Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ***ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ***

КАФЕДРА ***СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)***

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

Программный модуль формирования рекомендаций по выдаче банковского кредита

Студентка ВИУ5-81 (Группа)	_____ (Подпись, дата)	**М. О. Таран** (И.О.Фамилия)
Руководитель ВКР	_____ (Подпись, дата)	**Ю.Е. Гапанюк** (И.О.Фамилия)
Консультант	_____ (Подпись, дата)	_____ (И.О.Фамилия)
Консультант	_____ (Подпись, дата)	_____ (И.О.Фамилия)
Нормоконтролер	_____ (Подпись, дата)	_____ (И.О.Фамилия)

2017 г.

Реферат

Расчетно-пояснительная записка 88с., 45 рис., 26 табл., 20 источников, 5 приложений.

МАШИННОЕ ОБУЧЕНИЕ, АЛГОРИТМЫ, ANACODA, КРЕДИТЫ, ПОДБОР, РЕАЛИЗАЦИЯ, МОДУЛЬ, РЕКОМЕНДАЦИИ.

Объектом квалификационной работы является программный модуль формирования рекомендаций по выдаче банковского кредита для автоматизированной банковской системы, который позволит специалисту банка принимать решение о дальнейших действиях с каждой вновь поступившей кредитной заявкой.

Цель работы – создание модуля рекомендаций на основе алгоритмов машинного обучения для решения задачи бинарной классификации для предоставленного набора данных с кредитными заявками из магазинов электроник.

Конструкторская часть пояснительной записки содержит общетехническое обоснование разработки, включающее постановку задачи, описание предметной области, анализ данных, разработку инфологической и даталогической моделей, описание и разработку программного модуля.

В технологической части пояснительной записки рассмотрены вопросы взаимодействия с системой и разработка интерфейса пользователей. Дано описание экранных форм пользовательской части.

Научно-исследовательская часть содержит постановку задачи машинного обучения для бинарной классификации, описание метрики AUC ROC, принципы тренировки и валидации моделей, типичные ошибки и переобучение, описание алгоритмов: логистическая регрессия, K-ближайших соседей (k-nearest neighbors, KNN), случайный лес (RandomForest), многослойная нейронная сеть (MLP), сверточные нейронные сети (CNN), а также определение алгоритма, который лучше всего справляется с поставленной задачей.

В экономическом обосновании разработки приведена смета затрат на разработку данной системы.

Разработан эргономический сертификат, содержащий общие и частные эргономические характеристики, оценки яркостно-контрастных, пространственных, временных, информационных характеристик.

Разработка системы проводилась на основании задания на квалификационную работу бакалавра, подписанного руководителем квалификационной работы бакалавра и утвержденного заведующим кафедрой ИУ5 МГТУ им. Н.Э. Баумана и документа "Техническое задание", утвержденного руководителем квалификационной работы бакалавра.

Содержание

Введение

Большинство крупных банков уже в той или иной мере используют машинное обучение и анализ данных в своей деятельности. Помимо задачи скоринга, могут быть и другие приложения, например, определение индивидуальной кредитной ставки, оценка качества залога, оптимизация количества операционистов в зале в зависимости от часа и дня приема и т.д.

Решение подобных задач может существенно повысить рентабельность бизнеса и сократить издержки из-за человеческого фактора.

Организация всегда стремится к максимизации процентной ставки, а заемщик желает получить вообще беспроцентный займ. Рыночная и конкурентная среда позволяет заемщику не только брать кредиты, но и найти наиболее приемлемый банк и тариф.

Это порождает еще один вид задач. Определить по анкетным данным возьмет ли заемщик кредит в конкретном банке или пойдет в другой. Это типичная задача машинного обучения, а именно, задача бинарной классификации. Когда на вход подаются некоторые значения из анкеты заемщика, в результате должны получить некоторый класс заемщика (возьмет, не возьмет).

Для решения подобных задач необходимо минимум две вещи. Во-первых, некоторый математический аппарат, применив который получим необходимый результат. Во-вторых, данные, на которых необходимо обучить модель.

В современных курсах по машинному обучению большую роль отводят именно математической стороне, а работу с данными просто опускают, принимая как само собой разумеющийся факт.

Однако, в реальных задачах именно работа с данными занимает около 80% рабочего времени. А качество входных данных непосредственно влияет на качество построенной модели. Поэтому в данной квалификационной работе

будут рассмотрены основные этапы как машинного обучения, так анализа данных.

1. Конструкторская часть

1.1. Постановка задачи

Необходимо разработать модуль рекомендаций для сотрудника банка, который будет прогнозировать взятие кредита в целевом банке на основе предоставленного набора данных. Данные взяты с соревнования, организованного банком Тинькофф (конкурс проводился с 10 января 2017 года по 27 февраля 2017 года).

В рамках квалификационной работы были поставлены следующие задачи:

- Анализ предметной области.
- Анализ предоставленного набора данных.
- Постановка задачи машинного обучения и определение алгоритма, который лучше всего справляется с поставленной задачей.
- Настройка рабочего окружения для развертывания Django и библиотек машинного обучения.
- Разработка модуля рекомендаций.
- Разработка программного модуля.
- Экономический расчет.

В настоящий момент существует большая потребность в применении машинного обучения для оптимизации работы с кредитными заявками. Такие системы разрабатываются индивидуально под каждый банк, поэтому полностью универсального решения на данный момент не существует.

1.2. Описание предметной области

Первые формы потребительского кредитования возникли еще во времена земледелия. Тогда в заем брали семена или другие необходимые вещи. Отдавать же приходилось уже, например, не мешок семян, а два. Целью такого займа была необходимость прокормить семью и обеспечить её выживание.

Со временем цель кредитования изменилась. Сегодня кредиты берут не только на необходимые вещи, но и на предметы роскоши и статуса. Хорошим примером являются кредиты на автомобиль и электронику.

Кредитование населения экономически выгодно как для государства, так и для самого кредитного учреждения. Движение капитала и товарооборот являются основой современной экономической системы. Поэтому выдача кредитов поощряется и стимулируется через средства массовой информации, путем всеобщей рекламы, маркетинговых акций и других приемов.

Кредитованием населения, в основном, занимаются банки. Банковские кредиты отличаются определенной спецификой. Так банк почти не дает кредиты из собственного капитала. Их источником становятся деньги вкладчиков и/или расчетные счета коммерческих организаций, причем эти денежные средства являются временно свободными. Помимо этого, кредит носит возмездный характер, а значит заемщик должен не только вернуть взятую сумму, но и заплатить дополнительный процент.

Сам по себе кредит – это финансовая услуга, а значит, как и любая другая услуга, она должна приносить прибыль организации. Ключевыми факторами в этом случае становятся размер процентной ставки и возвратность кредита.

И если установить минимально допустимую процентную ставку по кредиту еще возможно, опираясь на процент вклада, издержки при оказании услуг, то предсказать будущее поведение заемщика весьма непросто.

Первыми попытками решить эту проблему было введение достаточно подробного анкетирования. Затем специалисты банка проверяли анкетные данные и принимали решение о выдаче кредита или отказе в нем.

Со временем в банках стали накапливаться кредитные истории заемщиков. В общем виде - это совокупность анкетных данных и поведения заемщика, например, отдал ли он кредит, были ли платежи регулярными и т.д. Возросший объем информации стал толчком к автоматизации анализа и

принятия решений по выдаче кредита. Это так называемая задача кредитного скоринга.

Варианты этой задачи могут немного отличаться, но суть заключается в предсказании некоторого значения по имеющимся анкетным данным. Например, можно определить вероятность просрочки кредита заемщиком, а конкретное решение оставить за специалистом. Другой вариант, полностью переложить на компьютер решение этого вопроса, тогда задача сводится к определению действия, т.е. дать или не дать кредит.

Эволюция технологий, широкое применение интернета в коммерческой деятельности и сотрудничество между торговыми предприятиями и банками привели к появлению новой задачи в банковской сфере.

Теперь в магазинах электроники можно взять кредит на тот или иной товар. Для этого уже не нужно идти в банк. Во многих крупных гипермаркетах техники работают выездные точки кредитования от разных банков. Помимо физического присутствия в магазинах, покупателям доступны онлайн формы для подачи заявок на кредит через интернет-магазины электроники. Покупатель оставляет заявки на кредит и ожидает решения банка. После одобрения заявки у человека появляется выбор между несколькими кредитными учреждениями.

Такая свобода выбора у покупателей и конкуренция между банками приводит к повышению издержек при обработке кредитных заявок. Ведь проверить качество заемщика, оценить потенциальные риски, рассчитать процентную ставку нужно для всех поступающих заявок, а в конечном итоге кредит в этом банке возьмёт лишь какая-то часть покупателей. Остальные заявки будут обработаны без какой-либо отдачи.

Для сокращения этих издержек необходимо разработать модуль рекомендаций, который с хорошей вероятностью сможет определять покупателей, которые действительно возьмут кредит в этом банке. А значит и проводить для них полный цикл проверок будет экономически оправдано.

Так как рекомендательные системы, в основном, строятся на основе конкретных исторических данных, которые у каждого банка будут отличаться, то разработка универсального решения затруднительна. Поэтому в данной квалификационной работе модуль рекомендаций разрабатывается для определенного банка, а не всех банков в целом.

1.3. Анализ данных

1.3.1. Источник данных

Для выполнения квалификационной работы взят набор данных с соревнования, организованного банком Тинькофф. Конкурс проводился с 10 января 2017 года по 27 февраля 2017 года. Перед участниками была поставлена задача бинарной классификации. Суть задачи заключается в следующем.

В магазинах электроники можно взять кредит на тот или иной товар. Для этого уже не нужно идти в банк. Во многих крупных гипермаркетах техники работают выездные точки кредитования от разных банков. Покупатель оставляет заявки на кредит и ожидает решения банка. После одобрения заявки у человека появляется выбор между несколькими кредитными учреждениями. Этот выбор в отношении банка Тинькофф и нужно предсказать.

Другими словами, нужно определить выберет ли тот или иной человек именно банк Тинькофф, если его заявка будет одобрена банком. Скорее всего, решение этой задачи поможет банку не проводить лишнюю работу с клиентами, если они все равно могут не выбрать этот банк.

Для конкурса был предоставлен набор данных (далее - датасет) с описанием заявок на кредит из магазинов электроники. Все эти заявки были одобрены банком.

1.3.2. Подготовка данных

Данные редко попадают к аналитику в виде пригодном для использования методов машинного обучения. Поэтому предварительно данные

нужно обработать и подготовить. Качество данных напрямую влияет на полученные результаты на этапе моделирования. Чем качественней произведена предобработка, тем меньше сил будет тратиться на попытку объяснений аномальных результатов. Обычно этот этап включает в себя очистку и преобразование данных.

1.3.3. Очистка данных

Очистка данных предполагает устранение ошибок в данных. Наиболее типичные ошибки:

- Орфографические ошибки и опечатки. Появляются при вводе данных в базу оператором. Могут значительно повлиять на качество моделирования. Например, москва и Москва, 100 и 1000.
- Отсутствие данных. Могут возникать из-за отсутствия данных при вводе. Также возможно появление при создании сводных таблиц, разных по структуре.
- Фиктивные значения. Появляются, когда поле обязательно для заполнения, но точных данных у оператора нет. Тогда вводятся хоть какие-нибудь данные. Чаще всего такие ошибки появляются в номерах телефона, адресах, зарплатах и т.д.
- Логически неверные данные. Под эту ошибку подпадают значения, которые не соответствуют смыслу поля таблицы. Например, в регионе указывают город или район.
- Закодированные данные. Сокращенная запись или кодировка реальных данных. Применяется с целью уменьшения занимаемого места.
- Составные значения. Обычно характерно для полей с произвольным вводом текстовой информации. Особенно становится проблемой, если нет проверки и формата ввода.

1.3.4. Очистка данных датасета

Чтобы оценить качество набора данных и необходимость предварительной очистки данных, необходимо посмотреть на сами данные.

Общие представления о данных можно получить по первым и последним строчкам датасета (таблица 1 и таблица 2).

Таблица 1 - Первые пять строк из датасета

	gender	age	marital_status	job_position	credit_sum	credit_month	tariff_id	score_shk	education	living_region	monthly_income	credit_count	overdue_credit_count	open_account_flg
client_id														
1	M	48	MAR	UMN	59998.00	10	1.6	0.770249	GRD	КРАСНОДАРСКИЙ КРАЙ	30000.0	1.0	1.0	0
2	F	28	MAR	UMN	10889.00	6	1.1	0.248514	GRD	МОСКВА	43000.0	2.0	0.0	0
3	M	32	MAR	SPC	10728.00	12	1.1	0.459589	SCH	ОБЛ САРАТОВСКАЯ	23000.0	5.0	0.0	0
4	F	27	DIV	SPC	12009.09	12	1.1	0.362536	GRD	ОБЛ ВОЛГОГРАДСКАЯ	17000.0	2.0	0.0	0
5	M	45	MAR	SPC	16908.89	10	1.1	0.421385	SCH	ЧЕЛЯБИНСКАЯ ОБЛАСТЬ	25000.0	1.0	0.0	0

Таблица 2 - Последние пять строк из датасета

	gender	age	marital_status	job_position	credit_sum	credit_month	tariff_id	score_shk	education	living_region	monthly_income	credit_count	overdue_credit_count	open_account_flg
client_id														
170742	F	27	UNM	SPC	64867.00	12	1.1	0.535257	GRD	РЕСПУБЛИКА ТАТАРСТАН	40000.0	6.0	0.0	0
170743	F	24	MAR	SPC	17640.00	6	1.6	0.573287	SCH	САНКТ-ПЕТЕРБУРГ Г	30000.0	1.0	0.0	0
170744	F	31	UNM	SPC	27556.47	10	1.32	0.416098	GRD	ПРИМОРСКИЙ КРАЙ	40000.0	1.0	0.0	0
170745	F	53	DIV	PNA	6189.00	12	1.1	0.482595	SCH	ПЕНЗЕНСКАЯ ОБЛ	31000.0	2.0	0.0	0
170746	M	49	MAR	SPC	12787.00	10	1.1	0.316087	GRD	ОБЛ МОСКОВСКАЯ	40000.0	3.0	0.0	0

Всего в датасете 14 столбцов и 170746 строк. Как можно увидеть в датасете представлены значения в разных форматах и на разных языках. Описание колонок и их возможные значения предоставлены организатором конкурса указаны в таблице 3.

Таблица 3 – Описание столбцов датасета

Описание столбцов датасета	
Название	**Описание**
client_id	Уникальный идентификатор клиента
education	Образование
job_position	Работа
marital_status	Семейное положение
gender	Пол
age	Возраст
credit_sum	Сумма кредита
credit_month	Срок кредитования
tariff_id	Номер предлагаемого тарифа
living_region	Регион проживания
monthly_income	Месячный заработок
credit_count	Количество кредитов у клиента
overdue_credit_count	Количество просроченных кредитов клиента

Для одного столбца, а именно score_shk, описание организатором не предоставлено. Возможно оно как-то связано с прогнозом возврата кредита или с тарифной политикой.

1.3.5. Описание возможных значений в столбцах

В таблице 4 представлены все значения, которые может принимать признак “образование”.

Таблица 4 - Образование

education (образование)	
Значение признака	**Описание**
SCH	Начальное, среднее
PGR	Второе высшее
GRD	Высшее
UGR	Неполное высшее
ACD	Ученая степень

Некоторые значения в колонке job_position отражают не должности, а нечто другое и могут ввести в заблуждение (таблица 5). Например, в реальном мире владелец собственного бизнеса может иметь какую-либо должность, как в своем предприятии, так и в чужом. Тогда что в таком случае проставляется в этой графе? Такой же момент касается безработицы. Судя по значениям, причина безработицы может быть в инвалидности, пенсионном возрасте, отсутствию работы без причины и статусе домохозяйки. И под все это выделены отдельные значения. Скорее всего, такое многообразие ввода в этой колонке связано с ошибкой на этапе проектирования системы, в которой эти данные заносятся. Можно было бы ввести бинарные признаки работает/не работает, владелец бизнеса/нет бизнеса. Если человек работает, ввести данные об уровне должности: служащий/руководитель/специалист и т.д. Иначе ввести причину отсутствия работы. Это бы увеличило количество столбцов в таблице и полей при заполнении, но могло бы улучшить процесс моделирования на основе данных.

Таблица 5 - Работа

job_position (работа)	
Значение признака	**Описание**
SPC	Неруководящий сотрудник - специалист
DIR	Руководитель организации
HSK	Домохозяйка
INV	Не работает (инвалидность)
WOI	Работает на ИП
WRK	Неруководящий сотрудник - рабочий
ATP	Неруководящий сотрудник - обслуживающий персонал
WRP	Работающий пенсионер
UMN	Руководитель подразделения
NOR	Не работает
PNS	Пенсионер
BIS	Собственный бизнес
INP	Индивидуальный предприниматель

Похожая ситуация и в столбце marital_status (таблица 6). С точки зрения закона есть только два состояния в семейном положении: заключенный брак и его отсутствие. Все остальные значения лишь описывают причину одного из этих двух состояний. Более того, официально понятие гражданский брак в законодательстве отсутствует, и ему соответствует не столь красивый термин, как сожительство. Поэтому его в обычной жизни стараются избегать. Тоже можно было бы сделать два столбца. В одном бинарный признак семейного положения, в другом описание причин.

Таблица 6 - Семейное положение

marital_status (семейное положение)	
Значение признака	**Описание**
UNM	Холост/не замужем
DIV	Резведен (а)
MAR	Женат/замужем
WID	Вдовец, вдова
CIV	Гражданский брак

Признак пол представлен двумя очевидными значениями (таблица 7).

Таблица 7 – Пол

gender (пол)	
F	Женский
M	Мужской

Таблица 8 - Типы данных в столбцах

Столбец	Тип
gender	object
age	int64
marital_status	object
job_position	object
credit_sum	float64
credit_month	int64
tariff_id	object
score_shk	float64
education	object
living_region	object
monthly_income	float64
credit_count	float64
overdue_credit_count	float64
open_account_flg	int64

Таблица 9 - Описание числовых столбцов

	count	mean	std	min	25%	50%	75%	max
age	170746.0	36.50	10.55	18.0	28.00	34.00	43.00	71.00
credit_sum	170746.0	26095.05	16234.79	2736.0	14908.00	21229.00	32068.00	200000.00
credit_month	170746.0	10.98	3.54	3.0	10.00	10.00	12.00	36.00
score_shk	170746.0	0.47	0.12	0.0	0.38	0.46	0.55	1.13
monthly_income	170745.0	40138.29	25044.21	5000.0	25000.00	35000.00	50000.00	950000.00
credit_count	161516.0	2.11	1.78	0.0	1.00	2.00	3.00	21.00
overdue_credit_count	161516.0	0.05	0.21	0.0	0.00	0.00	0.00	3.00
open_account_flg	170746.0	0.18	0.38	0.0	0.00	0.00	0.00	1.00

Таблица 10 - Описание объектных столбцов

	count	unique	top	freq
gender	170746	2	F	88697
marital_status	170746	5	MAR	93956
job_position	170746	18	SPC	134680
tariff_id	170746	33	1.1	69355
education	170746	5	SCH	87539
living_region	170554	301	ОБЛ МОСКОВСКАЯ	12228

Из таблиц выше видно, что в датасете присутствуют некоторые проблемы, которые необходимо решить перед созданием и обучением моделей (таблица 11).

Таблица 11 – Сводная таблица проблем в данных

Проблема	**Столбцы**
Отсутствие данных	monthly_income, credit_count, overdue_credit_count, living_region
Логически неверные данные и орфографические ошибки (опечатки)	living_region
Закодированные и текстовые данные	gender, marital_status, job_position, tariff_id, education, living_region

1.3.6. Заполняем пропущенные значения

Для некоторых алгоритмов машинного обучения необходимо обеспечить отсутствие пропущенных значений. Иначе их нельзя применить или их результаты будут не самыми лучшими.

Существует несколько вариантов обработки и заполнения строк с пропусками:

- Удаление строки с пропуском. Если количество данных позволяет проще всего удалить такие примеры.
- Подстановка фиктивных значений. Например, null или 0.

- Подстановка среднего значения или медианы.
- Подстановка часто используемого значения. Обычно применяется для категориальных значений.
- Предсказание наиболее вероятного значения.

Заполним числовые столбцы медианным значением. При заполнении пропущенных значений, всегда существует риск искажения в данных. Но это только риск, а окончательная оценка возможна только при проведении эксперимента. На этапе машинного обучения может выясниться, что строки с пропущенными значениями лучше удалить, тогда для улучшения качества модели это и будет сделано.

Столбцы с категориальными значениями заполним часто используемым значением.

1.3.7. Обработка столбца living_region

Столбец living_region содержит регионы России в которых были сформированы заявки на кредит. В РФ существует 85 субъектов федерации, а уникальных значений в этом столбце 301. Что предполагает наличие дублирующих и ошибочных записей.

Из частичного перечня значений видно, что регионы записаны в базу совершенно по-разному.

```
'КРАСНОДАРСКИЙ КРАЙ' 'МОСКВА' 'ОБЛ САРАТОВСКАЯ' 'ОБЛ ВОЛГОГРАДСКАЯ'
'ЧЕЛЯБИНСКАЯ ОБЛАСТЬ' 'СТАВРОПОЛЬСКИЙ КРАЙ' 'ОБЛ НИЖЕГОРОДСКАЯ'
'МОСКОВСКАЯ ОБЛ' 'ХАНТЫ-МАНСИЙСКИЙ АВТОНОМНЫЙ ОКРУГ - ЮГРА'
'КРАЙ СТАВРОПОЛЬСКИЙ' 'САНКТ-ПЕТЕРБУРГ' 'РЕСП. БАШКОРТОСТАН'
'ОБЛ АРХАНГЕЛЬСКАЯ' 'ХАНТЫ-МАНСИЙСКИЙ АО' 'РЕСП БАШКОРТОСТАН'
'ПЕРМСКИЙ КРАЙ' 'РЕСП КАРАЧАЕВО-ЧЕРКЕССКАЯ' 'САРАТОВСКАЯ ОБЛ'
'ОБЛ КАЛУЖСКАЯ' 'ОБЛ ВОЛОГОДСКАЯ' 'РОСТОВСКАЯ ОБЛ' 'УДМУРТСКАЯ РЕСП'
'ОБЛ ИРКУТСКАЯ' 'ПРИВОЛЖСКИЙ ФЕДЕРАЛЬНЫЙ ОКРУГ' 'ОБЛ МОСКОВСКАЯ'
'ОБЛ ТЮМЕНСКАЯ' 'ОБЛ БЕЛГОРОДСКАЯ' 'РОСТОВСКАЯ ОБЛАСТЬ' 'ОБЛ КОСТРОМСКАЯ'
'РЕСП ХАКАСИЯ' 'РЕСПУБЛИКА ТАТАРСТАН' 'ИРКУТСКАЯ ОБЛАСТЬ'
'ОБЛ СВЕРДЛОВСКАЯ' 'ОБЛ ПСКОВСКАЯ' 'КРАЙ ЗАБАЙКАЛЬСКИЙ' 'СВЕРДЛОВСКАЯ ОБЛ'
'ОБЛ ОРЕНБУРГСКАЯ' 'ОБЛ ВОРОНЕЖСКАЯ' 'ОБЛ АСТРАХАНСКАЯ'
'ОБЛ НОВОСИБИРСКАЯ' 'ОБЛ ЧЕЛЯБИНСКАЯ' 'ОРЕНБУРГСКАЯ ОБЛ'
'СВЕРДЛОВСКАЯ ОБЛАСТЬ' 'ОБЛ КУРГАНСКАЯ' 'ЧЕЛЯБИНСКАЯ ОБЛ'
'НИЖЕГОРОДСКАЯ ОБЛАСТЬ' 'ТАТАРСТАН РЕСП' 'УЛЬЯНОВСКАЯ ОБЛ' 'МОСКВА Г'
'ОБЛ МУРМАНСКАЯ' 'КРАСНОЯРСКИЙ КРАЙ' 'РЕСП БУРЯТИЯ' 'РЕСП. САХА (ЯКУТИЯ)'
'ОБЛ АМУРСКАЯ' 'ХАБАРОВСКИЙ КРАЙ' 'САНКТ-ПЕТЕРБУРГ Г' 'ЯМАЛО-НЕНЕЦКИЙ АО'
'ОБЛ САМАРСКАЯ' 'ТЮМЕНСКАЯ ОБЛАСТЬ' 'ТВЕРСКАЯ ОБЛАСТЬ'
'ЯРОСЛАВСКАЯ ОБЛАСТЬ' 'ОБЛ ВЛАДИМИРСКАЯ' 'ОБЛ ЛЕНИНГРАДСКАЯ'
```

Рисунок 1 – Запись регионов в датасете

Признак с регионами включает разные варианты записей одних и тех же регионов (таблица 12).

Таблица 12 - Варианты записей регионов

Регион	**Варианты написания**
Москва	МОСКВА Г МОСКВА МОСКВА Г Г. МОСКВА Г.МОСКВА
Московская область	МОСКОВСКАЯ ОБЛ ОБЛ МОСКОВСКАЯ МОСКОВСКАЯ ОБЛАСТЬ МОСКОВСКАЯ ОБЛ.МОСКОВСКАЯ
Ханты-Мансийский автономный округ	ХАНТЫ-МАНСИЙСКИЙ АВТОНОМНЫЙ ОКРУГ - ЮГРА ХАНТЫ-МАНСИЙСКИЙ АО АО ХАНТЫ-МАНСИЙСКИЙ АВТОНОМНЫЙ ОКРУГ - Ю

Помимо разнообразия написания регионов в датасете представлены ошибочные или некорректные данные.

