65	10,66	6,05
66	13,64	6,39
67	10	5,55
68	23,18	8,66
69	31,03	8,54
70	10,61	14,23
71	6,25	3,51
72	43,9	2,63
73	13,04	6,64
74	10,58	9,62
75	13,16	10,27
76	8,06	5,88
77	25	14,1
78	22,22	9,28
79	27,47	11,42
80	14,54	6,82
81	23,53	7,63
82	16	8,24
83	4,27	8,75
84	12,58	2,74
85	3,12	12,17

Таблица 15: Данные для решения задачи главы 2

6.4. Приложение 4. Программный код для главы 3

```
import random
import math
import numpy
import matplotlib.pyplot as plt

n = 10
n_show = 1000
n_start = 100
p = [0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.6]
page_show = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
page_target = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
total_show_page = 0
total_target = 0
best_page_index = 0
i = 0;

print('---------a/b - testing!--------- \n')

#--------------------------- Show page ----------------------------
def show_page(number_page):
    global total_target
    global total_show_page
    x = random.random()
    total_show_page = total_show_page + 1
    page_show[number_page] = page_show[number_page] + 1
    if x <= p[number_page]:
        total_target = total_target + 1
        page_target[number_page] = page_target[number_page] + 1
    return 0

#----------------------------  UCB1 -------------------------------
def UCB1():
    best_page_index = 0
    max = page_target[0]/page_show[0] + math.sqrt((2*math.log(total_show_page))/(
       page_show[0]))
    i = 1
```

```
    while i < n:
        if page_target[i]/page_show[i] + math.sqrt((2*math.log(total_show_page))
            /(page_show[i])) > max:
            max = page_target[i]/page_show[i] + math.sqrt((2*math.log(
                total_show_page))/(page_show[i]))
            best_page_index = i
        i = i + 1
    return best_page_index

#---------------------------- e-greedy -------------------------------
def e_greedy(epsilon):
    x = random.random()
    best_page_index = 0
  max = page_target[0]/page_show[0]
    i = 1
    while i < n:
        if page_target[i]/page_show[i] > max:
            max = page_target[i]/page_show[i]
            best_page_index = i
        i = i + 1

    if x < 1 - epsilon:
        return best_page_index
    else:
        best_page_index = random.randint(0,n-1)
        return best_page_index

#---------------------------- Softmax --------------------------------
def softmax(tau):
    best_page_index = 0
    p1 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
  summ = 0
    j = 0
    while j < n:
        summ = summ + math.exp((page_target[j]/page_show[j])/tau)
        j = j + 1
  i = 0
    while i < n:
```

```
        p1[i] = math.exp((page_target[i]/page_show[i])/tau) / summ
        i = i + 1
    x = random.random()
    temp_sum=p1[0]
    i=1
    best_page=0
    while i<=n:
        if x<temp_sum:
           best_page = i-1
           break
        else:
            temp_sum = temp_sum+p1[i]
            i = i + 1
    return best_page

#--------------------------------- Pursuit --------------------------------
def pursuit(beta):
    maxmu = page_target[0]/page_show[0]
    max_number = 0

    j = 1
    while j < n:
        if page_target[j]/page_show[j] > maxmu:
            maxmu = page_target[j]/page_show[j]
            max_number = j
        j = j + 1

    j = 0
    while j < n:
        if j == max_number:
            p1[j] = p0[j] + beta * (1 - p0[j])
        else:
            p1[j] = p0[j] + beta * (0 - p0[j])
        j = j + 1

    x = random.random()
    temp_sum=p1[0]
    i=1
```

```
    best_page=0
    while i<=n:
        if x<temp_sum:
           best_page = i-1
           break
        else:
            temp_sum = temp_sum+p1[i]
            i = i + 1

    j = 0
    while j < n:
        p0[j] = p1[j]
        j = j + 1

    return best_page

#--------------------------- en-greedy ---------------------------------
def en_greedy(c,d,j):
    x = random.random()
    best_page_index = 0
  temp = c*n/(d*d*(j+1))
    if temp < 1:
        epsilon = temp
    else:
        epsilon = 1

    max = page_target[0]/page_show[0]
    i = 1
    while i < n:
        if page_target[i]/page_show[i] > max:
            max = page_target[i]/page_show[i]
            best_page_index = i
        i = i + 1

    if x < 1 - epsilon:
        return best_page_index
    else:
        best_page_index = random.randint(0,n-1)
```

```
        return best_page_index

#---------------------- prepare for testing -----------------------------
def clean():
    global total_show_page
    global total_target
    total_show_page = 0
    total_target = 0
  j = 0
    while j < n:
        page_target[j] = 1
        page_show[j] = 1
        j = j + 1

    return 0

#------------------------------- TESTING -------------------------------
#UCB1
UCB = []
k = 0
while k < n_start:
    clean()
    i = 0
    while i < n:
        show_page(i)
        i = i + 1

  j = 0;
    while j < n_show-n:
        best_index = UCB1()
        show_page(best_index)
        j = j + 1

    UCB.append(total_target)

  k = k + 1

i = 0
```

```
summ = 0
while i < n_start:
    summ = summ + UCB[i]
    i = i + 1
print('sred UCB = ', summ/n_start/n_show)

#e-greedy
greedy = []
k = 0
while k < n_start:
    clean()
    j = 0
    while j < n_show:
        j = j + 1
        best_index = e_greedy(0.05)
        show_page(best_index)

    greedy.append(total_target)
    k = k + 1
i = 0
summ = 0
while i < n_start:
    summ = summ + greedy[i]
    i = i + 1
print('sred greedy = ', summ/n_start/n_show)

#Softmax
softmax_massiv = []
k = 0
while k < n_start:
    clean()
    j = 0
    while j < n_show:
        best_index = softmax(0.01)
        show_page(best_index)
        j = j + 1

  softmax_massiv.append(total_target)
```

```
    k = k + 1

i = 0
summ = 0
while i < n_start:
    summ = summ + softmax_massiv[i]
    i = i + 1

print('softmax = ', summ/n_start/n_show)

#Pursuit
pursuit_massiv = []
k = 0
while k < n_start:
    clean()
    p0 = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
    p1 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    j = 0
    while j < n_show:
        best_index = pursuit(0.01)
        show_page(best_index)
        j = j + 1

  pursuit_massiv.append(total_target)
    k = k + 1

i = 0
summ = 0
while i < n_start:
    summ = summ + pursuit_massiv[i]
    i = i + 1

print('pursuit = ', summ/n_start/n_show)

#en-greedy
en_greedy_mass = []
k = 0
while k < n_start:
```

```
    j = 0
    clean()

    while j < n_show:
        best_index = en_greedy(0.01, 0.1, j)
        show_page(best_index)
        j = j + 1

    en_greedy_mass.append(total_target)
    k = k + 1

i = 0
summ = 0
while i < n_start:
    summ = summ + en_greedy_mass[i]
    i = i + 1
print('en_greedy = ', summ/n_start/n_show)
```