Таблица 13 – Некорректные и ошибочные записи в регионах

Ошибочные или некорректные записи	**Причина**
Г.ОДИНЦОВО МОСКОВСКАЯ ОБЛ	Не субъект РФ
МОСКВОСКАЯ ОБЛ	Опечатка
ПРИВОЛЖСКИЙ ФЕДЕРАЛЬНЫЙ ОКРУГ	Не субъект РФ
ДАЛЬНИЙ ВОСТОК	Не субъект РФ
КАМЧАТСКАЯ ОБЛАСТЬ	Камчатский край
ГОРЬКОВСКАЯ ОБЛ	Нижегородская область с 1990 года
ЧИТИНСКАЯ ОБЛ	Забайкальский край
МЫТИЩИНСКИЙ Р-Н	Не субъект РФ
98	Код региона

Требуется привести эти данные к единому стилю. Принято решение отбросить указания на тип субъекта (область, край и т.д.), удалить все небуквенные символы, в том числе и пробелы, привести к нижнему регистру, а ошибочные или некорректные данные заменить на актуальные субъекты или наиболее близкие к ним.

После всех преобразований получилось 83 региона. По всей видимости данные были собраны за длительный промежуток времени, начиная с 1990 года и по 2014 год. Так как отсутствуют два современных региона: Республика Крым и город Севастополь, которые вошли в состав России в 2014 году. Диаграмма принятых и отклоненных заявок по кредитам по регионам показана ниже на рисунке 2.

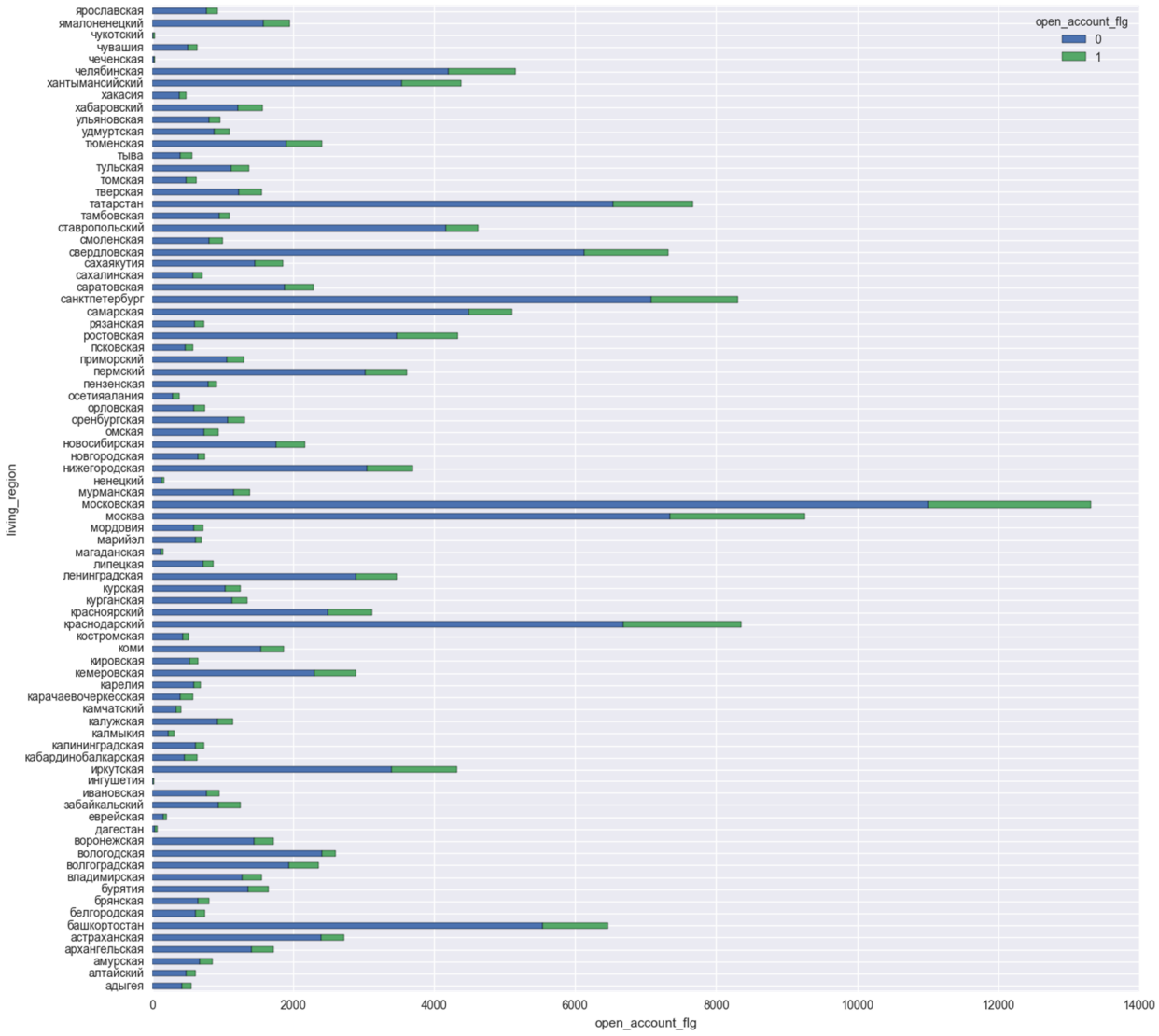


Рисунок 2 - Диаграмма принятых и отклоненных заявок по кредитам по регионам

1.3.8. Кодируем категориальные признаки

Когда категориальные данные уже очищены, можно приступить к их кодированию. Это необходимо, так как алгоритмы машинного обучения в общем случае работают с числовыми признаками. Соответственно, текстовые значения так или иначе необходимо перевести в цифры.

Различают бинарные и многоклассовые признаки. Бинарные признаки чаще всего описывают наличие или отсутствие чего-либо. Соответственно, кодируют их 0 и 1.

Многоклассовые признаки кодирует обычно одним из двух способов.

Во-первых, всем уникальным значениям признака ставят в соответствие какое-либо число, например, от 0 до n, где n - количество уникальных значений признака.

Во-вторых, используют dummy или one-hot кодирование. (описать более подробно).

Столбец с бинарным признаком (пол) кодируем в 0 и 1.

Все остальные столбцы кодируем one-hot.

1.3.9. Исследование данных

Этот этап помогает понять, как признаки соотносятся с друг другом, есть ли какие-либо явные закономерности. Еще на этом шаге оценивается распределение и наличие или отсутствие выбросов. В основном применяется визуализация и статистические методы (рисунок 3).

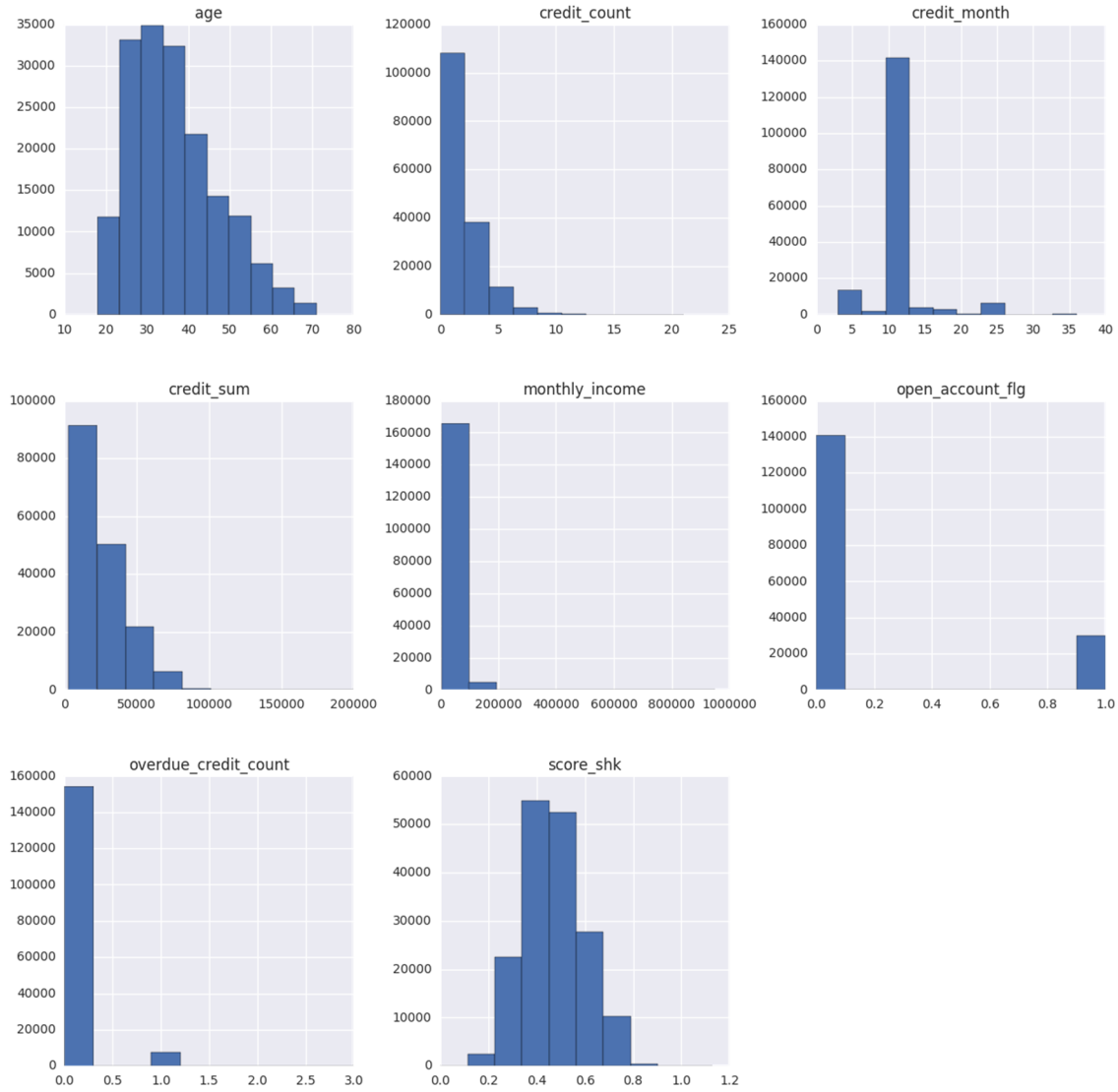


Рисунок 3 - Распределение числовых признаков

По графикам можно заметить, что в основном люди берут кредит на сумму до 100 000 рублей. Скорее всего, примеры с большей суммой можно отсечь как незначительные.

Посмотрим на зависимость суммы кредита от возраста (рисунок 4).

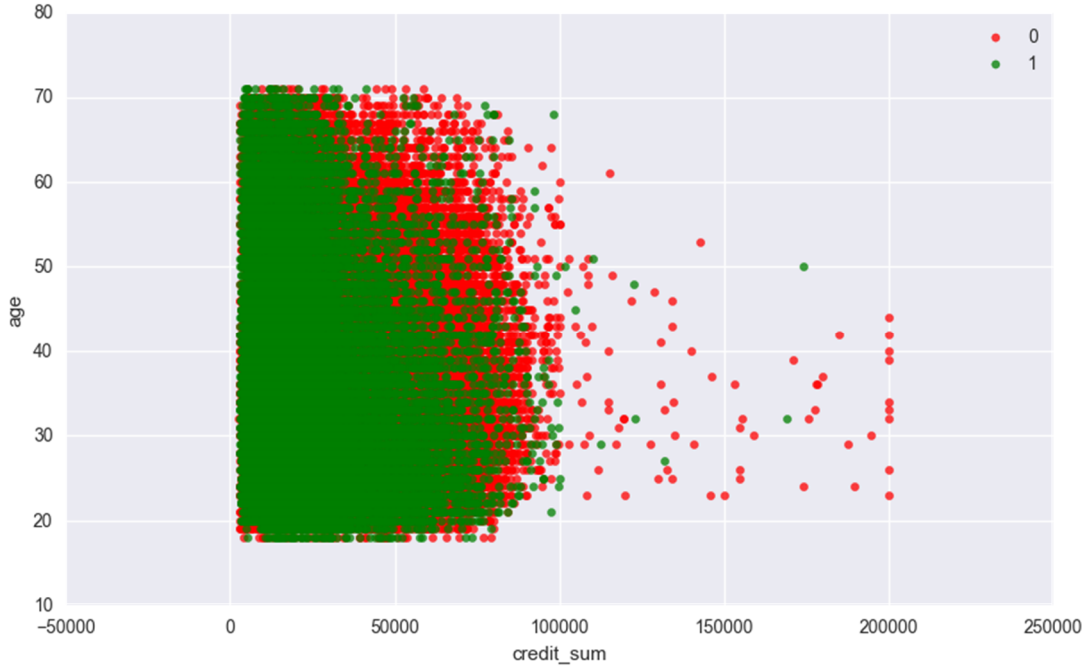


Рисунок 4 - Зависимость суммы кредита от возраста

Посмотрим на зависимость суммы кредита от зарплаты (рисунок 5).

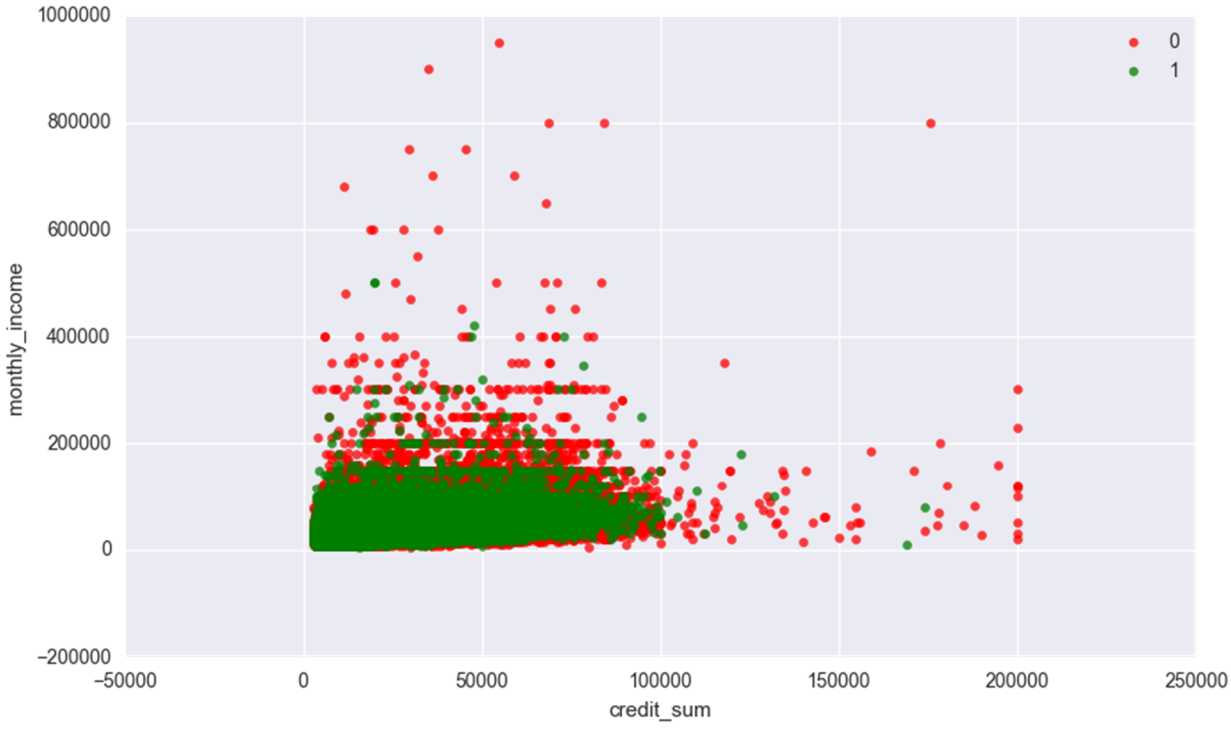


Рисунок 5 - Зависимость суммы кредита от зарплаты

Для устранения выбросов удалим все примеры с суммой кредита больше 100 000 рублей, зарплатой больше 150 000 рублей.

Получаем уже результат намного лучше (рисунок 6).

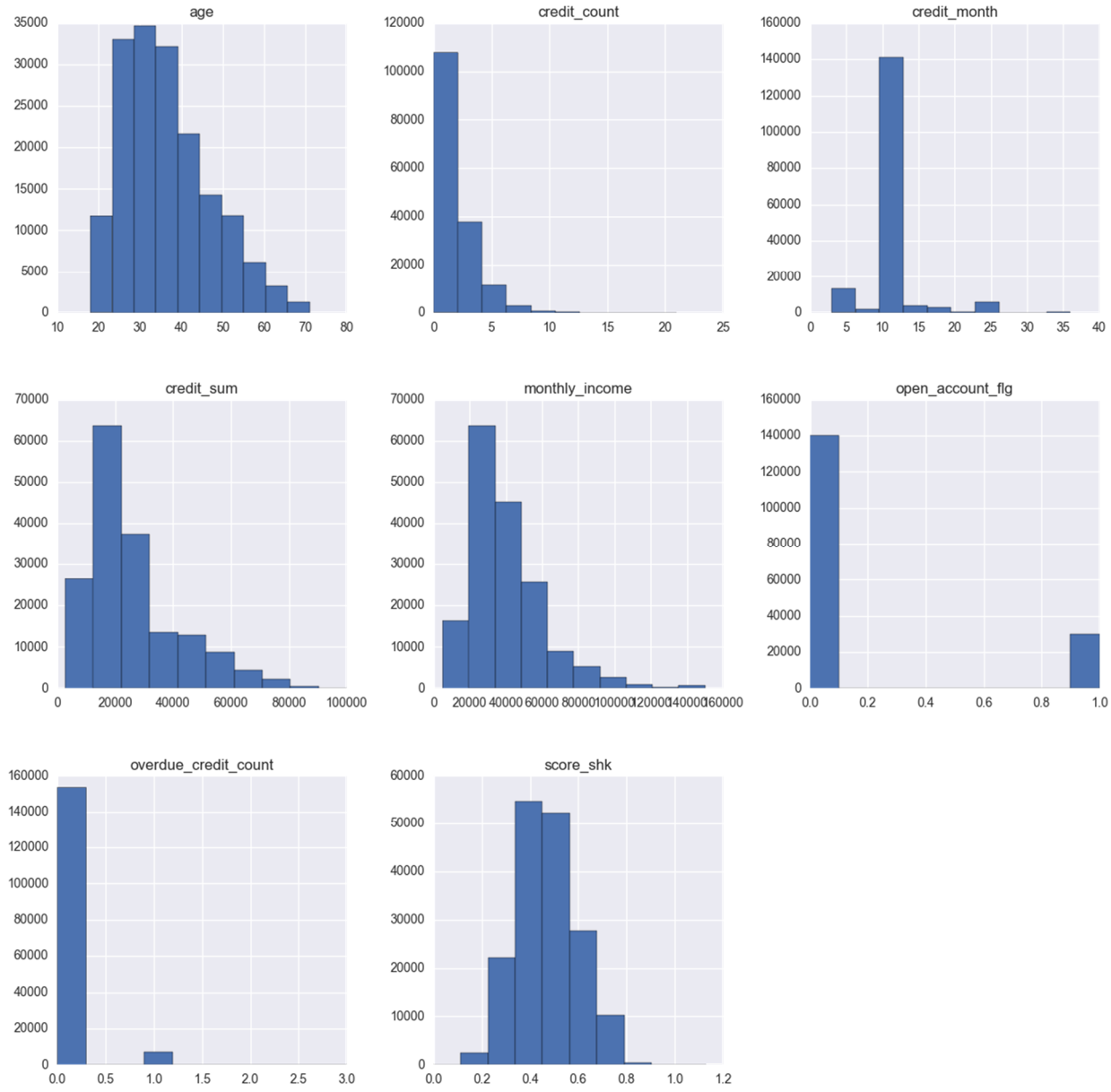


Рисунок 6 - Распределение числовых признаков после удаления выбросов

1.3.10. Приводим к нормальному распределению

Работа с выбросами, как и пропущенными значениями подразумевает в общем два подхода. Во-первых, удаление примеров с выбросами и потенциальную потерю данных. Во-вторых, преобразование данных и потенциально искажение в них.

По графикам выше видно, что у некоторых столбцов значения далеки от нормального распределения. А оно необходимо для применения некоторых

алгоритмов. Помимо удаления примеров, существует несколько вариантов приведения данных к нормальному распределению или близкому к нему.

Для этого используют нормированные Z-оценки или логарифмирование.

$$f' = \frac{f - \mu}{\sigma}$$

Здесь f - исходное значение признака, f' - нормированное значение,

μ - среднее значение признака, а σ - стандартное отклонение.

Будем использовать в качестве отправной точки простое логарифмирование.

Финальное описание датасета после всех преобразований

Таблица 14 – Сводная таблица датасета после всех преобразований

	count	mean	std	min	25%	50%	75%	max
gender	169985.0	0.479672	0.499588	0.000000	0.000000	0.000000	1.000000	1.000000
age	169985.0	3.557052	0.280846	2.890372	3.332205	3.526361	3.761200	4.262680
credit_sum	169985.0	9.989392	0.598104	7.914252	9.607706	9.960340	10.372178	11.512925
credit_month	169985.0	10.976727	3.526567	3.000000	10.000000	10.000000	12.000000	36.000000
score_shk	169985.0	0.469747	0.124184	0.000000	0.379817	0.461759	0.552641	1.128291
monthly_income	169985.0	10.458535	0.489296	8.517193	10.126631	10.463103	10.819778	11.918391
credit_count	169985.0	2.096456	1.723468	0.000000	1.000000	2.000000	3.000000	21.000000
overdue_credit_count	169985.0	0.043398	0.205620	0.000000	0.000000	0.000000	0.000000	3.000000
open_account_flg	169985.0	0.176204	0.380995	0.000000	0.000000	0.000000	0.000000	1.000000
marital_status_CIV	169985.0	0.024585	0.154856	0.000000	0.000000	0.000000	0.000000	1.000000
marital_status_DIV	169985.0	0.099385	0.299179	0.000000	0.000000	0.000000	0.000000	1.000000
marital_status_MAR	169985.0	0.549766	0.497519	0.000000	0.000000	1.000000	1.000000	1.000000
marital_status_UNM	169985.0	0.305839	0.460763	0.000000	0.000000	0.000000	1.000000	1.000000
marital_status_WID	169985.0	0.020425	0.141451	0.000000	0.000000	0.000000	0.000000	1.000000
job_position_ATP	169985.0	0.016396	0.126992	0.000000	0.000000	0.000000	0.000000	1.000000
job_position_BIS	169985.0	0.031920	0.175789	0.000000	0.000000	0.000000	0.000000	1.000000
job_position_BIU	169985.0	0.000729	0.026999	0.000000	0.000000	0.000000	0.000000	1.000000
...	...	...	...	...	...	...	...	...
living_region_приморский	169985.0	0.007554	0.086583	0.000000	0.000000	0.000000	0.000000	1.000000
living_region_ямалоненецкий	169985.0	0.011407	0.106192	0.000000	0.000000	0.000000	0.000000	1.000000
living_region_ярославская	169985.0	0.005406	0.073329	0.000000	0.000000	0.000000	0.000000	1.000000
153 rows × 8 columns								

1.4. Инфологическая и даталогическая модели

Описание инфологической модели базируется на проведенном анализе набора данных и исследовании предметной области. Для описания хорошо подходит модель “сущность-связь”.

Из набора данных можно выделить следующие объекты:

- Заказ (кредитная заявка).
- Прогноз.

Также можно выделить дополнительные объекты, связанные с заказом:

- Образование.
- Работа.
- Семейный статус.
- Регион.

Инфологическую модель в виде ER-диаграммы можно посмотреть на рисунке 7.

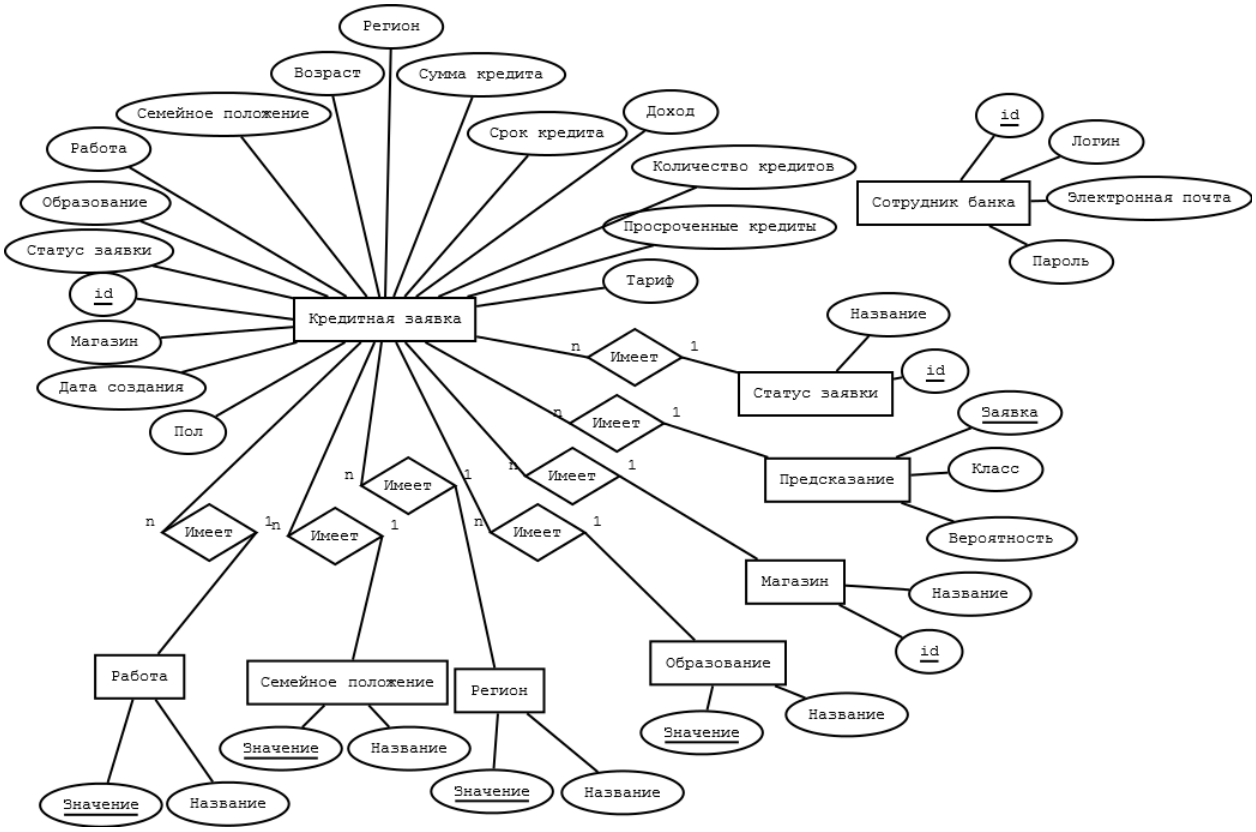


Рисунок 7 – Инфологическая модель

Таблица 15 – Описание сущностей и атрибутов

Сущность	**Атрибут**	**Тип данных**	**Краткое описание**
Заказ	id	целое	Идентификатор заказа
	education	символьное(3)	Идентификатор образования
	job_position	символьное(3)	Идентификатор работы
	marital_status	символьное(3)	Идентификатор семейного статуса
	living_region	символьное(3)	Идентификатор региона
	age	целое	Возраст
	credit_sum	целое	Сумма кредита
	credit_month	целое	Длительность кредита в месяцах
	monthly_income	целое	Доход в месяц
	credit_count	целое	Количество кредитов
	overdue_credit_count	целое	Количество просроченных кредитов
	gender	символьное(1)	Пол
	tariff_id	символьное(5)	Тариф
Прогноз	order_id	целое	Идентификатор заказа
	predict_proba	С плавающей точкой	Вероятность принадлежности к классу
	predict_class	целое	Предсказанный класс
Образование	value	символьное(3)	Идентификатор образования
	title	символьное(200)	Расшифровка
Работа	value	символьное(3)	Идентификатор работы
	title	символьное(200)	Расшифровка

Продолжение таблицы 15

Семейный статус	value	символьное(3)	Идентификатор семейного статуса
	title	символьное(200)	Расшифровка
Регион	id	целое	Идентификатор региона
	title	символьное(200)	Название региона

Даталогическую модель можно посмотреть на рисунке 8.

Рисунок 8 – Даталогическая модель датасета

1.5. Архитектура системы

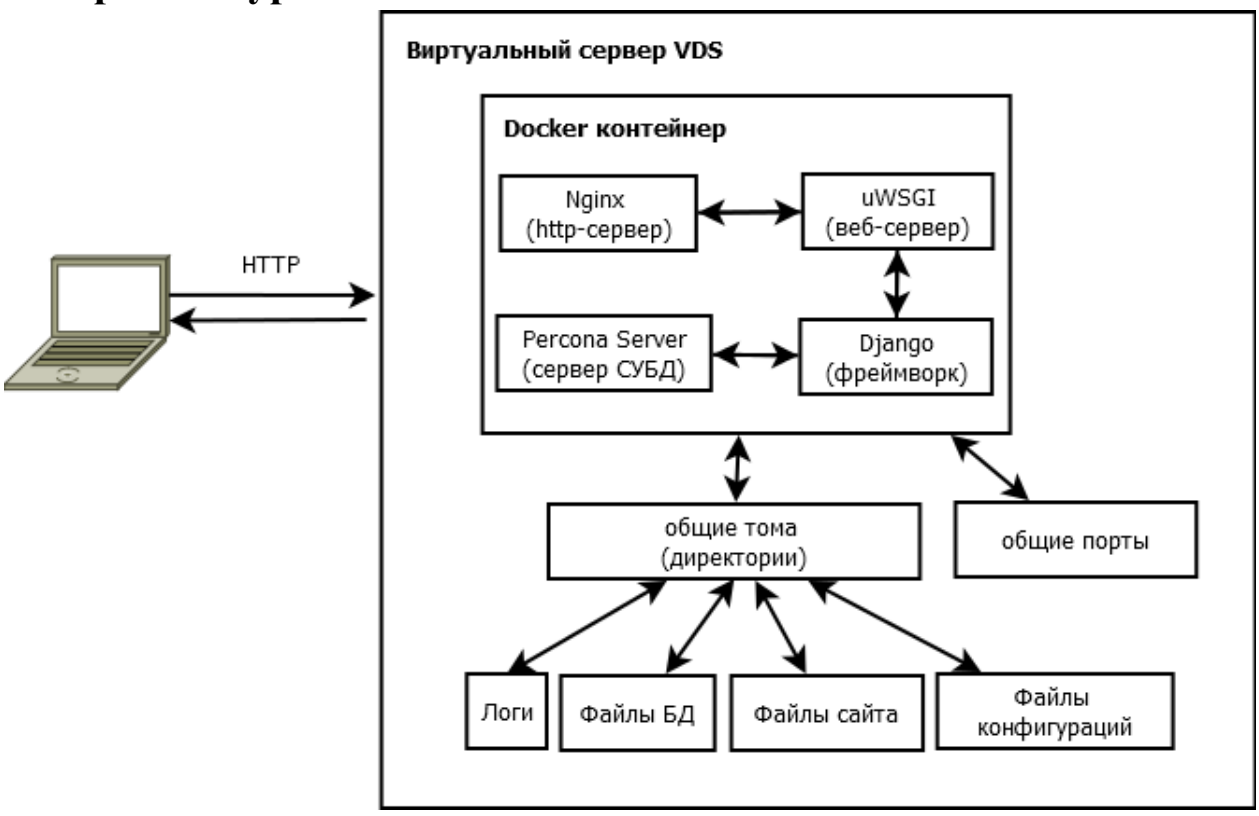


Рисунок 9 – Архитектура системы

1.6. Архитектура модуля

1.6.1. Принцип работы

Модуль рекомендация является встраиваемой подсистемой для автоматизированной банковской системы конкретного банка. Исходя из этого, модуль не предусматривает наличие какого-либо графического интерфейса. Отображение результатов работы модуля является частью процесса его внедрения в функционирующую систему.

Составными частями модуля являются обученная рекомендательная модель, программный код на языке Python для интеграции этой модели в сайты на основе Фреймворка Django. Для снижения нагрузки и сохранения рассчитанных значений модуль предусматривает сохранение вероятности и предсказанного класса в базу данных. Модуль в автоматическом режиме обрабатывает заявки при их создании, а также может произвести перерасчет при ручном запросе от специалиста банка.

Фактически, модуль получает какие-то данные на входе и отдает предсказанные значения на выходе. И основная система не должна зависеть от конкретных моделей и расчетов внутри модуля.

Ввиду того, что машинное обучение включает несколько этапов, то и модуль должен решать несколько задач, а именно:

Подготовка данных – полученные данные из базы данных кредитных заявок обрабатывать и преобразовывать в формат, пригодный для использования в математической модели.

Загрузка модели – обученную математическую модель загружать в оперативную память для дальнейшего использования.

Предсказание – по преобразованным данным предсказывать вероятность возврата клиента в банк.

Сохранение и возврат прогноза – сохранение в базу данных предсказанных значений конкретной заявки и возвращение в основную систему предсказанных результатов.

В соответствии с требованиями, модуль при включении создает таблицу в базе данных (таблица 16).

Таблица 16 – Описание таблицы модуля

Таблица	**Столбцы**	**Типы столбцов**
Prediction	order_id predict_proba predict_class	int float int

1.6.2. Источник данных

Модуль предсказывает определенные значения на основе поступающих данных из кредитных заявок. Однако в базе данных кредитные заявки хранятся в непригодном для использования в машинном обучении виде. А так как

основная система не должна знать о подходящей структуре, то предобработка данных происходит в самом модуле.

Препроцессинг включает в себя создание новых признаков, кодирование информации и удаление лишних столбцов. В итоге получается вектор со 152 признаками.

1.6.3. Результирующие данные

Подготовленные данные отправляются в загруженную модель, которая создает два выходных значения: вероятность, что определен класс 1, и определенный класс.

Эти два значения возвращаются в основную систему.

1.6.4. Внедрение

Очевидно, что сама по себе модель особой ценности не имеет. Она приобретает положительные свойства для банка лишь внутри работающей системы. Поэтому помимо разработки модели и модуля рекомендаций, необходимо его также внедрять в основной сайт банка.

Конкретный способ и вариант использования в зависимости от требований может отличаться. В рамках этой квалификационной работы разработан небольшой демонстрационный сайт, в который и будет произведена интеграция модуля рекомендаций.

1.7. Программный модуль

1.7.1. Описание пользовательского сценария

Модуль рекомендаций наиболее полезен в момент принятия решения об отклонении кредитной заявки или её одобрения для дальнейшей оценки качества заемщика. Поэтому в демонстрационном сайте именно этот сценарий и будет показан.

Итак, из разных источников (магазинов) поступают кредитные заявки на получение потребительских кредитов на покупку электроники. Эти заявки

формируются на сторонних сервисах (сайтах) непосредственно будущими заемщиками. Причем, одни и те же заявки могут поступать сразу в несколько банков или только в один по выбору покупателя.

У специалиста банка есть интерфейс, в котором он видит все поступившие заявки из разных магазинов. На странице со списком заявок видны только основные данные. Подробная информация по каждой заявке также доступна для просмотра специалистом. Причем он не может отредактировать или создать новые заявки. Единственное допустимое действие – это изменение статуса заявки.

Соответственно, его задача по имеющимся данным принять решение отправлять ли заявку в дальнейшую работу или нет. Если потенциальный заемщик скорее всего не возьмет кредит в этом банке, то лучше такую заявку не обрабатывать.

1.7.2. Модель базы данных

За основу базы данных взята инфологическая модель, созданная ранее. Однако для полноценной работы сайта и отражения основного сценария её недостаточно. Поэтому были введены дополнительные объект – Пользователь, а объект Заказ дополнен несколькими атрибутами.

Итоговую модель можно увидеть в таблице 17.

Таблица 17 – Описание сущностей и атрибутов программного модуля

Сущность	**Атрибут**	**Тип данных**	**Краткое описание**
Пользователь	id	целое	Идентификатор пользователя
	login	символьное(50)	Имя пользователя
	email	символьное(50)	Адрес электронной почты
	password	символьное(50)	Пароль
Заказ	id	целое	Идентификатор заказа
	shop	целое	Идентификатор магазина
	order_status	целое	Идентификатор статуса заявки
	education	символьное(3)	Идентификатор образования
	job_position	символьное(3)	Идентификатор работы
	marital_status	символьное(3)	Идентификатор семейного статуса
	living_region	символьное(3)	Идентификатор региона
	age	целое	Возраст
	credit_sum	целое	Сумма кредита
	credit_month	целое	Длительность кредита в месяцах
	monthly_income	целое	Доход в месяц
	credit_count	целое	Количество кредитов
	overdue_credit_count	целое	Количество просроченных кредитов
	gender	символьное(1)	Пол
	tariff_id	символьное(5)	Тариф
	created	дата	Дата создания заявки

Продолжение таблицы 17

Статус заказа	id	целое	Идентификатор статуса заявки
	title	символьное(200)	Название
Магазин	id	целое	Идентификатор магазина
	title	символьное(200)	Название
Прогноз	order_id	целое	Идентификатор заказа
	predict_proba	с плавающей точкой	Вероятность принадлежности к классу
	predict_class	целое	Предсказанный класс
Образование	value	символьное(3)	Идентификатор образования
	title	символьное(200)	Расшифровка
Работа	value	символьное(3)	Идентификатор работы
	title	символьное(200)	Расшифровка
Семейный статус	value	символьное(3)	Идентификатор семейного статуса
	title	символьное(200)	Расшифровка
Регион	id	целое	Идентификатор региона
	title	символьное(200)	Название региона

1.7.3. Архитектура сайта

Демонстрационный сайт сделан на базе фраймворка Django 1.8, что подразумевает модульную разработку. В процессе разработки были использованы следующие встроенные или сторонние модули:

- bootstrap3 – интеграция библиотеки bootstrap 3 в Django.
- django_ajax – использование ajax запросов.
- admin_tools – использование встроенного административного раздела.

Дополнительно были разработаны следующие модули:

- Demotheme – базовые шаблоны страниц и стили.
- Userprofile – переопределение и дополнение встроенной модели пользователя.
- Orders – определение модели кредитной заявки, страницы со списком заявок и описанием конкретной заявки.
- Api – определение путей для ajax запросов и их обработки.
- Stats – страница с отчетами.

1.7.4. Внедрение модуля

Результат работы модуля рекомендаций должен быть понятным и легко интерпретироваться специалистом банка. Поэтому принято решение выводить результат не в виде конкретных значений, а использовать цветовое кодирование. Т.е. добавить в список заявок колонку, в которой в зависимости от предсказанного значения будет выводиться красный (не возьмет кредит) или зеленый (возьмет кредит) прямоугольник. Это позволит наглядно классифицировать заявки и не тратить время на обдумывание результата.

Так как заявок может быть очень много и поступать они могут в реальном времени, то запускать предсказание по каждой заявке не целесообразно. Поэтому модуль рекомендаций будет срабатывать каждый раз при создании новой заявки. Соответственно, специалист банка сможет увидеть информацию по заявкам уже с учетом рекомендации модели. Для этого в Django предусмотрен механизм сигналов, которые запускаются при определенных условиях. И любой модуль может подписываться на те или иные сигналы.

Однако, встраивание модуля идет в уже работающей системе, что подразумевает наличие уже созданных, но еще не обработанных заявок. Поэтому при интеграции модуля предусмотрен ручной запуск прогноза по каждой заявке. Для этого добавлена дополнительная кнопка и кнопка для отправки запроса на прогноз. После обработки запроса, информация будет обновлена без перезагрузки страницы.

Ручной механизм проверки может быть также полезен, когда будет изменена или дополнена рекомендательная модель, что позволит пересчитать ранее полученные результаты.

Спрогнозированные значения сохраняются в базу данных.

1.7.5. Рабочее окружение

Для разработки сайта и модуля требуется настроить рабочее окружение на виртуальном сервере. В соответствии с техническим заданием на сервер должна быть установлена операционная система CentOS 6.8, а также python 3.5, Django 1.8, Percona Server 5.6.

Так как в ТЗ присутствуют ограничения на язык и Фреймворк, то для анализа данных и машинного обучения также будут применяться библиотеки, написанные на языке python и совместимые с версией 3.5, а именно:

- Pandas – открытое ПО для анализа и структурирования данных на языке Python. Предлагает удобное манипулирование данными, создание сводных таблиц, фильтрацию и изменение значения, выборки и т.д.
- Numpy – открытое ПО для научных вычислений, предоставляющие многомерные массивы и функции для работы с ними.
- Matplotlib, seaborn – открытое ПО для визуализации данных и расчетов.
- Sklearn – открытое ПО для машинного обучения. Содержит множество готовых алгоритмов, а также классы для создания своих реализаций и модификации существующих.
- Scipy – открытое ПО для научных вычислений. Содержит статистические и другие методы.
- uWSGI – веб-сервер для запуска python-приложений в сети интернет. Необходим для работы Django.

- Nginx – http-сервер и обратный прокси сервер. Используется для балансировки нагрузки, раздачи статики и других задач. В последних версиях предусматривается расширение через динамические модули.

Для изолирования окружения и облегчения повторного развёртывания приложений принято решение использовать Docker.

Docker – свободное ПО для автоматизации развертывания и управления приложениями (виртуализация на уровне ОС). Позволяет сохранять окружение в контейнер для дальнейшего переноса на разные Linux системы.

Docker-образ строится также на базе CentOS 6.8. В процессе создания образа в него устанавливаются все необходимые библиотеки. Также указываются общие порты и директории, которые будут доступны из хост-машины. В результате на хост-машине будут доступны все файлы приложений, базы данных и логи (рисунок 10).

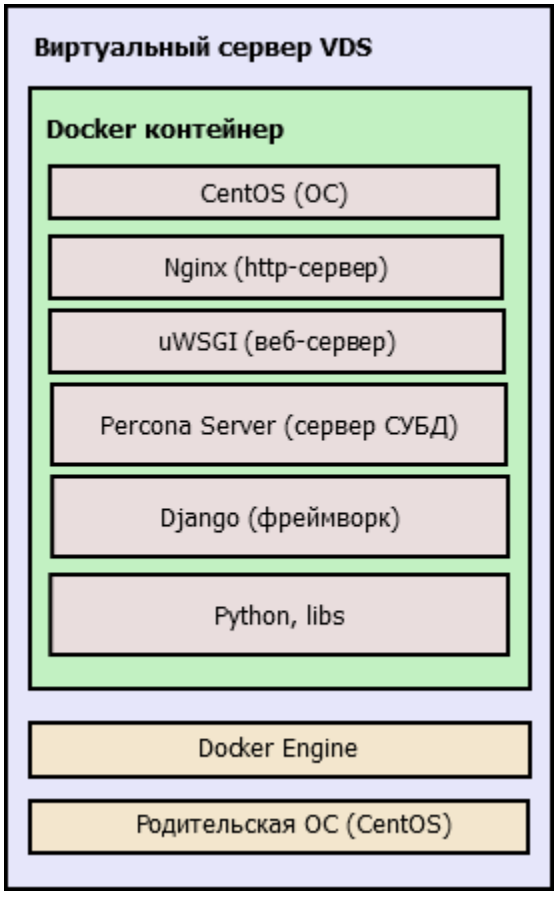


Рисунок 10 – Рабочее окружение

В процессе установлены все необходимые зависимости для этих библиотек.

1.8. Сравнение аналогов

Автоматизация банковских бизнес-процессов является одной из самых востребованных задач. В частности, для розничного кредитования созданы программный комплексы “кредитных конвейеров” для работы с заявками из разных источников. Однако, продуктов, связанных с предсказанием взятия кредитов потенциальным заемщиков, на рынке нет. Скорее всего это связано со специфичностью задачи и сильной зависимостью от данных конкретного банка, поэтому создать массовый продукт достаточно сложно или почти невозможно. Поэтому сравнивать модуль рекомендаций будем с наиболее близкими решениями из банковского сектора.

Для сравнения представленных вариантов воспользуемся методом взвешенной суммы. Данный метод позволяет объединить ряд критериев сравнения в один интегральный показатель, по которому затем выбирается наилучший вариант, соответствующий максимальному значению этого интегрального показателя. Интегральный показатель рассчитывается по формуле:

$$I = \sum_{i=1}^{n} w_i x_i$$, где:

- I – интегральный показатель,
- w_i – весовой коэффициент фактора i (сумма всех весовых коэффициентов равна 1),
- x_i – значение фактора i,
- n – количество критериев

В таблице 18 представлены сводные данные по аналогам и разрабатываемому модулю рекомендаций.

Таблица 18 – Сравнительная характеристика аналогов

Критерии сравнения	Весовой коэффициент	LMS	Кредит-Конвейер	First Loan Bank	Модуль рекомендаций
Интерфейс	0,1	5	7	7	6
Наглядность	0,1	4	8	7	7
Интеграция	0,2	6	9	8	9
Функциональность	0,1	5	8	7	7
Машинное обучение	0,3	0	2	4	8
Использование собственных данных банка	0,2	3	4	4	8
Интегральный показатель	1	3,2	5,5	5,7	7,8

Все критерии выбраны на основании ТЗ.

Самым важным критерием выбрано наличие машинного обучения с коэффициентом 0,3. Многие системы по-прежнему полагаются на принятие решений в ручном режиме или на основе экспертных систем без изучения имеющихся данных в банке.

Также важными критериями являются использование собственных данных банка и возможность интеграции в существующую системы (коэффициент 0,2). Под использованием собственных данных понимается использование баз данных или хранилищ данных для создания моделей на их основе. Интеграция подразумевает более простое внедрение в существующие систему, а также возможность обновления и замены математической модели для актуализации работы модуля.

Интерфейс, наглядность и функциональность имеют более низкий коэффициент 0,1, чем другие критерии. Интерфейс отвечает за графическое оформление работы системы в целом. Наглядность подразумевает простоту восприятия информации, которая предоставляется рекомендательной системой. Функциональность отвечает за выполнение основных функций и наличие дополнительных.

По итогам сравнения с аналогами можно сделать вывод, что модуль рекомендаций может стать востребованным программным средствам среди банков.

2. Технологическая часть

2.1. Страница входа

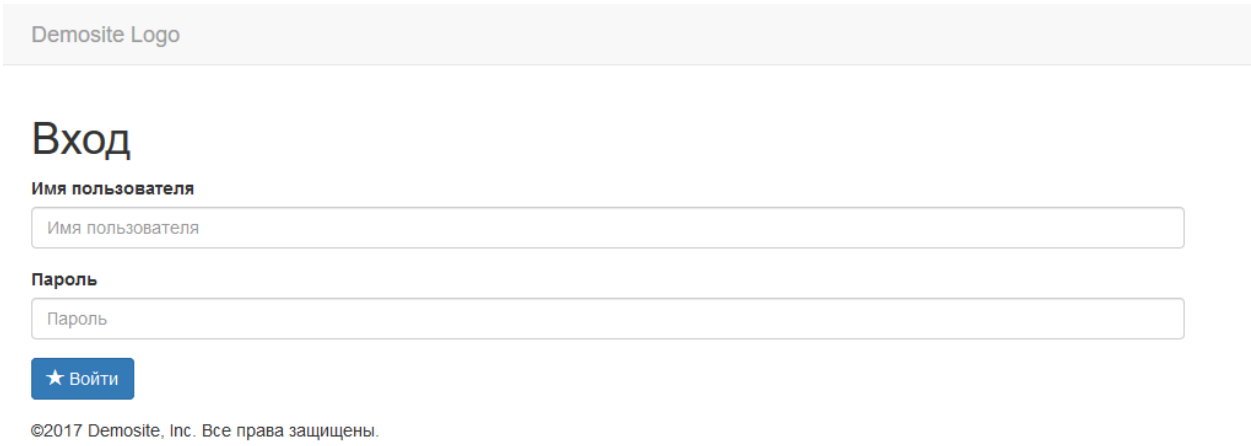
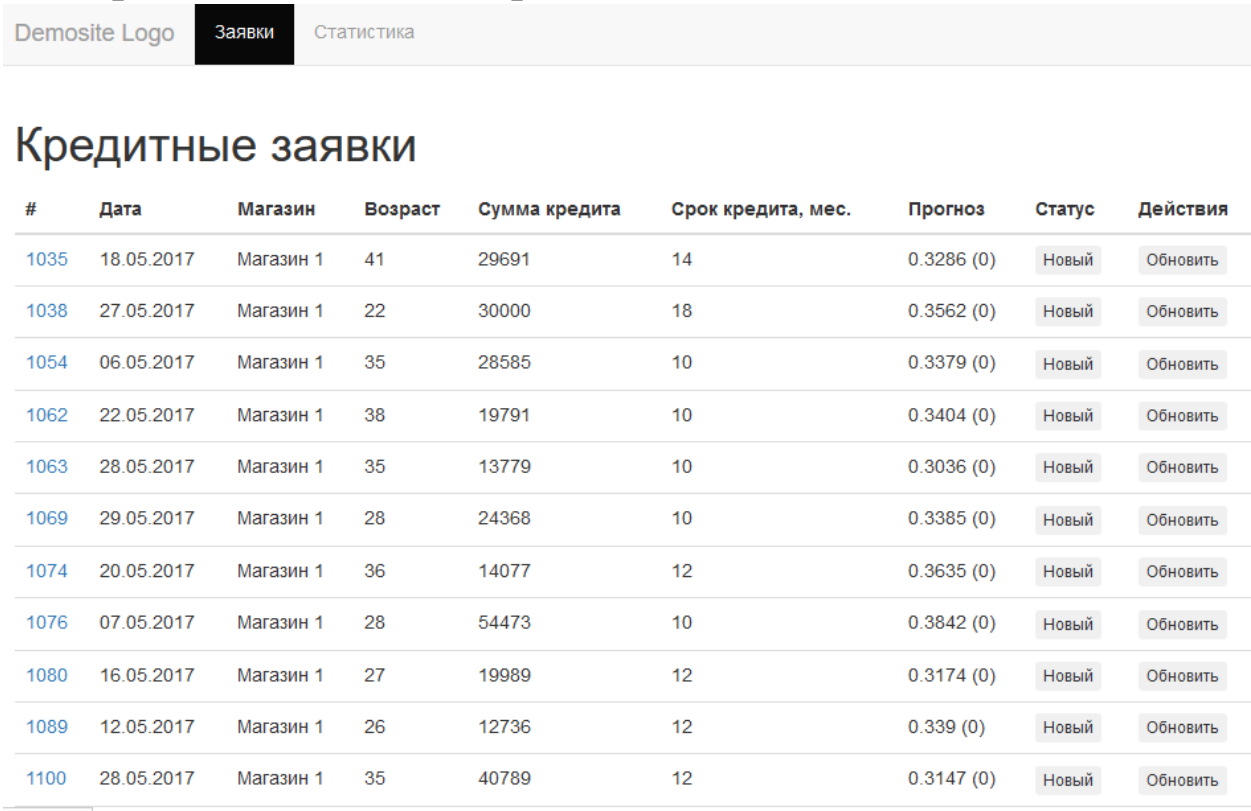


Рисунок 11 – Страница входа

2.2. Страница со списком кредитных заявок



#	Дата	Магазин	Возраст	Сумма кредита	Срок кредита, мес.	Прогноз	Статус	Действия
1035	18.05.2017	Магазин 1	41	29691	14	0.3286 (0)	Новый	Обновить
1038	27.05.2017	Магазин 1	22	30000	18	0.3562 (0)	Новый	Обновить
1054	06.05.2017	Магазин 1	35	28585	10	0.3379 (0)	Новый	Обновить
1062	22.05.2017	Магазин 1	38	19791	10	0.3404 (0)	Новый	Обновить
1063	28.05.2017	Магазин 1	35	13779	10	0.3036 (0)	Новый	Обновить
1069	29.05.2017	Магазин 1	28	24368	10	0.3385 (0)	Новый	Обновить
1074	20.05.2017	Магазин 1	36	14077	12	0.3635 (0)	Новый	Обновить
1076	07.05.2017	Магазин 1	28	54473	10	0.3842 (0)	Новый	Обновить
1080	16.05.2017	Магазин 1	27	19989	12	0.3174 (0)	Новый	Обновить
1089	12.05.2017	Магазин 1	26	12736	12	0.339 (0)	Новый	Обновить
1100	28.05.2017	Магазин 1	35	40789	12	0.3147 (0)	Новый	Обновить

Рисунок 12 – Страница кредитные заявки

2.3. Страница кредитной заявки

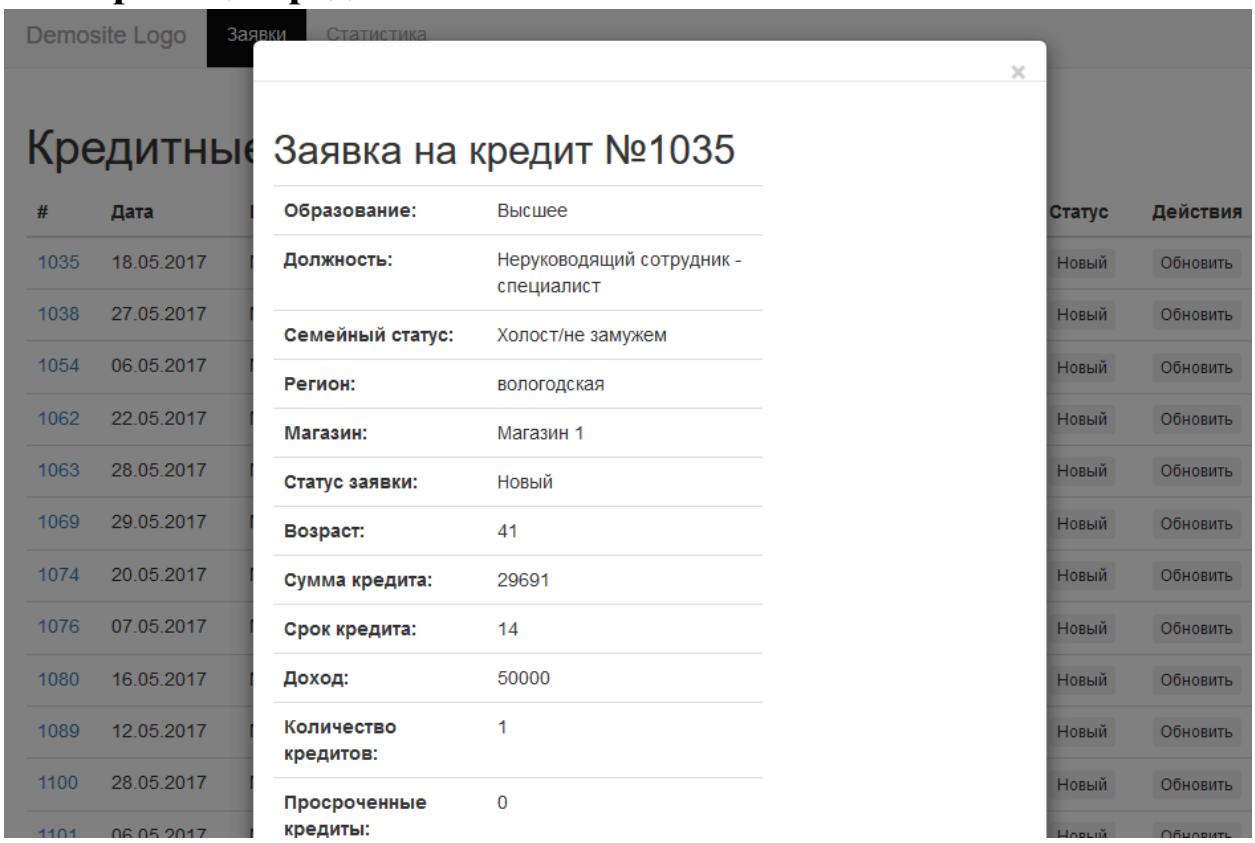


Рисунок 13 – Окно с описанием заявки

2.4. Страница статистики

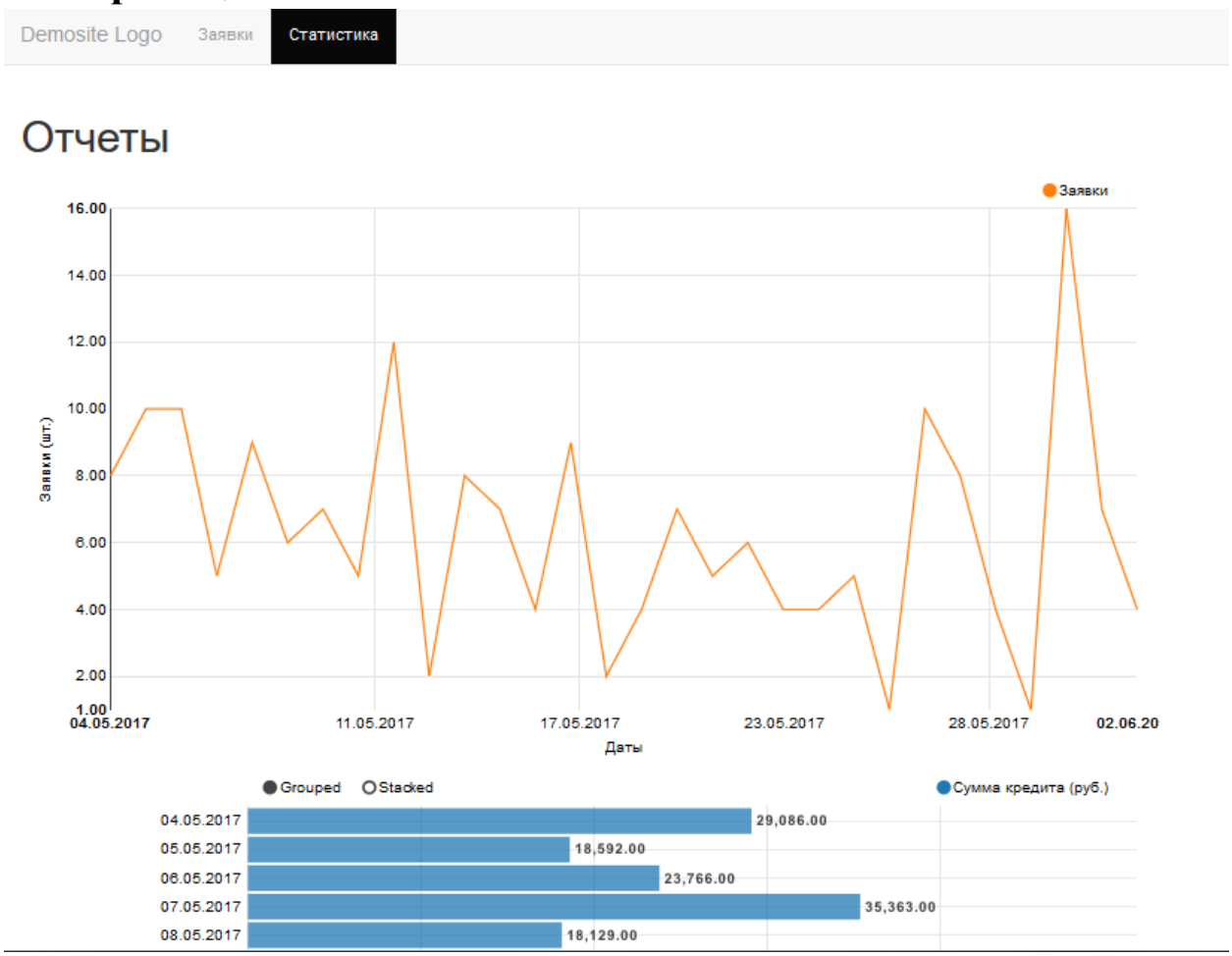


Рисунок 14 – Страница со статистикой

3. Научно-исследовательская часть

3.1. Постановка задачи

Основная цель исследований – подобрать модель машинного обучения для модуля рекомендаций автоматизированной банковской системы, который позволит специалисту банка принимать решение о дальнейших действиях с каждой вновь поступившей кредитной заявкой.

Рекомендации будут формироваться на основе исторических данных с признаками и принятым решением того или иного человека, предоставленных организатором конкурса.

Данная задача относится к задачам обучения по прецедентам, а именно, к обучению с учителем, так как результат известен заранее.

Более строго этот тип задач описывается следующим образом.

Задано множество объектов X, множество допустимых ответов Y, и существует целевая функция (target function) $y^*: X \to Y$, значения которой $y_i = y^*(x_i)$ известны только на конечном подмножестве объектов $\{x_1, \dots, x_l\} \subset X$. Пары “объект–ответ” (x_i, y_i) называются прецедентами. Совокупность пар $X^l = (x_i, y_i)_{i=1}^l$ называется обучающей выборкой (training sample).

Задача обучения по прецедентам заключается в том, чтобы по выборке X^l

восстановить зависимость y*, то есть построить решающую функцию (decision function) a: X → Y, которая приближала бы целевую функцию y*(x), причём не только на объектах обучающей выборки, но и на всём множестве X.

Каждый объект обладает определенным набором признаков. Конкретный признак является результатом измерения какой-то характеристики.

Признаки бывают нескольких видов, а именно:

- Бинарный (значения {0,1});
- Номинальный (конечное множество);
- Порядковый (конечное упорядоченное множество);
- Количественный (□).

Если все признаки относятся к одному типу, то данные называют

однородными, иначе - разнородными.

В зависимости от того, какие значения может принимать Y, различают задачи:

- Классификации;
- Регрессии.

Наша задача относится к бинарной классификации, т.е. интересующий нас ответ может принимать только одно из двух значений – 0 или 1.

По сути, решая задачу классификации, мы должны научить машину “угадывать” определенный класс объекта по заранее известным признакам. Так как это задача обучения с учителем, нам известен правильный класс объекта. И поэтому мы сможем точно узнать, где алгоритм ошибается. А значит, сможем попытаться минимизировать ошибки каждого конкретного алгоритма.

3.2. Метрика AUC ROC

Логично предположить, что не все алгоритмы будут решать задачу идеально. Одни будут ошибаться больше, другие меньше. Поэтому заранее определяется метрика, по которой будут сравниваться алгоритмы между собой.

По условиям конкурса такой метрикой будет AUC ROC, т.е. площадь под ROC-кривой (англ. receiver operating characteristic, рабочая характеристика приёмника). Именно эту метрику чаще всего используют в задачах бинарной классификации.

Так как это оценка бинарной классификации, то для её понимания и построения требуется ввести дополнительные понятия. Обычно один класс принимается за положительный, другой за отрицательный. Причем деление это весьма субъективное и зависит от конкретной задачи.

В нашей задаче положительным классом будем считать значение “1”, т.е. клиент взял кредит именно у нужного банка. Соответственно, отрицательным классом будет значение “0”, т.е. клиент взял кредит у какого-то другого банка.

При попытке определить класс у какого-то объекта, могут быть получены разные исходы. Например, правильное определение положительного класса и неправильное определение отрицательного класса. Все варианты указаны в

таблице 18.

Таблица 18 – Варианты определения классов

	Данные	
Классификатор	*Положительно*	*Отрицательно*
Положительно	TP	FP
Отрицательно	FN	TN

Где:

- TP (True Positives) – верно классифицированные положительные примеры;
- TN (True Negatives) – верно классифицированные отрицательные примеры;
- FN (False Negatives) – положительные примеры, классифицированные как отрицательные (ошибка I рода);
- FP (False Positives) – отрицательные примеры, классифицированные как положительные (ошибка II рода).

Для построения графика ROC-кривой используют относительные значения, а именно TPR и FPR, которые откладываются по вертикали и горизонтали соответственно.

TPR (True Positives Rate) – это доля или процент истинно положительных объектов. Также её называют чувствительностью (Sensitivity) алгоритма. $TPR = \frac{TP}{TP+FN} \cdot 100\%$, показывает насколько хорошо алгоритм определяет положительный класс. В нашем случае, это качество определения, что клиент возьмет кредит в банке.

FPR (False Positives Rate) – это доля или процент ложно положительных объектов. $FPR = \frac{FP}{FP+TN} \cdot 100\%$, показывает насколько алгоритм неправильно определяет положительный класс. Т.е. помечает клиента другого банка, как целевого клиента.

Специфичность (Specificity) алгоритма – это доля или процент истинно отрицательных объектов. $SP = \frac{TN}{TN+FP} \cdot 100\%$ или $SP = 100 - FPR$.

Чем выше чувствительность, тем лучше алгоритм определяет

положительные исходы и хуже отрицательные. Со специфичностью все наоборот, чем она больше, тем лучше определяются отрицательные исходы и хуже положительные. Изменение специфичности/чувствительности алгоритма зависит от порога отсечения, т.е. от некоторого параметра алгоритма, который мы можем менять.

Для удобства анализа и сравнения качества алгоритмов используют AUC ROC – площадь под ROC-кривой, измеряется от 0 до 1. Чем выше значение AUC ROC, тем лучше классификатор. На практике значение 0.5 является наихудшим, так как сопоставимо со случайным подбросом монетки. Эта метрика не содержит информацию о чувствительности/специфичности алгоритма. Оценка качества алгоритма по AUC ROC описана в таблице 19.

Таблица 19 – Оценка качества алгоритма

Интервал	**Качество модели**
0.9 - 1.0	Отличное
0.8 - 0.9	Очень хорошее
0.7 - 0.8	Хорошее
0.6 - 0.7	Удовлетворительное
0.5 - 0.6	Неудовлетворительное

3.3. Тренировка и валидация

Оценивать качество алгоритмов на тех же данных, на которых они учатся, не представляется разумным. Поэтому для проверки моделей используют два метода.

Во-первых, это разбиение общей выборки на тренировочную и тестовую (например, 70% - тренировочная, 30% - тестовая). Соответственно, все обучение проводится только на тренировочной выборке, а проверка обобщающей способности алгоритма проводится на тестовой части. Этот метод используется, когда данных достаточно много и разбиение на части не сильно ухудшит результат.

Во-вторых, это кросс-валидация (cross-validation, CV). Здесь применяется немного другая стратегия разбиения. Наши данные разбиваются на какое-то количество групп. Далее одна из этих групп убирается, а на остальных

проводится обучение. Выброшенная группа становится тестовой и на ней модель проверяется. Цикл повторяется столько раз, на сколько групп были разбиты исходные данные. Среднее значение метрики на тестовых данных становится итоговой оценкой качества алгоритма. Существует несколько вариантов реализации этого подхода. Используется в основном, когда данных немного, так как при большом количестве данных или большом количестве групп, кросс-валидация проводится очень долго.

Для решения нашей задачи исходный датасет разделим на тренировочную и проверочную выборки. На тренировочной выборке будем проводить кросс-валидацию. Обучающую способность алгоритмов будем проверять по проверочным данным. Такой подход обеспечивает максимальную независимость итогового значения от процесса обучения, так как проверочные данные совсем не будут участвовать в обучении моделей.

Исходные данные содержат 153 признака, 1 целевой ответ, 169985 примеров.

После разбиения на тренировочную (70%) и проверочную (30%) выборки получаем:

- Тренировочная – 118989 примеров.
- Проверочная – 50996 примеров.

Для кросс-валидации будем использовать разбиение на группы, которое сохранит соотношение классов как в тренировочной, так и тестовой подвыборках (StratifiedKFold). Количество групп выбираем равное 5.

3.4. Ошибки и переобучение

Предсказание конкретного класса у объекта все-таки не идеально и алгоритмы допускают ошибки. В некоторых случаях они достаточно малы, а в других настолько велики, что метод можно не использовать. Общая ошибка алгоритма может быть разложена на смещение, дисперсию и шум (свойство данных).

Смещение (bias) – это средняя ошибка на разных тренировочных наборах.

Дисперсия (variance) – показывает насколько чувствительной бывает

оценка к разным наборам тренировок.

При этом существует определенный компромисс между минимизацией смещения и дисперсии (the bias-variance tradeoff), т.е. уменьшение одного приводит к увеличению другого. Но так как оба показателя важны, не желательно проводить оптимизацию без оглядки на другой.

Переобучение – это когда модель хорошо предсказывает на тренировочной выборке, но сильно ошибается на тестовой, т.е. обладает низкой обобщающей способностью. Об этой проблеме сигнализируют высокая дисперсия и низкое смещение.

Недообучение – это когда модель еще не научилась предсказывать целевое значение. Об этой проблеме сигнализируют высокое смещение и низкая дисперсия.

Смещение и дисперсия используются как при выборе конкретного алгоритма, так и его гиперпараметров. При этом стараются выбирать модели с наименьшими показателями смещения и дисперсии.

Большое количество признаков затрудняет визуальное представление и анализ модели. Поэтому применяются другие инструменты, такие как валидационная кривая и кривая обучения.

Валидационная кривая – это зависимость оценки на тренировочной и тестовой выборках от конкретного значения гиперпараметра.

По этой кривой можно определить влияние гиперпараметра на переобучение или недообучение. Если оценки на обучающей и тестовой выборках низкие, то модель недообучилась. Низкая оценка на тестовой выборке и высокая на тренировочной указывает на переобучение модели. Все остальные варианты считаются хорошими.

Кривая обучения – это зависимость оценки от количества примеров в тренировочной выборке. Также эта кривая показывает, зависит ли оценка от ошибки дисперсии или ошибки смещения.

Если оценка на обучающей и валидационной выборке сходится к меньшему значению при увеличении размера выборки, то добавление новых

примеров не улучшит работу алгоритма. Очень большая оценка на тренировочной выборке и недостаточная на проверочной скорее всего показывает, что увеличение примеров может улучшить предсказания модели.

3.5. План исследования

Решение задачи машинного обучения обычно состоит из нескольких шагов:

- Определение подходящих алгоритмов для решения задачи;
- Выбор базовых параметров для алгоритмов;
- Обучение выбранных алгоритмов;
- Анализ результатов;
- Поиск оптимальных параметров для каждого алгоритма;
- Сравнение лучших результатов;
- Выбор итогового алгоритма.

На практике достаточно трудно определить идеальный алгоритм, только посмотрев на данные. Поэтому зачастую перед исследователем на первом этапе ставится задача по выбору и сравнению нескольких алгоритмов между собой.

Для решения этой задачи сравним алгоритмы, которые чаще всего используют для бинарной классификации, а именно:

- Логистическая регрессия;
- K-ближайших соседей (KNN);
- Случайный лес (Random forest);
- Многослойная нейронная сеть (MLP);
- Сверточная нейронная сеть (CNN).

3.6. Логистическая регрессия

Логистическая регрессия относится к линейным алгоритмам классификации. Она определяет не конкретный класс объекта, а вероятность принадлежности объекта к каждому из классов. Непосредственно вероятность вычисляется для положительного класса. При этом минимизируется

логарифмическая функция потерь, которая не позволяет алгоритму сильно ошибаться в вероятностях.

В библиотеке scikit-learn класс для логистической регрессии называется LogisticRegression (penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1).

Любая модель в машинном обучении имеет свои гиперпараметры, т.е. настройки, которые необходимо подобрать и задать до начала обучения.

Для логистической регрессии такими параметрами будут:

C - инверсный параметр регуляризации. Чем меньше значение, тем сильнее регуляризация. Сильная регуляризация приводит к недообучению, а слишком слабая к переобучению модели,

tol - толерантность для критериев остановки, т.е. поиск максимума или минимума прекратится, когда искомое число будет достаточно близко,

solver - алгоритм, используемый в задаче оптимизации. По умолчанию используется liblinear (метод покоординатного спуска)

Другие параметры оставим такими, как их рекомендуют авторы scikit-learn.

Для первого запуска положим значения tol = 0.3, C = 0.5, solver = ‘liblinear’. В результате обучения модели получаем AUC ROC = 0.5944 (рисунок 17).

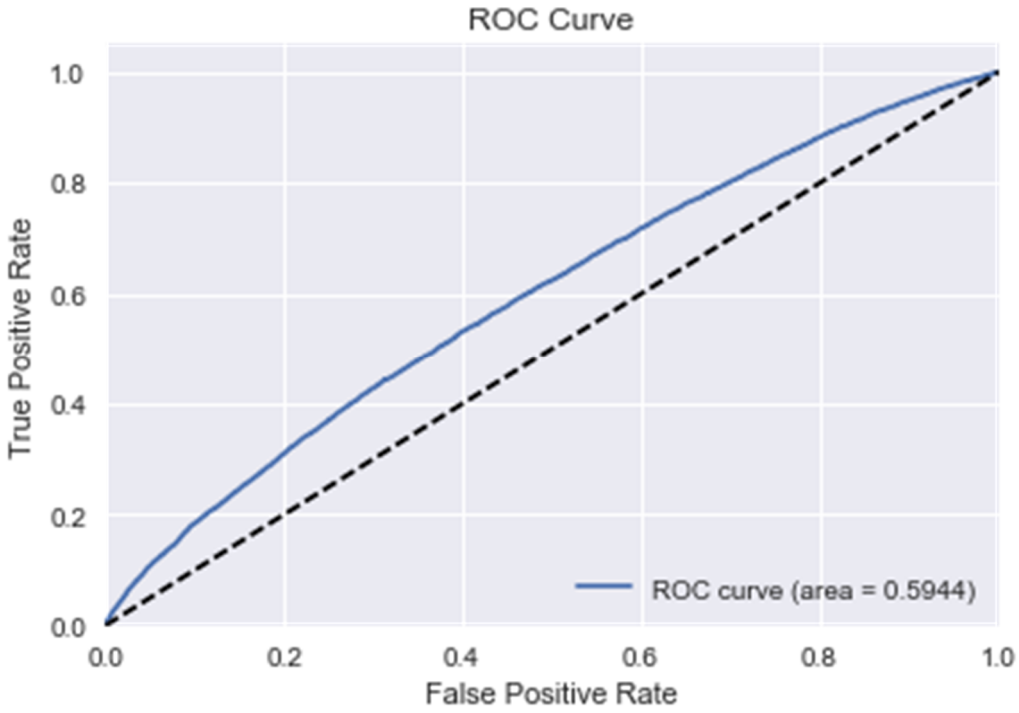


Рисунок 17 – ROC кривая для логистической регрессии (tol = 0.3, C = 0.5, solver = ‘liblinear’)

Возьмем эту площадь за базовое значение и попытаемся его улучшить. Логистическая регрессия считается одним из основных методов при бинарной классификации и подобный неудовлетворительный результат может говорить о неправильно подобранных гиперпараметрах.

Если рассмотреть кривую обучения, то становится понятно, что модель недообучилась (рисунок 18).

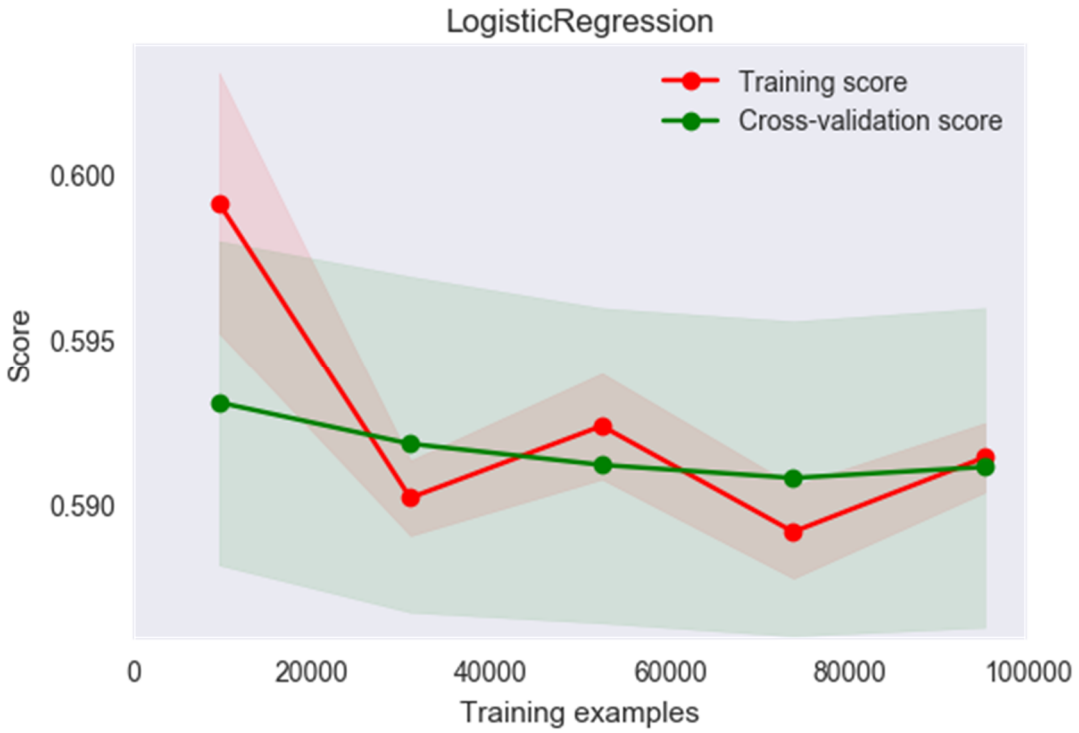


Рисунок 18 – Кривая обучения для логистической регрессии (tol = 0.3, C = 0.5, solver = ‘liblinear’)

С помощью валидационной кривой можно примерно понять, какой параметр нужно изменить, чтобы улучшить качество модели. Из рисунка 19 видно, что уменьшение гиперпараметра tol, может значительно улучшить алгоритм, причем значение должно быть как можно ближе к 0.

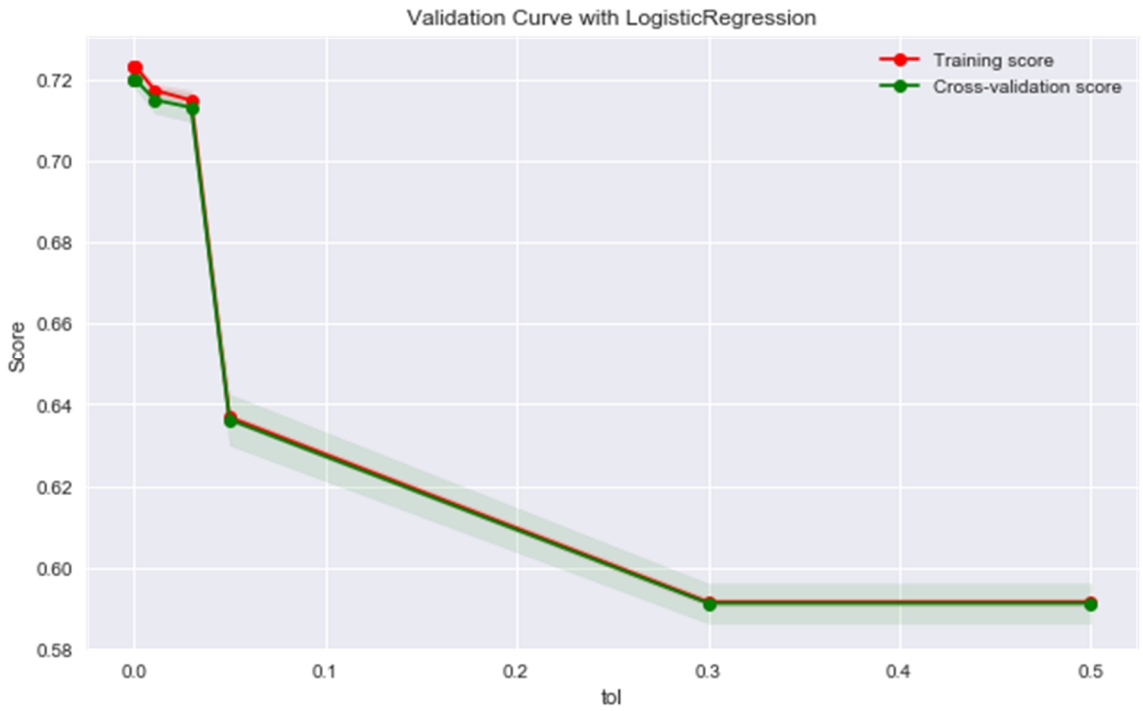


Рисунок 19 – Валидационная кривая логистической регрессии для параметра tol

При этом гиперпараметр C можно и не менять (рисунок 20).

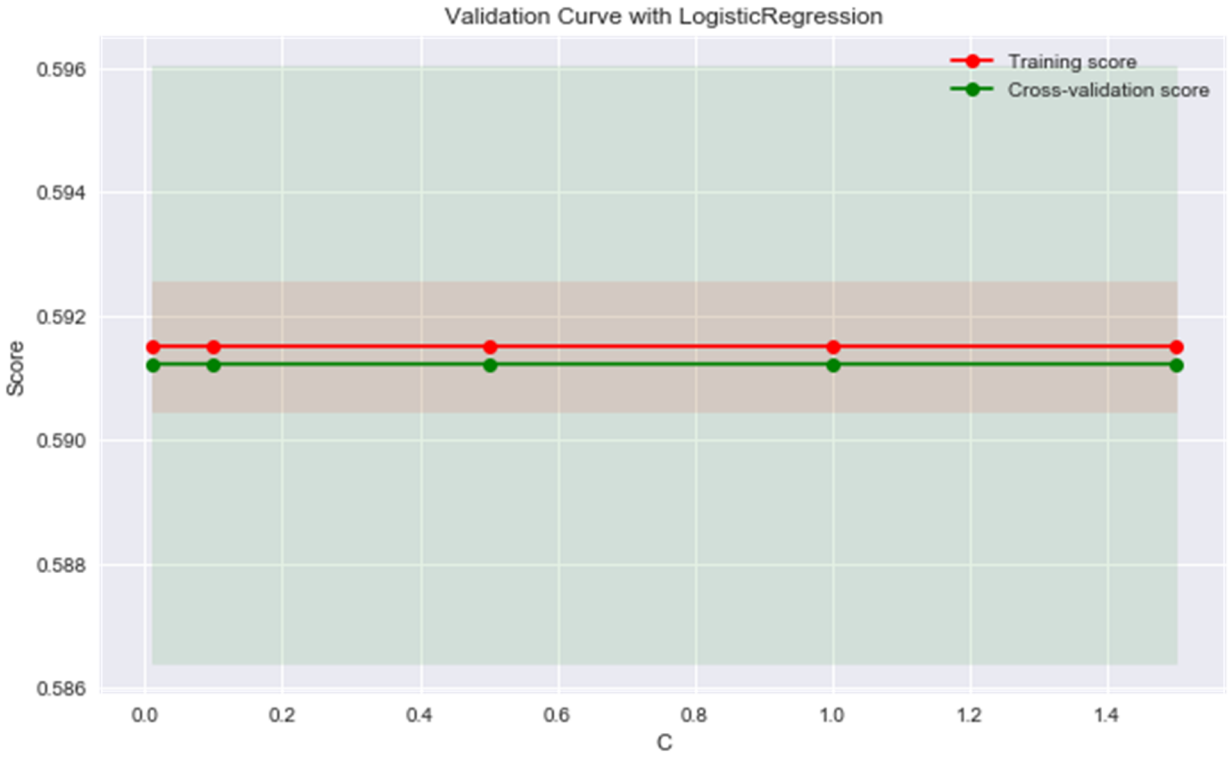


Рисунок 20 – Валидационная кривая логистической регрессии для параметра C

Изменение гиперпараметра tol сильно влияет на модель. Положим tol = 0.00001 и получим AUC ROC = 0.7198 (рисунок 21).

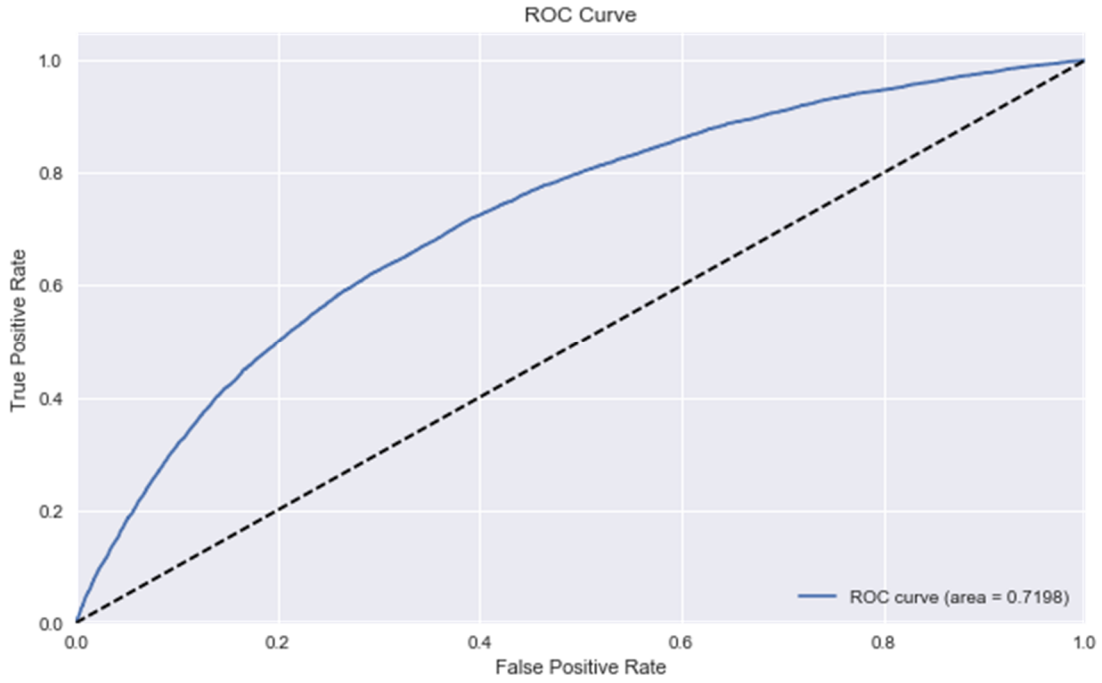


Рисунок 21 – ROC кривая для логистической регрессии (tol = 0.00001, C = 0.5, solver = ‘liblinear’)

Теперь можно приступить к изменению гиперпараметра C. Как видно из

рисунка 22, изменение гиперпараметра C скорее всего не даст каких-либо результатов. Зато видно, что значения меньше 0.4 постепенно ухудшают модель.

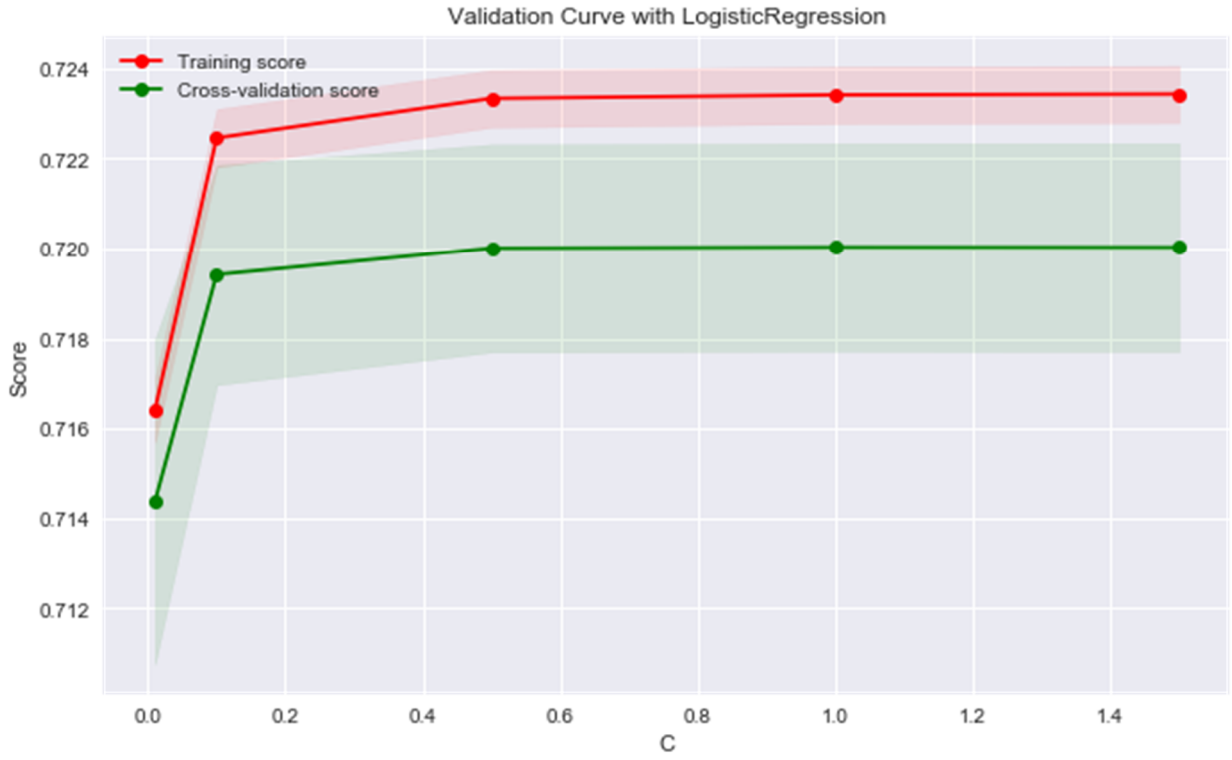


Рисунок 22 – Валидационная кривая логистической регрессии для параметра C (tol=0.00001)

Логистическая регрессия с параметрами C=0.5, tol=0.00001, solver= liblinear на кросс-валидации показывает AUC ROC=0.72, а на отложенной тестовой выборке AUC ROC=0.7198.

Найдем оптимальные гиперпараметры с помощью поиска по сетке. Для логистической регрессии предусмотрен специальный тип поиска LogisticRegressionCV. Вариант со следующими гиперпараметрами оказался лучшим: C=0.5, tol=0.00001, solver= liblinear, т.е. он полностью совпал с ручным вводом.

3.7. K-ближайших соседей (k-nearest neighbors, KNN)

Метод K-ближайших соседей относится к метрическим алгоритмами считается одним из самых простых. Для каждого объекта класс определяется относительно его соседей (других ближайших к нему объектов). Если k соседей

относятся к определенному классу, то и у текущего объекта будет этот класс.

Количество ближайших соседей, с которыми нужно сравнивать объект, и метрика расстояний между объектами являются гиперпараметрами и задаются до начала обучения модели.

При использовании этого алгоритма стоит помнить, что он не очень устойчив к зашумленным данным, и работает медленней при большом объеме данных. Также одной из проблем является подбор метрики расстояний под каждую задачу.

Для повышения качества алгоритма желательно удалить шумы из выборки, а еще лучше оставить только эталонные примеры.

Количество ближайших соседей обычно подбирается по валидационной кривой или автоматически по кросс-валидации.

В библиотеке scikit-learn класс для K-ближайших соседей называется KNeighborsClassifier (n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=1, **kwargs).

Для первого запуска положим значения n_neighbors = 3, metric='minkowski'. В результате обучения модели получаем AUC ROC = 0. 6162 (рисунок 23).

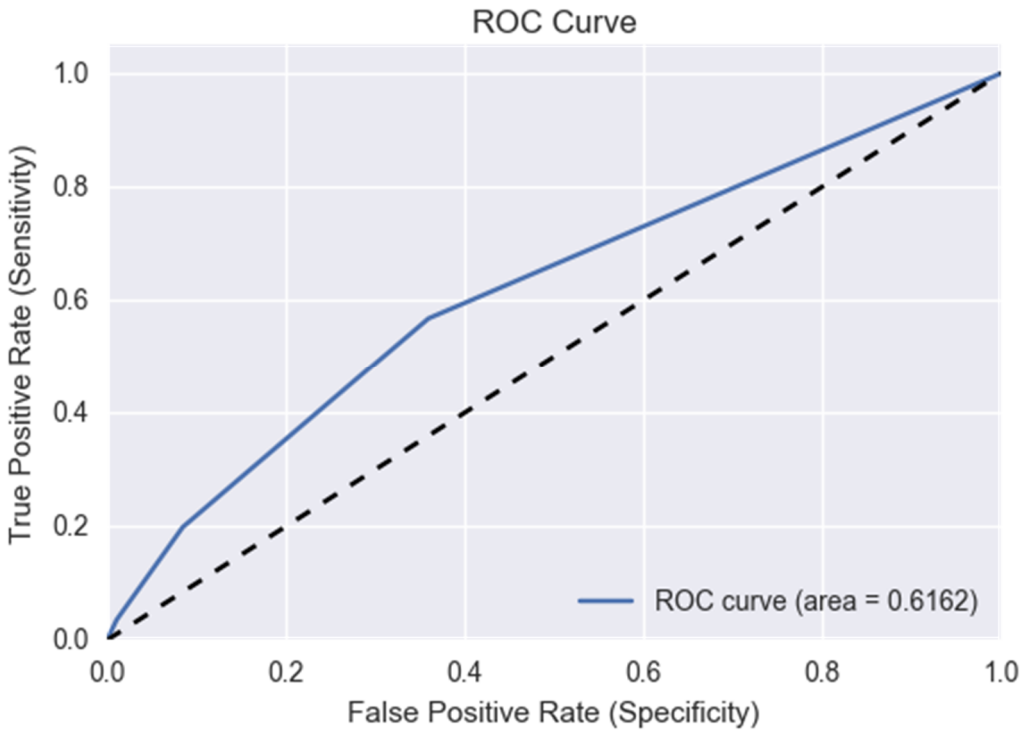


Рисунок 23 – ROC кривая для k-nearest neighbors (n_neighbors = 3, metric='minkowski')

Судя по кривой обучения модель недообучена (рисунок 24).

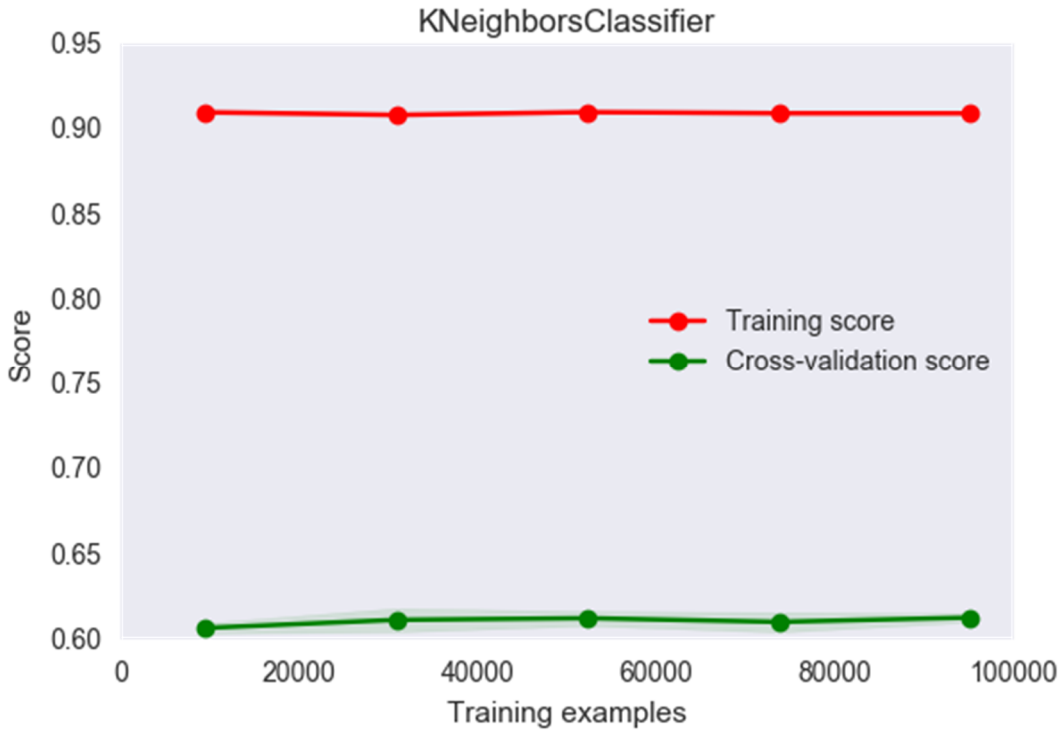


Рисунок 24 – Кривая обучения для k-nearest neighbors (n_neighbors = 3, metric='minkowski')

Увеличение количества соседей может улучшить предсказательные способности алгоритма (рисунок 25).

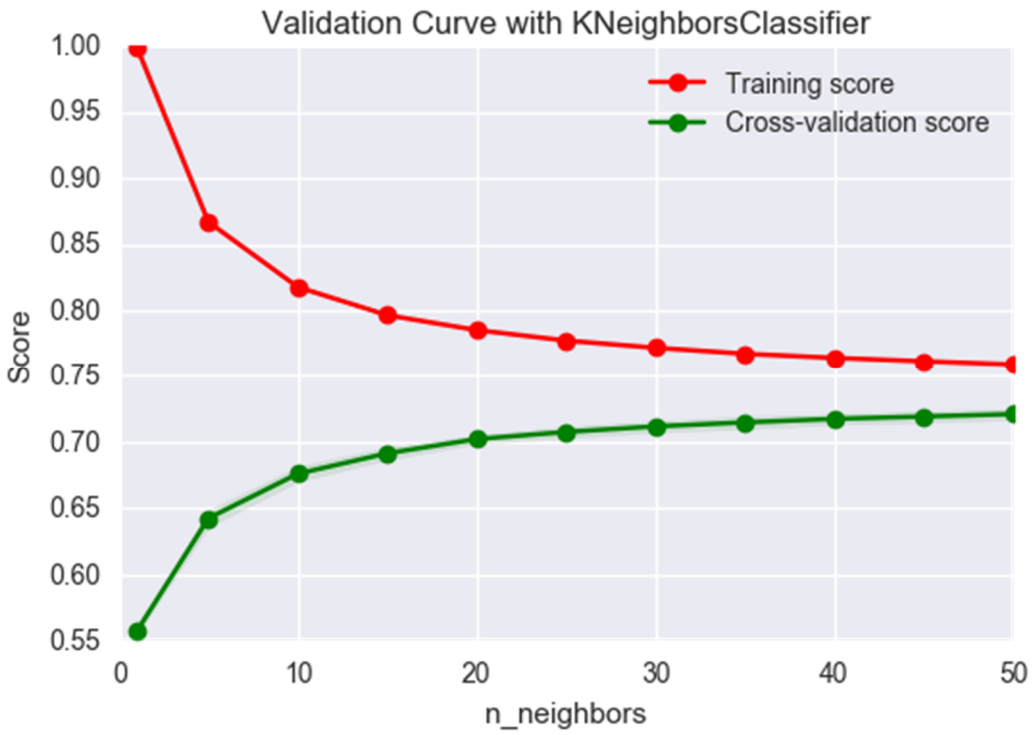


Рисунок 25 – Валидационная кривая k-nearest neighbors для параметра n_neighbors

Увеличение n_neighbors до 60, приводит к AUC ROC = 0.7249 на отложенной выборке.

По графикам видно, что текущие настройки далеки от совершенства, но из-за большого количества примеров и размерности выборки алгоритм очень долго производит расчеты. На будущее, для улучшения качества можно было бы попробовать другие метрики расстояния и провести отбор эталонных примеров.

3.8 Случайный лес (RandomForest)

Случайный лес основан на бэггинге над решающими деревьями. Сами по себе решающие деревья имеют низкое смещение (могут достигать почти 0 ошибки), но легко переобучаются. Бэггинг же позволяет совмещать подобные алгоритмы в несмещенную композицию с низкой дисперсией, что дает очень хорошие результаты.

В библиотеке scikit-learn класс для случайного леса называется RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_split=1e-07, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None).

Гиперпараметры для случайного леса:

- n_estimators – количество деревьев. Обычно, чем больше деревьев, тем лучше, но увеличивается длительность обучения,
- max_features - число признаков для выбора расщепления. Валидационная кривая на тестовой выборке должна быть унимодальна, на тренировочной строго возрастает. При увеличении этого параметра увеличивается время обучения. Настраивается после количества деревьев.
- max_depth – Максимальная глубина деревьев. Увеличение

глубины деревьев приводит к увеличению качества предсказания на контроле. Обычно используют максимальную глубину, неглубокие деревья применяются в задачах с большим количеством шумов.

- min_samples_split - Минимальное число объектов, при котором выполняется расщепление. При увеличении параметра качество на обучении падает, а время построения RF сокращается. Не оказывает сильного влияния, если используются неглубокие деревья.
- min_samples_leaf - Ограничение на число объектов в листьях. Не оказывает сильного влияния, если используются неглубокие деревья.

Известно, что качество алгоритма случайного леса очень зависит от выбранного количества деревьев. Чем их больше, тем лучше. Но большое количество замедляет работу алгоритма. Для выбора наиболее удачного значения удобно использовать валидационную кривую.

Из рисунка 26 видно, что после 200 рост качества замедляется, поэтому зафиксируем именно это значение для дальнейших расчетов.

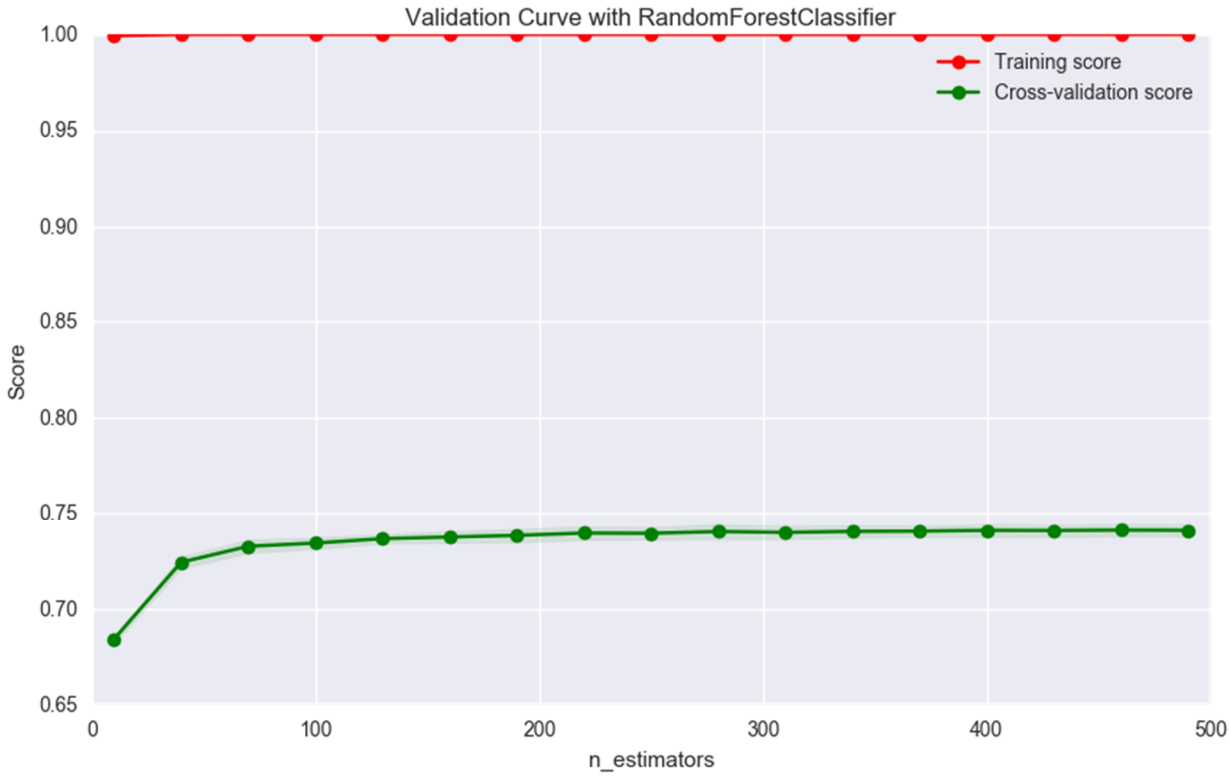


Рисунок 26 – Валидационная кривая RandomForest для параметра n_estimators

Для n_estimators=200 и остальными параметрами по умолчанию AUC

ROC=0.7391. На данный момент — это лучший результат, но можно попробовать его еще улучшить.

Следующий параметр, который следует выбрать – это количество признаков для расщепления. По умолчанию советуют устанавливать значение, равное квадратному корню от общего количества признаков. В нашем случае будет примерно 12 признаков.

Из рисунка 27 видно, что пик приходится на значения от 15 до 25 признаков. Скорее всего, в этом отрезке и находится оптимальное значение. Установим max_features=15, так большее количество признаков замедляет работу алгоритма.

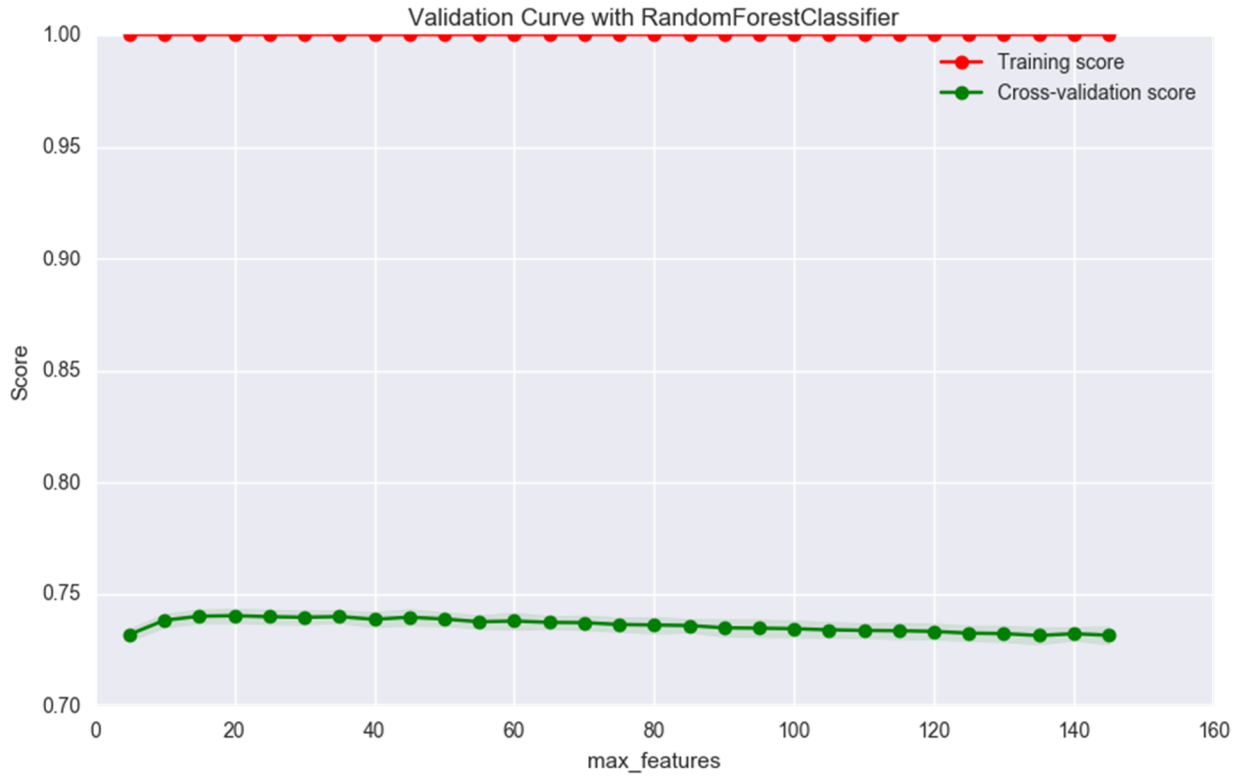


Рисунок 27– Валидационная кривая RandomForest для параметра max_features

Хотя рекомендуется не ограничивать глубину дерева, все-таки попытаемся найти оптимальное значение. Из рисунка 28 видно, что лучшее значение глубины приходится на 20. Значит установим max_depth=20.

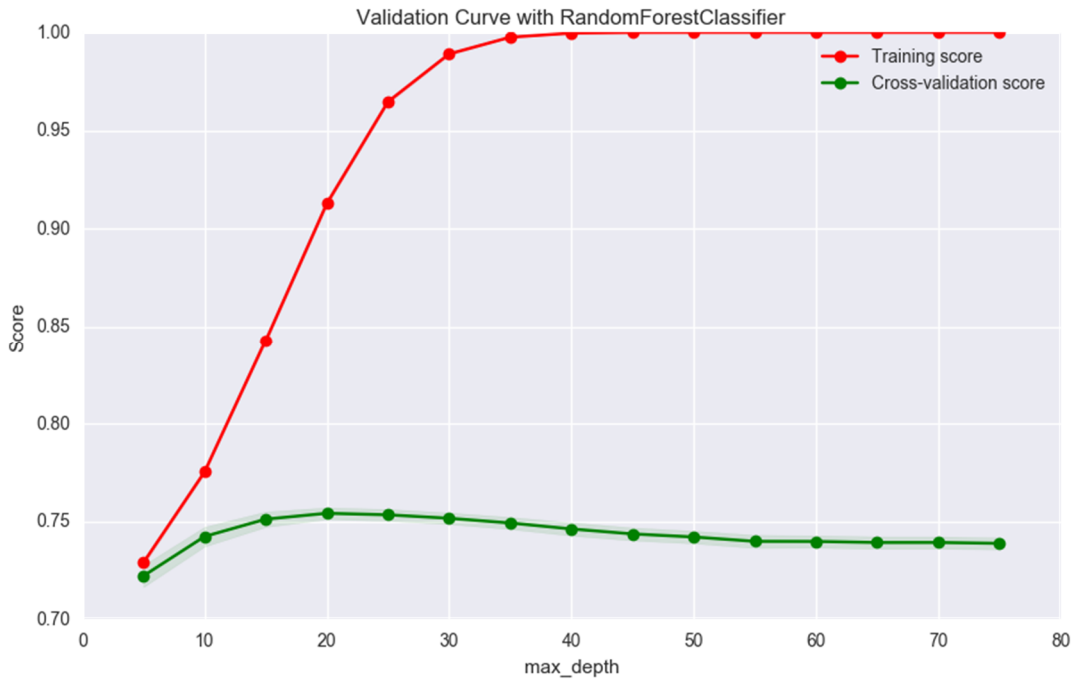


Рисунок 28 – Валидационная кривая RandomForest для параметра max_depth

Другие параметры из-за ограничения глубины можно не трогать. Итак, случайный лес с гиперпараметрами n_estimators=200, max_features=15, max_depth=20 демонстрирует AUC ROC= 0.7588, что видно из рисунка 29.

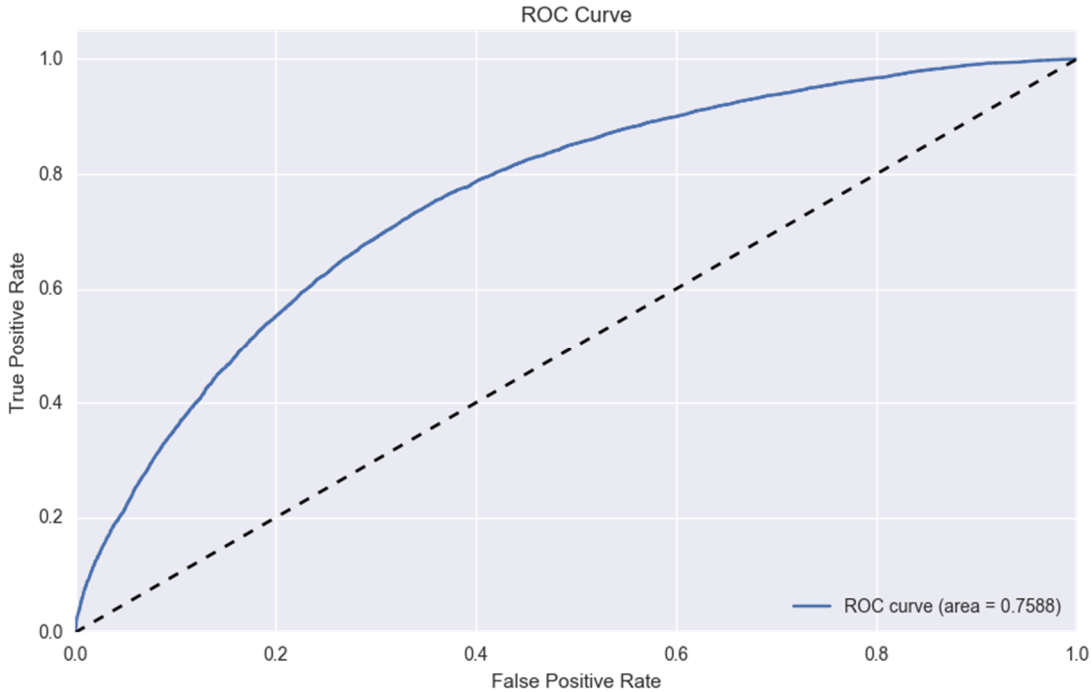


Рисунок 29 – ROC кривая для RandomForest (n_estimators=200, max_features=15, max_depth=20)

Рисунок 30 показывает кривую обучения для выбранных параметров. Из него видно, что немного уменьшилось переобучение, но есть еще над чем

работать.

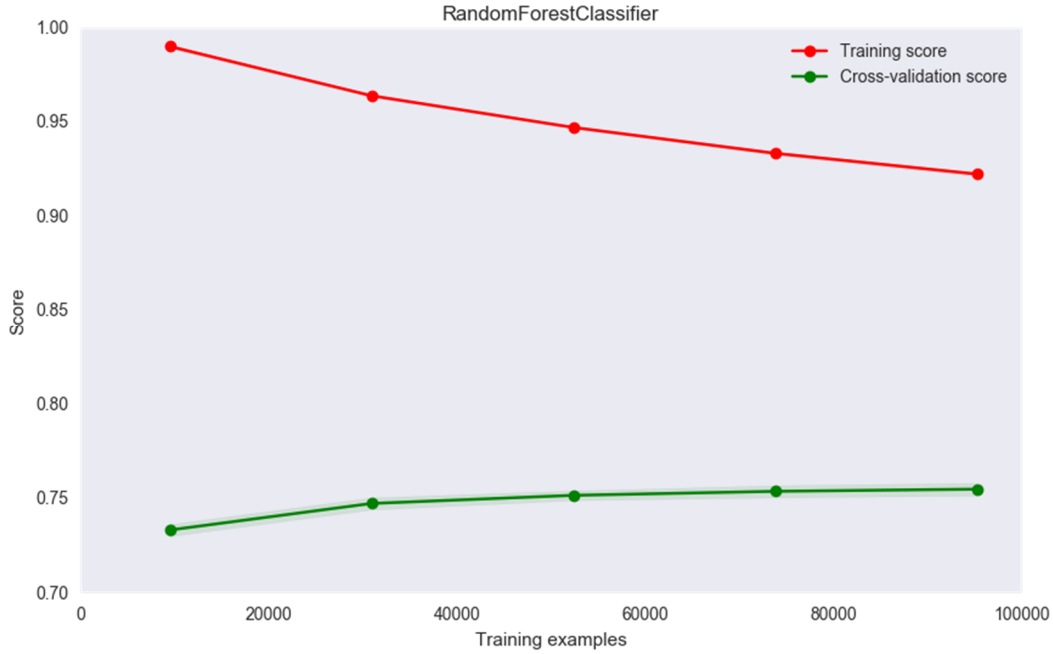


Рисунок 30 – Кривая обучения для RandomForest (n_estimators=200, max_features=15, max_depth=20)

Дальнейший подбор параметров в ручном режиме может занять слишком много времени. Поэтому запустим перебор параметров по сетке, установив границы каждого параметра, исходя из валидационных кривых, приведенных выше.

Самый лучший результат, а именно AUC ROC= 0.7594, дают следующие параметры: max_features=27, max_depth=18, n_estimators=250.

3.9. Многослойная нейронная сеть (MLP)

Нейронная сеть считается универсальным апроксиматором. Однако однослойная нейронная сеть имеет ряд ограничений, которые пропадают при увеличении слоев сети. Все слои в такой сети являются полносвязными.

Для классификации обычно используют многослойный персептрон с 3 – 4 скрытыми слоями. Считается, что именно такая архитектура сети помогает наиболее успешно решать задачи. На практике универсального ответа пока не существует, т.е. количество слоев и количество нейронов подбирают экспериментально к каждой задаче. Это же относится и к функциям активации,

они также подбираются для каждого слоя экспериментально.

Нейронные сети достаточно быстро переобучаются, поэтому очень желательно применять различные методы для снижения этого эффекта. Побочным действием становится увеличение длительности обучения и минимальное количество эпох.

Для построения многослойного персептрона будем использовать theano и keras, которые облегчают работу с нейронными сетями. Еще одним значительным плюсом этих библиотек является расчет с использованием CUDA, что значительно повышает скорость расчетов.

В первую очередь нужно определиться с топологией нейронной сети, т.е. выбрать оптимальное количество слоев и нейронов в них.

Чтобы понять, как нейронная сеть ведет себя в процессе обучения, проведем несколько экспериментов. Зафиксируем количество эпох nb_epoch=60 и количество объектов в пачке batch_size=838, а также количество нейронов в каждом слое будет равно 77 (в 2 раза меньше, чем количество признаков).

На рисунке 31 показан график зависимости AUC ROC от количества эпох для персептрона с одним скрытым слоем и 77 нейронами (в 2 раза меньше, чем количество признаков). Почти сразу после начала обучения предсказание тренировочной и валидационной выборки расходятся, что свидетельствует о стремительном переобучении.

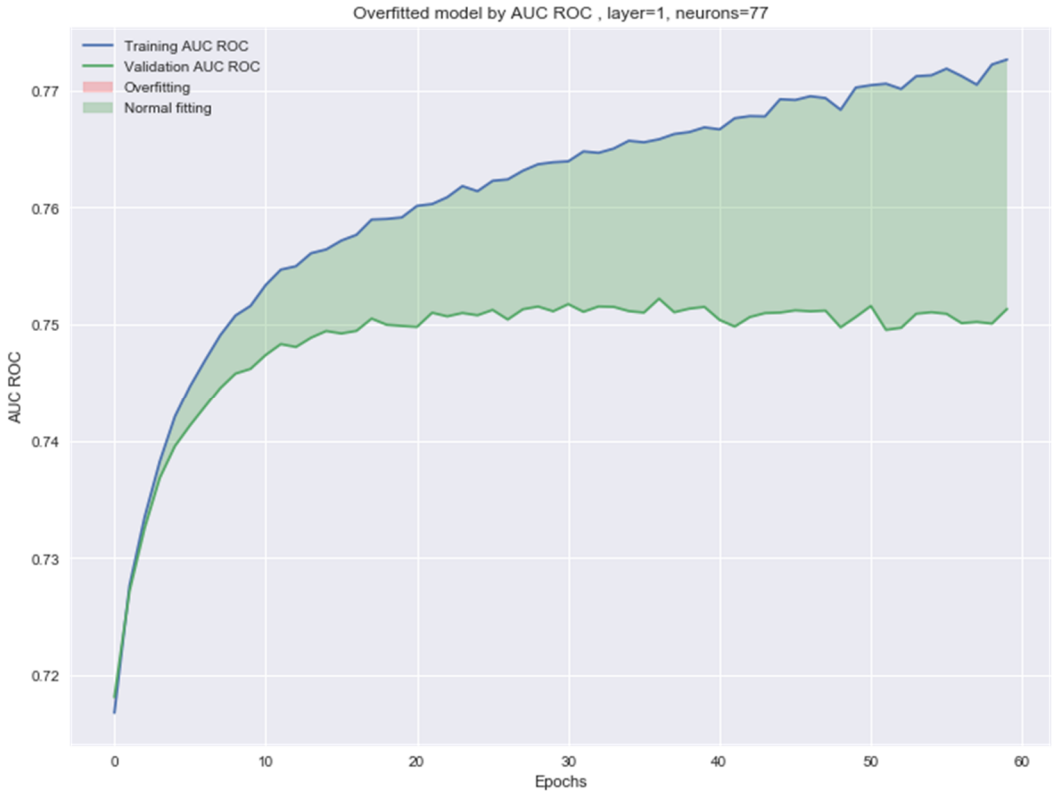


Рисунок 31 – Зависимость AUC ROC от количества эпох (1 слой, 77 нейронов)

Однако даже с учетом переобучения AUC ROC=0.7513 (рисунок 32).

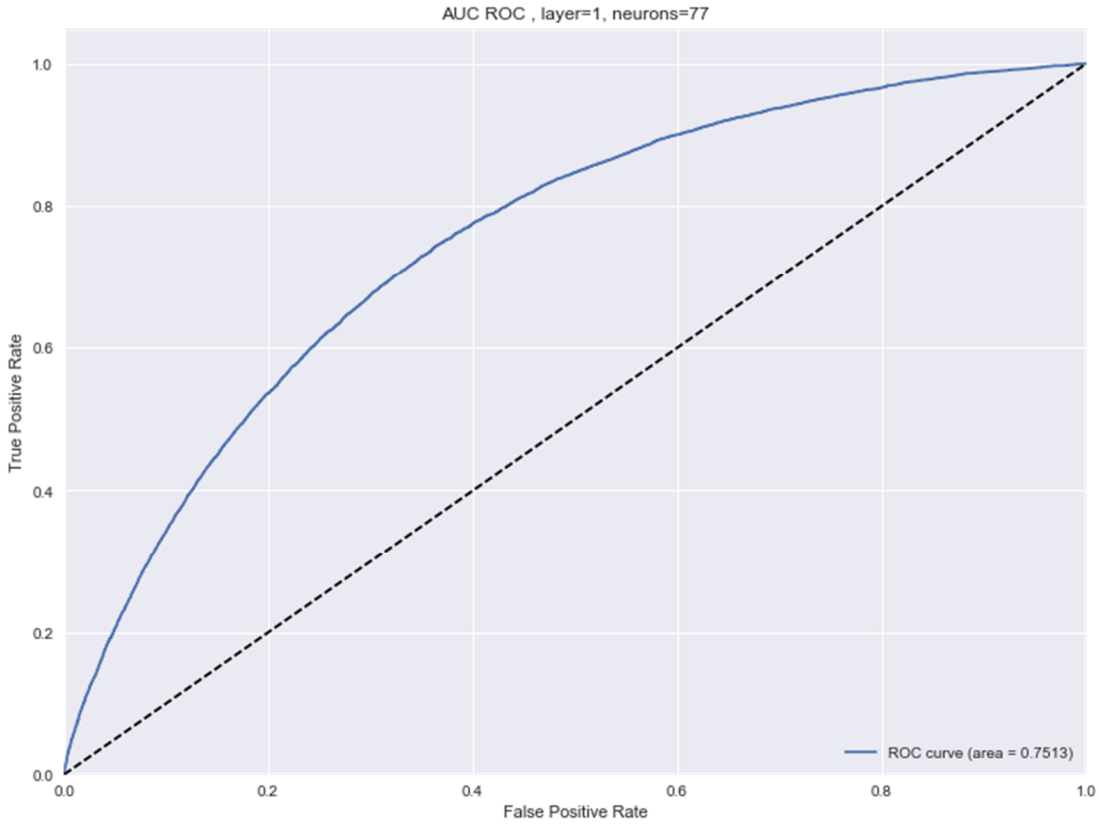


Рисунок 32 – ROC кривая для MLP (1 слой, 77 нейронов)

Нейронная сеть с двумя скрытыми слоями демонстрирует еще более

быстрое переобучение, как видно из рисунка. Что негативно сказалось на её качестве, а именно AUC ROC=0.7453. Если бы остановили обучение модели на 18 эпохе, можно было бы ожидать показатель AUC ROC в районе 0.7533 (рисунок 33).

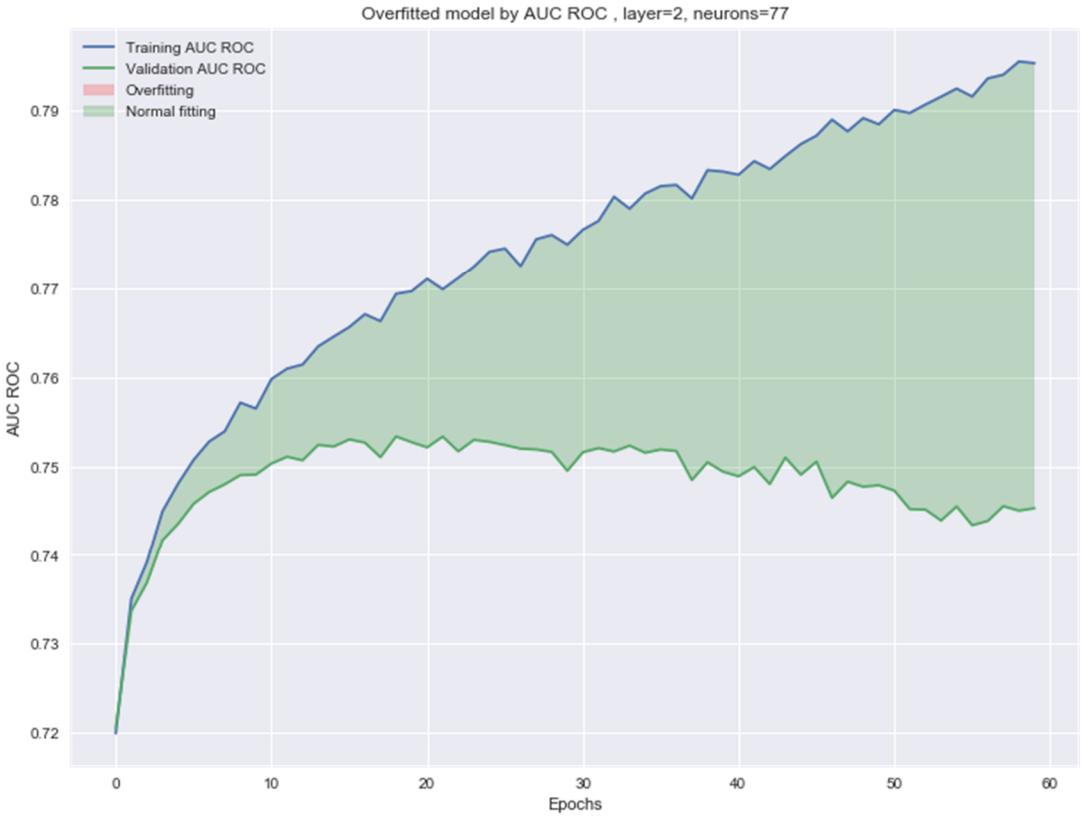


Рисунок 33 – Зависимость AUC ROC от количества эпох (2 слоя, 77 нейронов)

Нейронная сеть с тремя скрытыми слоями показала AUC ROC=0.7291 (рисунок 34).

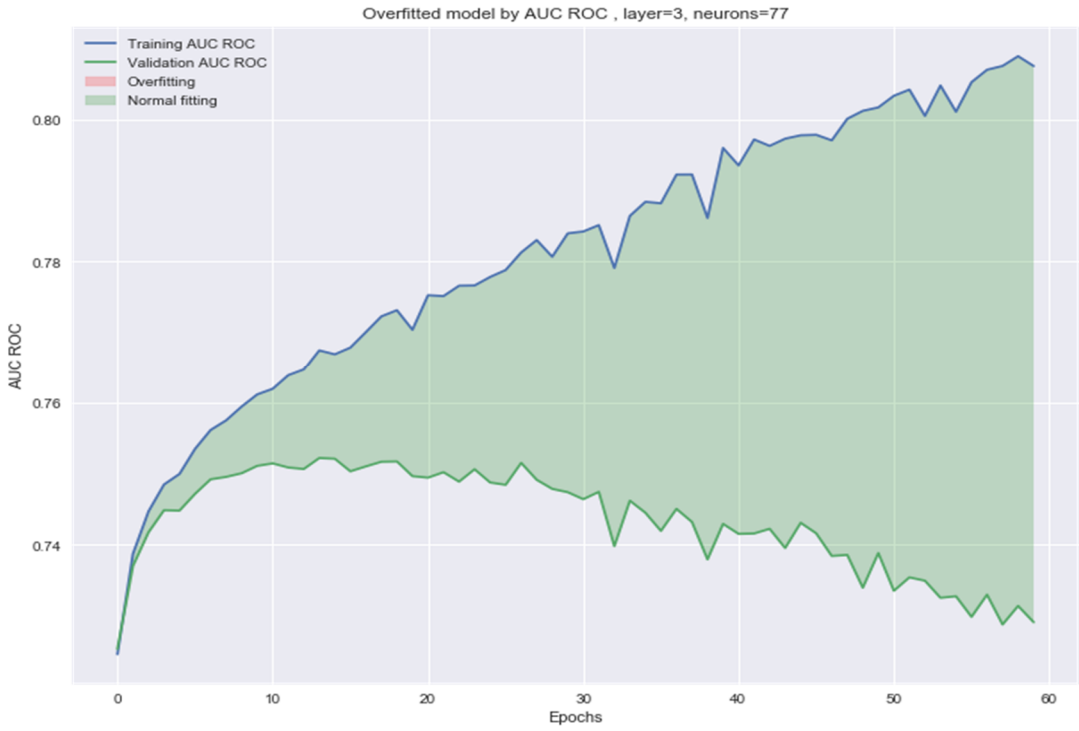


Рисунок 34 – Зависимость AUC ROC от количества эпох (3 слоя, 77 нейронов)

Четырехслойная нейронная сеть показала худший результат, а именно AUC ROC=0.7267 (рисунок 35).

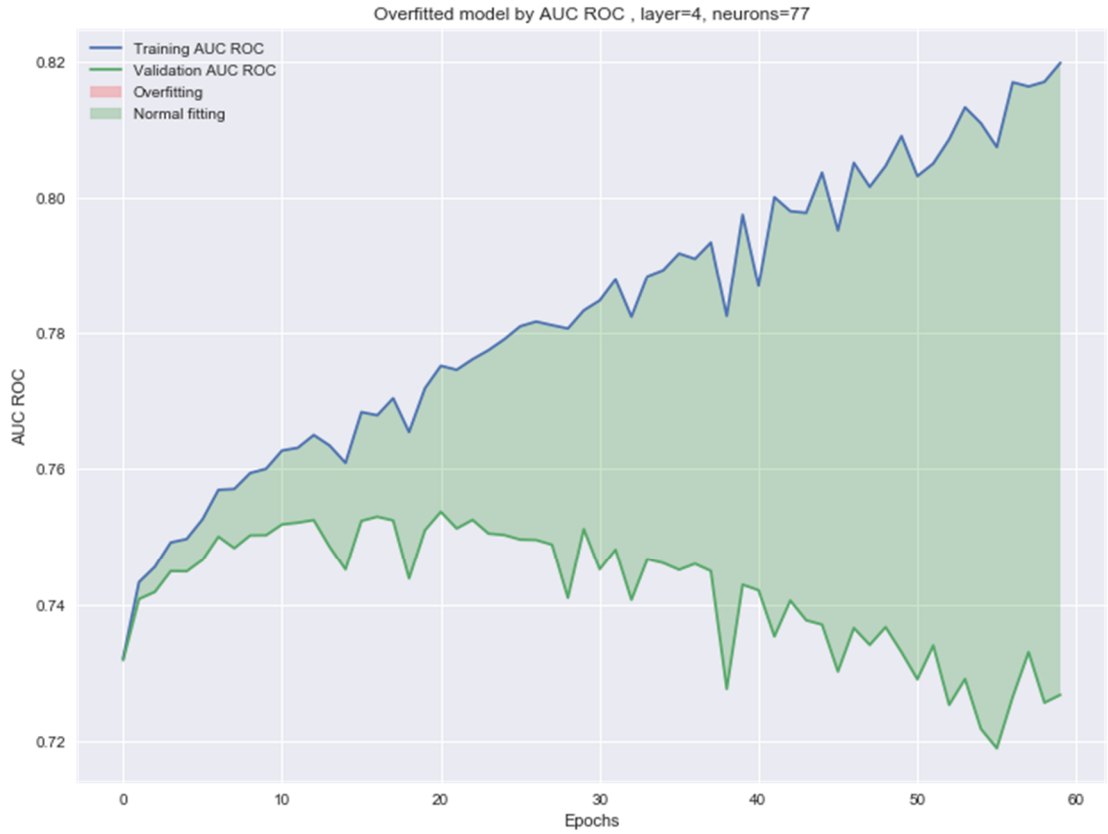


Рисунок 35 – Зависимость AUC ROC от количества эпох (2 слоя, 77 нейронов)

Из проведенных исследований становится понятно, что увеличение

скрытых слоев не только не улучшает качество модели, но и заметно её ухудшает. Максимальное значение на валидационной выборке наступает до 20 эпохи, а все последующие эпохи идут только на улучшение предсказания на тренировочной выборке. Сводные результаты экспериментов приводятся в таблице 20.

Таблица 20 – Результаты экспериментов многослойной нейронной сети без дополнительных настроек

Скрытые слои, шт	Валидационный AUC ROC
1	0.7513
2	0.7453
3	0.7291
4	0.7267

Переобучение – это одна из ключевых проблем нейронных сетей. Поэтому для её уменьшения были разработаны определенные методы, которые поддерживаются библиотекой keras. А именно:

- Dropout. На каждой итерации с вероятностью p отбрасывает нейроны в определенном слое. Дальше обучение проходит без этих нейронов до конца итерации. Это позволяет лучше обрабатывать ошибки.
- Регуляризация. Так называемое сокращение весов. Может применяться ко всем слоям или только определенным. Обычно используется L2-регуляризация с параметром C=0.0001.
- Выбор начальных весов. Для линейной функции активации и сигмоида хорошо подходит метод инициализации Xavier (или Glorot). Для ReLU используется метод инициализации He.

Применим все вышеперечисленные методы к моделям. Т.е. к каждому слою применим L2-регуляризацию и dropout=0.2, к выходному слою init='glorot_uniform', к остальным слоям init='he_uniform'. Повторим все эксперименты с нейронными сетями от 1 до 4 скрытых слоев. Количество нейронов в каждом слое по-прежнему 77.

Как видно из рисунка 36, однослойная нейронная сеть стала гораздо

медленнее переобучаться. Разница между тренировочной и валидационной выборкой сократилась.

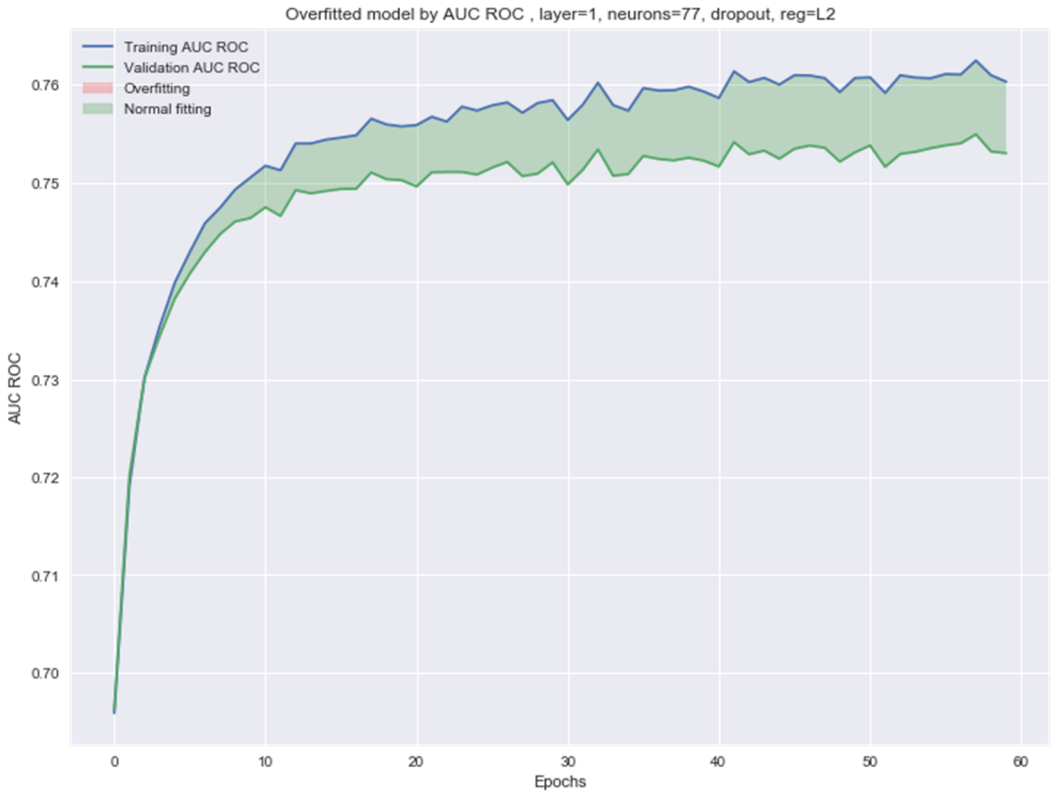


Рисунок 36 – Зависимость AUC ROC от количества эпох (1 слой, 77 нейронов, L2, dropout=0.2)

Качество модели стало лучше, а именно AUC ROC=0.7530 (рисунок 37).

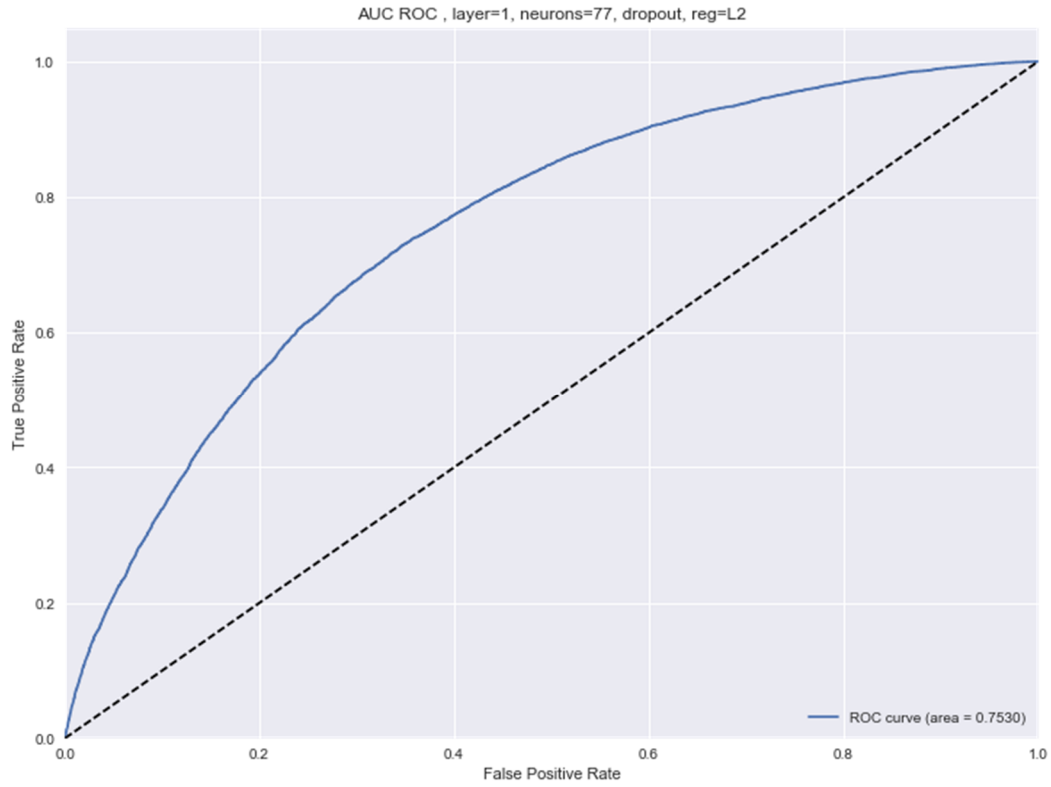


Рисунок 37 – ROC кривая для MLP (1 слой, 77 нейронов, L2, dropout=0.2)

Нейронная сеть с двумя скрытыми слоями показывает AUC ROC = 0.7562. Из рисунка 38 видно, что переобучение также находится под контролем, хотя оно немного больше, чем в однослойной сети.

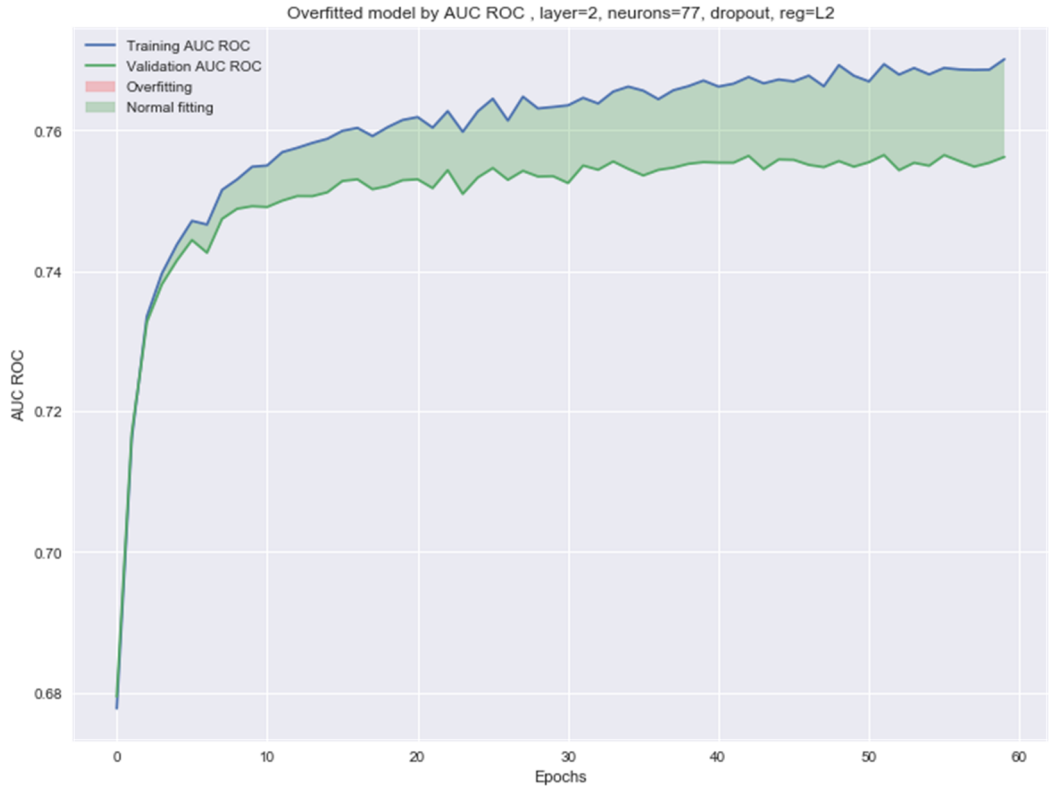


Рисунок 38 – Зависимость AUC ROC от количества эпох (2 слоя, 77 нейронов, L2, dropout=0.2)

Нейронная сеть с тремя слоями имеет ярко выраженное переобучение, как видно из рисунка 39. Качество AUC ROC=0.7491.

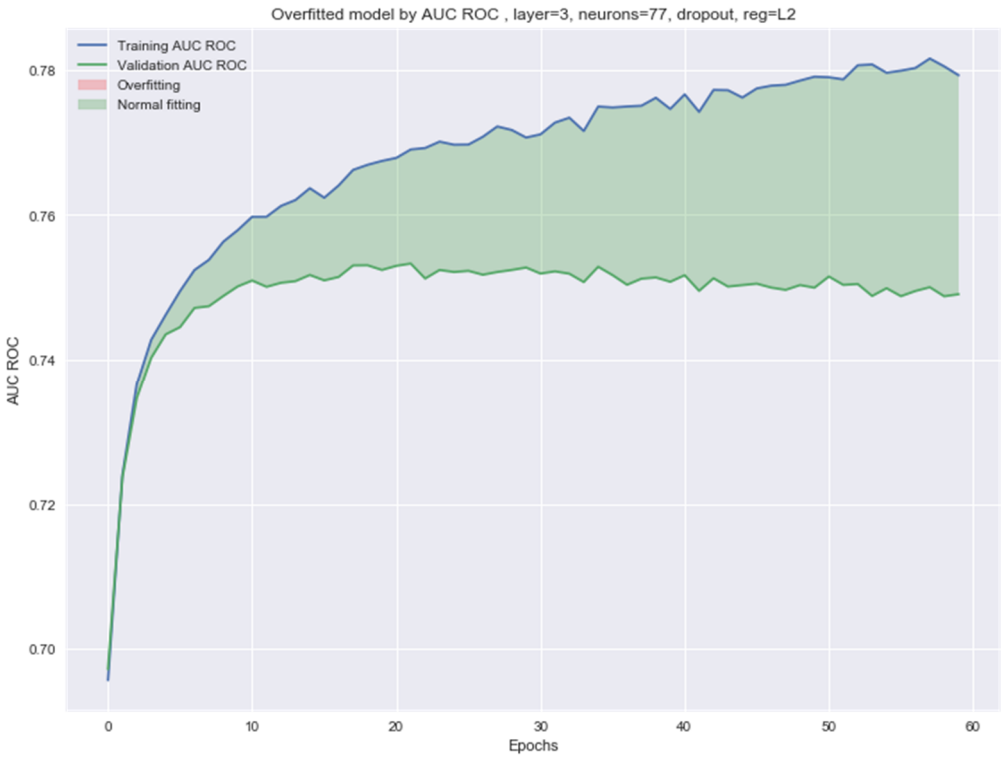


Рисунок 39 – Зависимость AUC ROC от количества эпох (3 слоя, 77 нейронов, L2, dropout=0.2)

Четыре скрытых слоя дают результат AUC ROC=0.7540, процесс обучения представлен на рисунке 40.

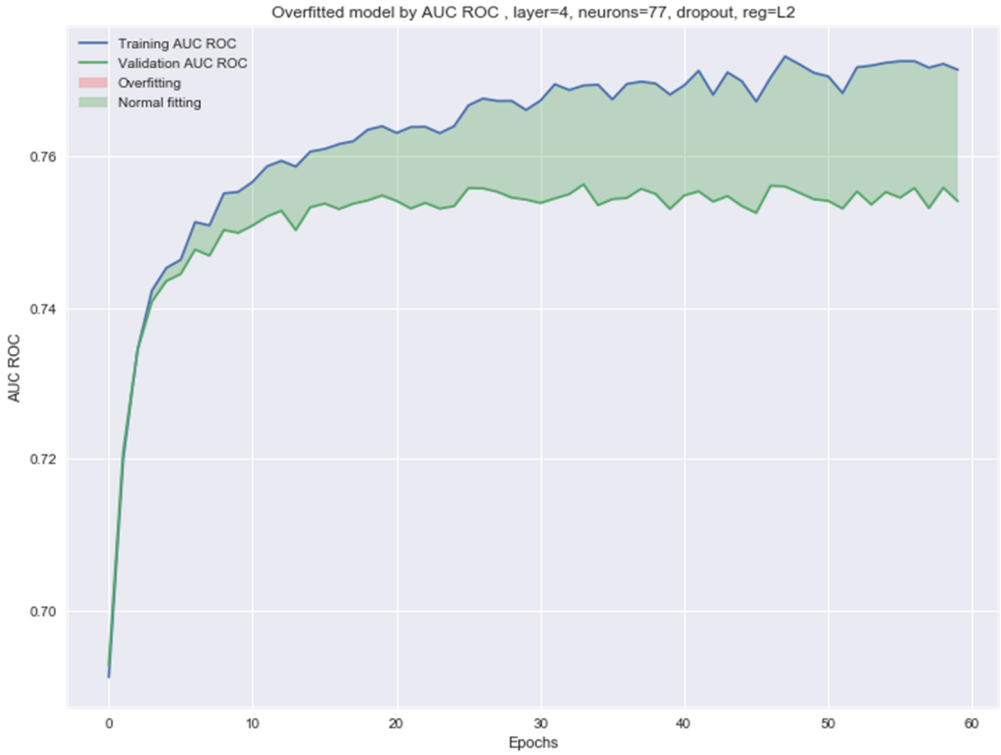


Рисунок 40 – Зависимость AUC ROC от количества эпох (4 слоя, 77 нейронов, L2, dropout=0.2)

Результат этих экспериментов не столь очевиден, как предыдущих. Самый лучший результат показала нейронная сеть с двумя скрытыми слоями, затем идет сеть с четырьмя скрытыми слоями (таблица 21).

Таблица 21 – Результаты экспериментов многослойной нейронной сети с dropout и регуляризацией

Скрытые слои, шт	Валидационный AUC ROC
1	0.7530
2	0.7562
3	0.7491
4	0.7540

Попробуем провести поиск по сетке оптимальных гиперпараметров для нейронной сети с тремя скрытыми слоями. В итоге получаем следующие параметры:

- Первый слой. Нейроны – 300, dropout - 0.2.
- Второй слой. Нейроны – 150, dropout - 0.25.
- Третий слой. Нейроны – 150, dropout - 0.2.

С такими параметрами AUC ROC=0.7573 (рисунок 41).

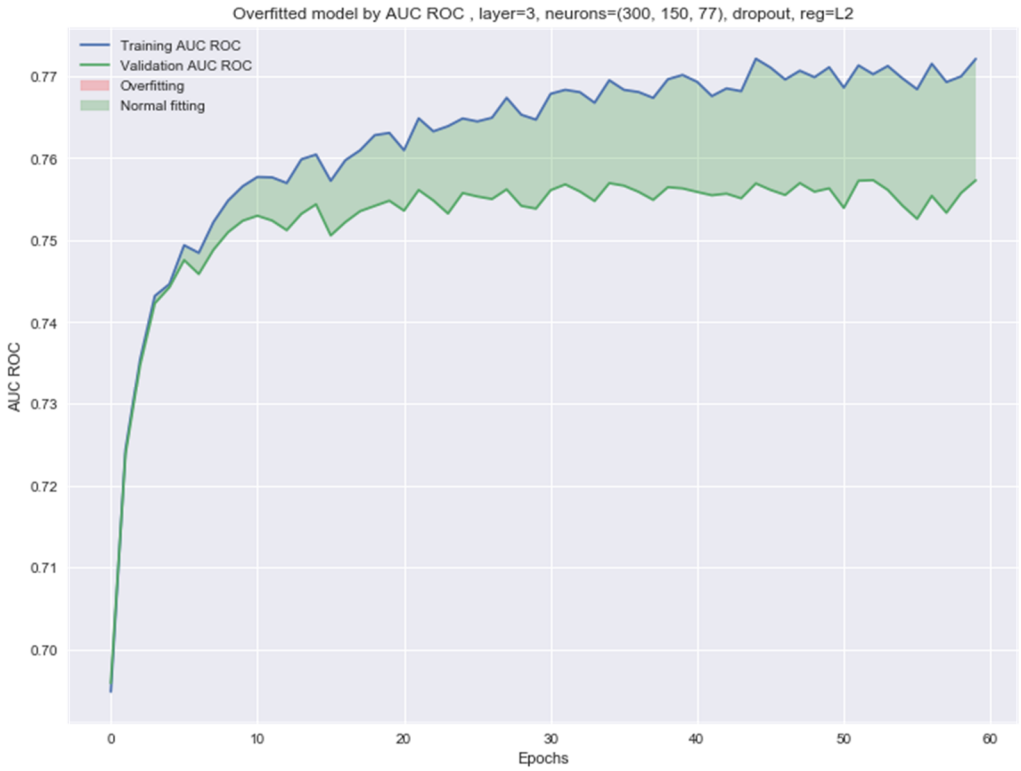


Рисунок 41 – Зависимость AUC ROC от количества эпох (3 слоя: 300, 150, 77 нейронов; L2; dropout)

3.10. Сверточные нейронные сети (CNN)

В отличие от персептрона, в котором есть только полносвязные слои, в сверточной сети несколько типов слоев. А именно:

- Сверточный слой;
- Активационный слой (ReLU);
- Пулинг (слой субдискретизации);
- Полносвязный слой.

Обычно сверточные нейронные сети используют для работы с изображениями, хотя в теории они могут подойти и для других данных.

Попробуем применить их для решения нашей задачи, т.е. для бинарной классификации. Возьмем топологию сети, которую предлагают в keras для этой цели: Conv -> ReLU -> MaxPooling -> Conv -> ReLU -> GlobalAveragePooling -> Dropout -> Dense. Размерность слоя 1D.

Как видно из рисунка 42 сверточная сеть обучается крайне медленно. В некоторых местах качество на валидационной выборке выше, чем на тренировочной.

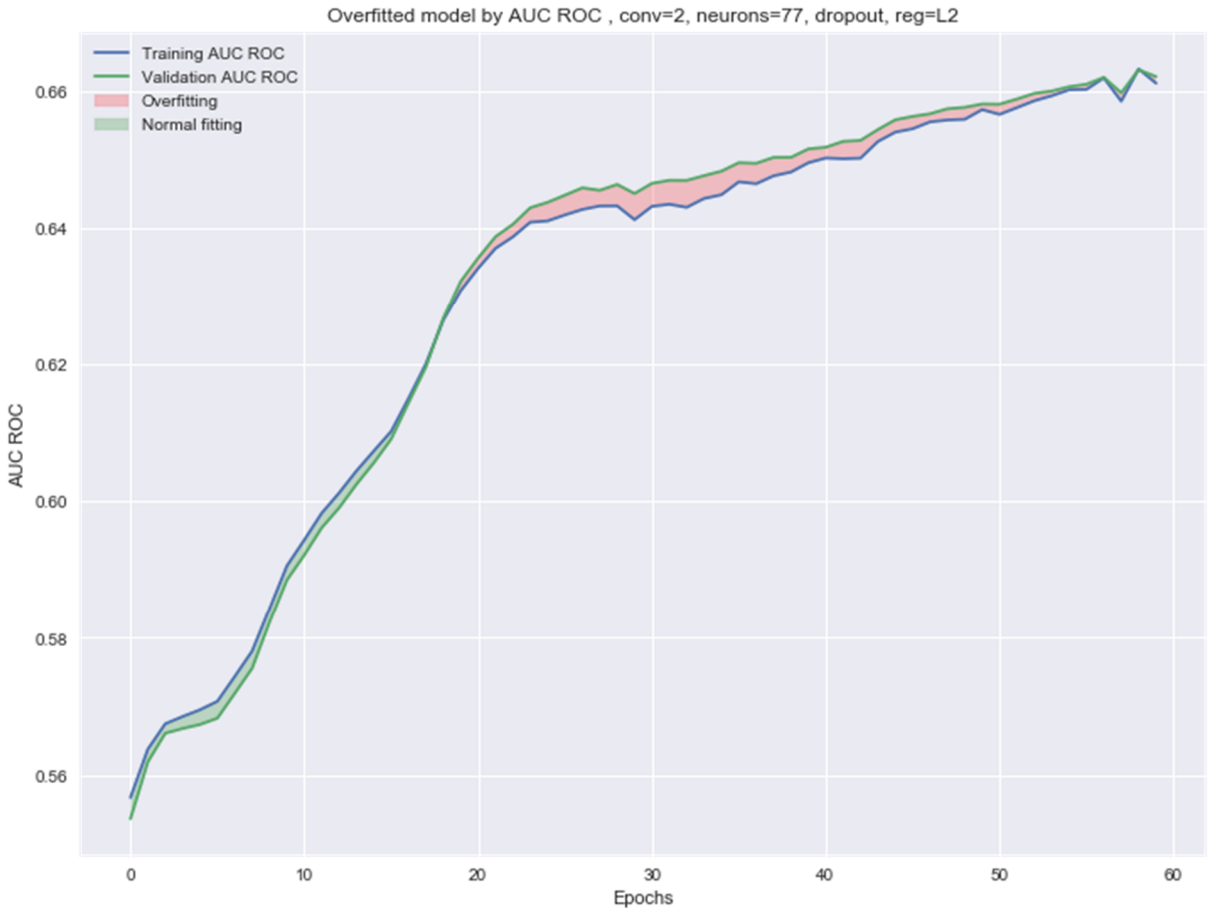


Рисунок 42 – Зависимость AUC ROC от количества эпох (2 скрытых слоя, 77 нейронов; L2; dropout)

Метрика AUC ROC=0.6620 (рисунок 43). Эта нейросеть не успела обучиться за 60 эпох, хотя при их увеличении, возможно, качество будет лучше.

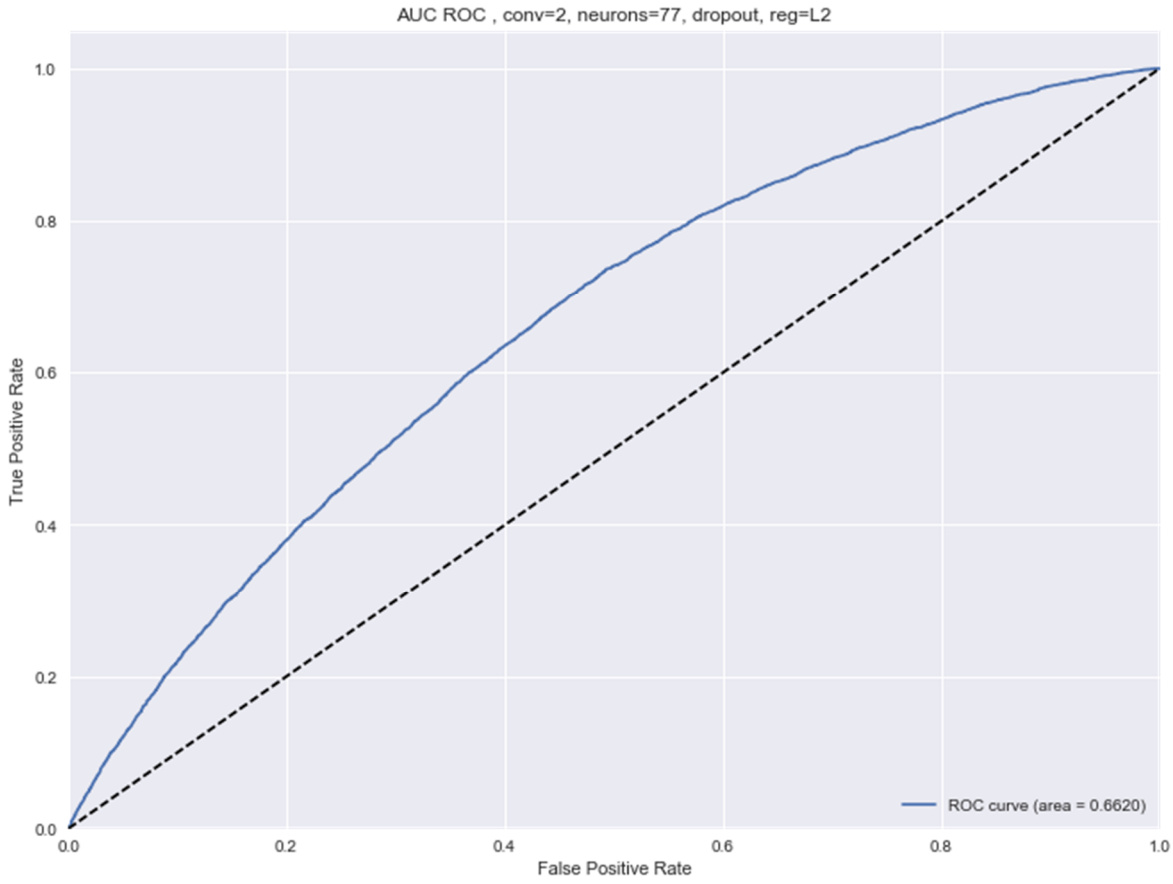


Рисунок 43 – ROC кривая для CNN (2 скрытых слоя, 77 нейронов, L2, dropout=0.2)

Попробуем топологию со сдвоенными сверточными слоями, а именно: Conv -> ReLU-> Conv -> ReLU -> MaxPooling -> Conv -> ReLU-> Conv -> ReLU -> GlobalAveragePooling -> Dropout -> Dense.

Эта сеть обучилась быстрее, чем предыдущая (рисунок 44). Предположительно, увеличение эпох положительно скажется на метрике. AUC ROC=0.7295.

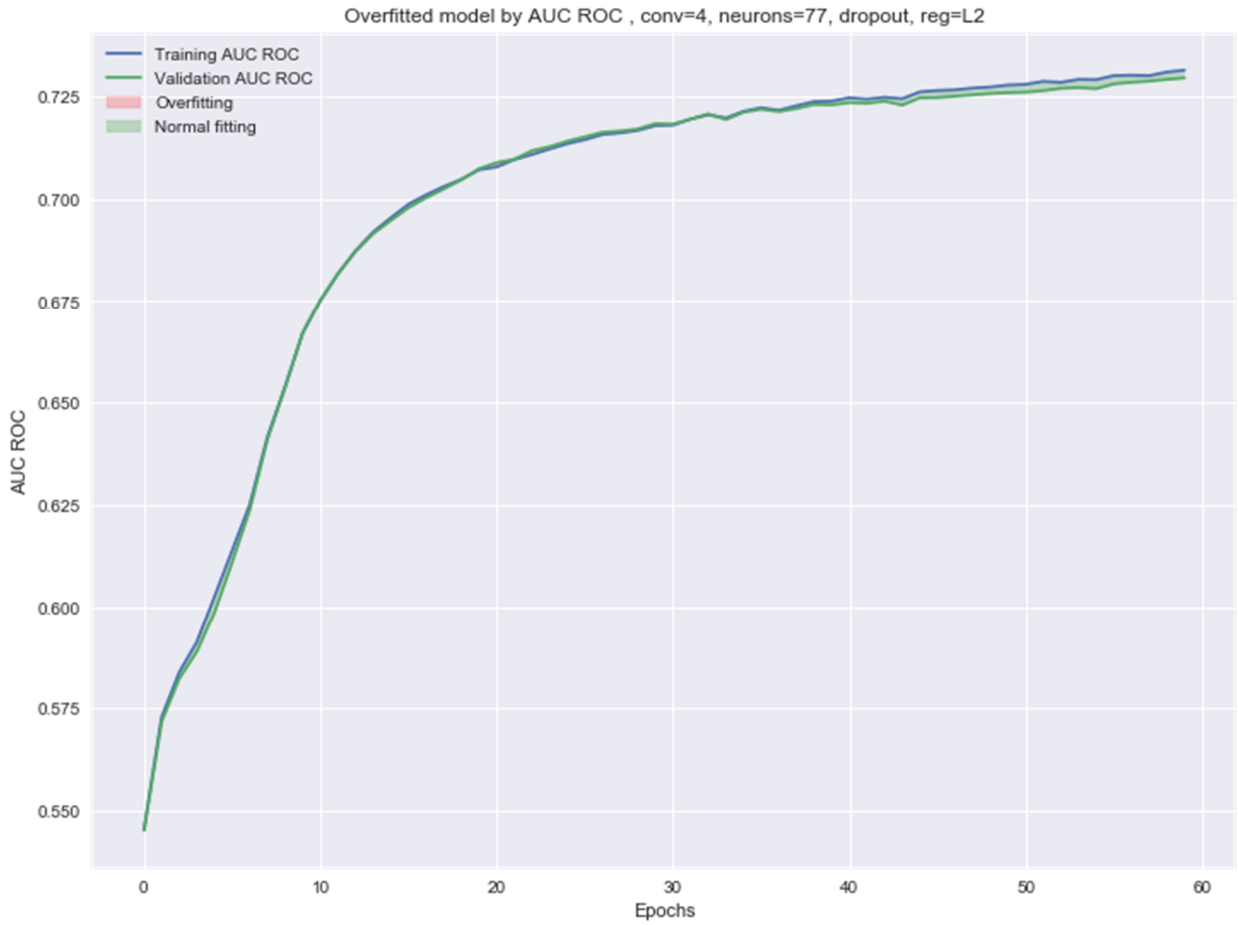


Рисунок 44 – Зависимость AUC ROC от количества эпох (4 скрытых слоя, 77 нейронов; L2; dropout)

Попробуем топологию со сдвоенными сверточными слоями, а именно: Conv -> ReLU -> Conv -> ReLU -> Conv -> ReLU -> MaxPooling -> Conv -> ReLU-> Conv -> ReLU -> Conv -> ReLU -> GlobalAveragePooling -> Dropout -> Dense.

Как видно из рисунка 45, эта модель обучается быстрее, чем предыдущая, но тенденция к росту сократилась. Скорее всего, через какое-то количество эпох, модель выйдет на плато и увеличение итераций не даст значительного прироста в качестве. AUC ROC= 0.7356.

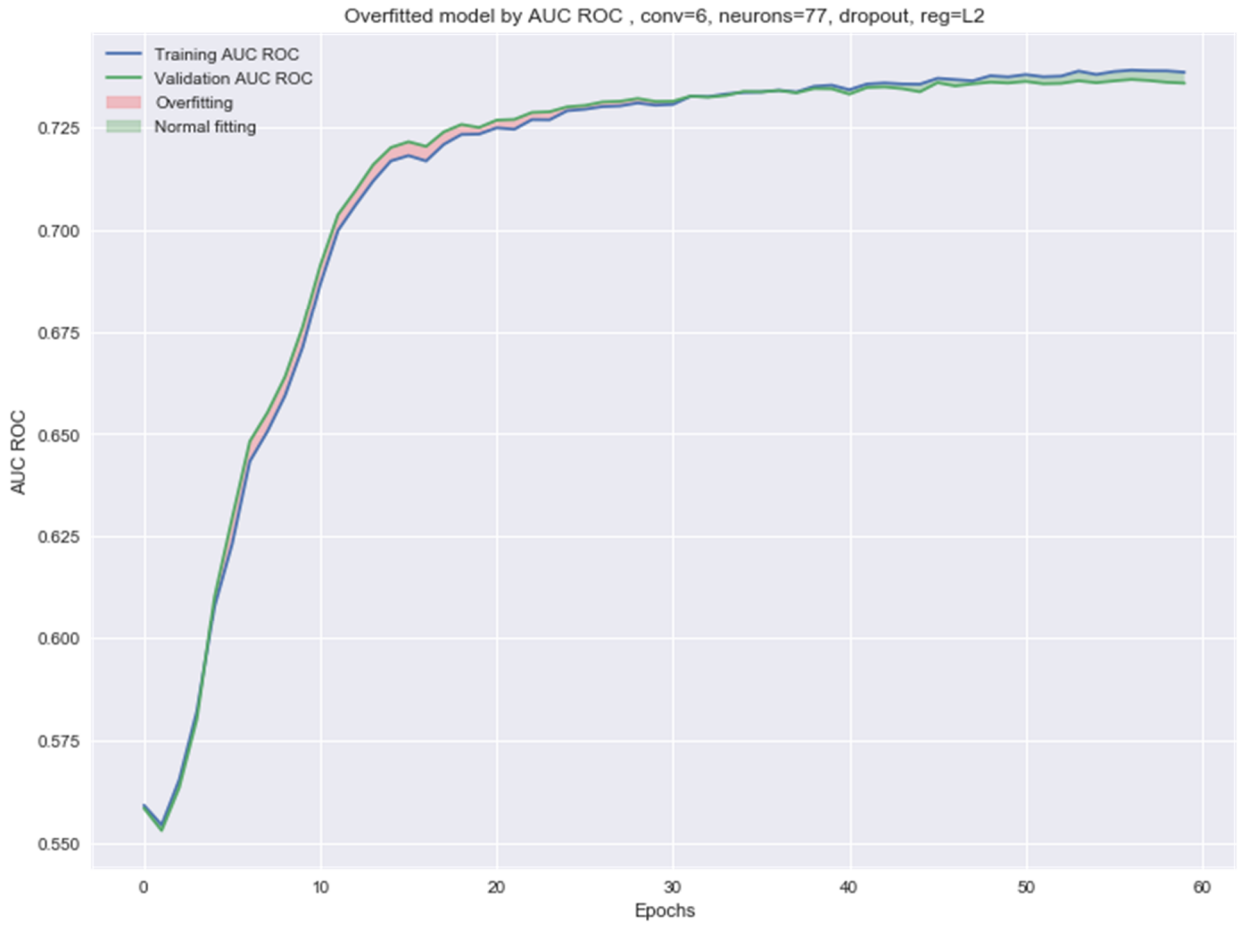


Рисунок 45 – Зависимость AUC ROC от количества эпох (6 скрытых слоев, 77 нейронов; L2; dropout)

Переобучение у сверточных сетей было самое низкое из всех моделей. Однако, скорость обучения и качество на валидационной выборке достаточно низкое. Возможно, при подборе параметров по сетке, результат можно будет незначительно улучшить. Но, скорее всего, придется подбирать другую топологию сети. В виду ограниченности времени и не полным соответствиям целям исследования, изучение этого вопроса лучше провести в рамках других исследований.

3.11. Определение алгоритма, который лучше всего справляется с поставленной задачей

Данное исследование проводилось с целью определить модель, которая лучше всего справится с поставленной задачей.

В ходе исследования были проведены эксперименты со следующими моделями:

- Логистическая регрессия;
- K-ближайших соседей (KNN);
- Случайный лес (Random forest);
- Многослойная нейронная сеть (MLP);
- Сверточная нейронная сеть (CNN).

Лучшие результаты по каждому виду моделей и их параметры представлены в таблице 22.

Таблица 22 – Сводная таблица результатов по разным алгоритмам

Название модели	**Параметры**	**AUC ROC**
Логистическая регрессия	C=0.5, tol=0.00001, solver= liblinear.	0.7198
K-ближайших соседей	n_neighbors=60.	0.7249
Случайный лес	max_features=27, max_depth=18, n_estimators=250.	0.7594
Многослойная нейронная сеть	Первый слой. Нейроны – 300, dropout - 0.2. Второй слой. Нейроны – 150, dropout - 0.25. Третий слой. Нейроны – 150, dropout - 0.2.	0.7573
Сверточная нейронная сеть	Conv -> ReLU -> Conv -> ReLU -> Conv -> ReLU -> MaxPooling -> Conv -> ReLU-> Conv -> ReLU -> Conv -> ReLU -> GlobalAveragePooling -> Dropout -> Dense.	0.7356

Для решения бинарной классификации с представленными данными лучше всего справился алгоритм случайного леса (random forest). Его мы и будем использовать для разработки модуля рекомендаций.

Многослойный персептрон незначительно уступает. Возможно, при более широком поиске по сетке, можно найти более удачные гиперпараметры, которые позволят нейронной сети сравниться со случайным лесом.

Стоит отметить, что на практике, именно эти два алгоритма побеждают в

разнообразных соревнованиях по машинному обучению и анализу данных при решении задач классификации, или хотя бы входят в топ-10.

Также существенное значение оказывает качество предоставленных данных и их дальнейшая предобработка для использования в алгоритмах. Поэтому следует уделять подготовке данных достаточно много времени и усилий, чтобы получить максимальный результат.

4. Экономическое обоснование разработки

4.1. Смета затрат на создание программного изделия

Программный продукт, как и любые другие товары и услуги, подразумевает наличие определенных затрат на его производство. Поэтому крайне важно уметь определять стоимость разрабатываемого программного обеспечения. Чаще всего подобный расчет делается в виде сметы, при этом подробная смета остается у исполнителя, а в договоре прописываются укрупненные счета.

Материальные расходы на создание модуля рекомендаций могут включать следующие затраты:

- Оборудование и программное обеспечение
- Оплата труда
- Накладные расходы
- Прочие расходы

Таблица 23 – Себестоимость

№	**Категория**	**Сумма, руб.**
1	Оборудование и программное обеспечение	5 700
2	Оплата труда	336 480
3	Накладные расходы	52 092
4	Прочие расходы	5 100
5	Итого	399 372

Таблица 24 – Экономическое обоснование

№	Категория	Сумма, руб.
1	Себестоимость	399 372
2	Прибыль	120 000
3	Стоимость с НДС	612 859
4	Стоимость без НДС	519 372

4.2. Обоснование сметы затрат

4.2.1. Расчет затрат на оборудование

Для выполнения квалификационной работы был арендован виртуальный сервер. Стоимость аренды за 3 месяца составила 1 500 рублей.

Также для разработки использовалась уже существующая вычислительная техника. Общая стоимость затрат на оборудование складывается из амортизационных отчислений и стоимости ремонта.

Сначала определим амортизационные отчисления (АО).

$$AO = \frac{\text{cost} \cdot time}{months},$$

где cost – стоимость вычислительной техники, time – время использования для создания модуля рекомендаций в месяцах, months – срок полезного использования вычислительной техники в месяцах.

АО = 42 000 * 3 / 60 = 2 100 рублей

На ремонт обычно закладывают 5% от стоимости вычислительной техники, а именно, 2 100 рублей.

Затраты на собственное оборудование составляют 4 200 рублей.

Всего на оборудование с учетом арендованной техники было затрачено 5 700 рублей.

Использованное программное обеспечение для создания модуля рекомендаций является свободным ПО, поэтому затраты на программное обеспечение отсутствуют.

4.2.2. Расчет оплаты труда

Для работодателя расходы на оплату труда (ОТ) включают не только оклад, прописанный в трудовом договоре, но и отчисления в различные социальные фонды, такие как ПФ РФ, ФСС, ФФОМС. Конкретные тарифы зависят от применяемого налогообложения работодателя, условий труда и заработной платы сотрудников.

Так для организаций на ОСНО без каких-либо льгот с нормальными условиями труда в 2017 году действуют следующие ставки:

- ПФР – 22%.
- ФФОМС – 5.1%.
- ФСС – 2.9%.
- Взнос на “травматизм” – 0.2%

Итого: 30.2% дополнительных отчислений, если сумма начисляемых вознаграждений сотруднику менее 755 000 рублей с начала года.

Если эта сумма превышена – 27.3%. Для расчета будем применять общую ставку в 30.2%.

Также в оплату труда можно заложить расходы на отпуск, различные компенсации и т.д. На эти нужды будем прибавлять еще 10% от заработной платы. Т.е. суммарно – 40.2% к окладу.

Сумма расходов на оплату труда для одного человека составит:
ОТ = 1.402 * оклад * месяцы.

Итого: ОТ = 1.402 * 80 000 * 3 = 336 480 рублей.

4.2.3. Расчет прочих расходов

К этой группе расходов отнесем затраты на канцелярские товары и услуги сторонних организаций.

Стоимость канцелярских товаров (бумага, ручки, картридж и т.д.): 4 500 рублей.

Стоимость услуг (распечатка плакатов): 600 рублей.

Итого прочие расходы: 5 100 рублей.

4.2.4. Расчет накладных расходов

Накладные расходы включают в себя расходы, которые напрямую не связаны с основным производством. Сюда включают расходы на управленческий персонал, оплату офисов, обязательные платежи (для удобства расчета эти суммы учтены в расчете оплаты труда), коммунальные платежи, оплата телефона и т.д.

Обычно точный расчет производит бухгалтерия, так как у другого персонала доступа к подобной информации обычно нет.

Для упрощения расчетов к этой категории отнесем 15% от всех других расходов.

Накладные расходы: (5 700 + 336 480 + 5 100) * 0.15 = 52 092 рублей.

4.2.5. Расчет себестоимости разработки модуля рекомендации

Себестоимость модуля складывается из всех вышеперечисленных расходов, а именно 5 700 + 336 480 + 5 100 + 52 092 = 399 372 рублей.

4.2.6. Расчет прибыли от проекта

Любая организация создается с целью получения прибыли. Поэтому окончательную цену заказчику необходимо указывать с учетом желаемой прибыли. Также необходимо помнить, что организации на ОСНО должны платить 20% налога на прибыль. Если не вдаваться в налоговое законодательство и порядок учета доходов и расходов для определения

налогооблагаемой базы, то прибыль можно определить, как доходы минус расходы (себестоимость). Соответственно, налог на прибыль платится с этой разницы. А чистая прибыль – это прибыль после уплаты налогов.

Чистая прибыль по проекту: 100 000 рублей.

Налог на прибыль: 20 000 рублей.

Прибыль: 120 000 рублей.

4.2.7. Расчет итоговой цены

Общая стоимость проекта включает себестоимость и прибыль: 399 372 + 120 000 = 519 372 рублей.

Также может потребоваться включить в цену налог на добавленную стоимость. Стоит отметить, что НДС применяется не во всех режимах налогообложения. Так как все расчеты были сделаны для организации на ОСНО, то в итоговую цену добавим НДС.

Итоговая цена для клиента: 1.18 * 519 372 = 612 859 рублей

Заключение

При разработке и реализации программного модуля формирования рекомендаций по выдаче банковского кредита были получены следующие результаты:

- Была изучена предметная область.
- Проведен анализ предоставленного набора данных.
- Была поставлена и изучена задача машинного обучения.
- Было проведено исследование алгоритмов для бинарной классификации.
- Был определен лучший алгоритм для решения поставленной задачи.
- Настроено рабочее окружение для развертывания Django и библиотек машинного обучения.
- Разработана инфологическая модель.
- Разработан модуль рекомендаций.
- Разработан программный модуль.
- Разработан интерфейс для программного модуля.
- Построены графики работы модели.
- Произведено экономическое обоснование.
- Подготовлена документация.
- Подготовлены графические материалы.
- Экономический расчет.

Список использованных источников

1. Коэльо Л. П, Ричард В. Построение систем машинного обучения на языке Python. - М.: ДМК Пресс, 2016.

2. Силен Д., Мейсман А., Али М. Основы Data Science и Big Data. Python и наука о данных. - М.: ООО «Питер Пресс», 2017.

3. Маккинни У. Python и анализ данных. - М.: ДМК Пресс, 2015.

4. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. Springer, 2014. — 739 p.

5. Мерков А. Б. Распознавание образов. Введение в методы статистического обучения. 2011. 256 с.

6. Мерков А. Б. Распознавание образов. Построение и обучение вероятностных моделей. 2014. 238 с.

7. Машинное обучение (курс лекций, К.В.Воронцов) [Электронный ресурс] // MachineLearning.ru. URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%BA%D1%83%D1%80%D1%81_%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D0%B9%2C_%D0%9A.%D0%92.%D0%92%D0%BE%D1%80%D0%BE%D0%BD%D1%86%D0%BE%D0%B2) (дата обращения: 05.03.2017).

8. Соколов Е.А.Семинары по машинному обучению, ВМК МГУ [Электронный ресурс] // Персональная страница Соколова Е.А. на GitHub. URL: https://github.com/esokolov/ml-course-msu (дата обращения: 15.03.2017).

9. Машинное обучение (курс лекций, Н.Ю.Золотых) [Электронный ресурс] // Персональный раздел в сети Нижегородского государственного университета им. Н.И. Лобачевского URL: http://www.uic.nnov.ru/~zny/ml (дата обращения: 05.03.2017).

10. Tinkoff Data Science Challenge [Электронный ресурс] // Boosters.pro - Соревнования по программированию и аналитике. URL: https://boosters.pro/champ_3 (дата обращения: 03.03.2017).

11.Django [Электронный ресурс] // The Web framework for perfectionists with deadlines | Django. URL: https://www.djangoproject.com/ (дата обращения: 03.03.2017).

12.Nginx [Электронный ресурс] // nginx. URL: https://nginx.ru/ru/ (дата обращения: 03.03.2017).

13.Docker [Электронный ресурс] // Docker - Build, Ship, and Run Any App, Anywhere. URL: https://www.docker.com/ (дата обращения: 03.03.2017).

14.Pandas [Электронный ресурс] // Python Data Analysis Library — pandas: Python Data Analysis Library. URL: http://pandas.pydata.org/ (дата обращения: 03.03.2017).

15.SciPy [Электронный ресурс] // SciPy.org — SciPy.org. URL: https://www.scipy.org/ (дата обращения: 03.03.2017).

16.Percona Server for MySQL [Электронный ресурс] // Percona Server– An enhanced, drop-in MySQL Replacement URL: https://www.percona.com/software/mysql-database/percona-server (дата обращения: 03.03.2017).

17.scikit-learn [Электронный ресурс] // scikit-learn: machine learning in Python — scikit-learn 0.18.1 documentation. URL: http://scikit-learn.org/stable/ (дата обращения: 03.03.2017).

18.uWSGI [Электронный ресурс] // The uWSGI project — uWSGI 2.0 documentation. URL: https://uwsgi-docs.readthedocs.io/en/latest/ (дата обращения: 03.03.2017).

19.Numpy [Электронный ресурс] // NumPy — NumPy. URL: http://www.numpy.org/ (дата обращения: 03.03.2017).

20.Matplotlib [Электронный ресурс] // Matplotlib: Python plotting — Matplotlib 2.0.2 documentation. URL: https://matplotlib.org/ (дата обращения: 03.03.2017).

Приложение А

Федеральное государственное бюджетное образовательное учреждение
высшего образования

***«Московский государственный технический университет
имени Н.Э. Баумана»
(национальный исследовательский университет)***

УТВЕРЖДАЮ	СОГЛАСОВАНО
_____	_____
"___" _____ 2017 г.	"___" _____ 2017 г.

ПРОГРАММНЫЙ МОДУЛЬ ФОРМИРОВАНИЯ РЕКОМЕНДАЦИЙ ПО ВЫДАЧЕ БАНКОВСКОГО КРЕДИТА

Техническое задание
(вид документа)

Листы А4
(вид носителя)

_____5_____
(количество листов)

Исполнитель: студент группы ВИУ5-81

_____ М. О. Таран

"_____"_____ 2017 г.

Москва, 2017

1. Наименование

Модуль рекомендаций для сотрудника банка, который будет прогнозировать взятие кредита в целевом банке на основе предоставленного набора данных с целью сокращения издержек на проверку заявок. Полное название «Программный модуль формирования рекомендаций по выдаче банковского кредита». Краткое название «Модуль рекомендаций»

2. Основание для разработки

Основанием для разработки является задание на квалификационную работу, подписанное исполнителем квалификационной работы, руководителем квалификационной работы, заведующим кафедрой СОИУ МГТУ им. Баумана.

3. Исполнитель

Студентка группы ВИУ5- 81.

4. Назначение и цель разработки

Объектом квалификационной работы является программный модуль формирования рекомендаций по выдаче банковского кредита, который позволит специалисту банка принимать решение о дальнейших действиях с каждой вновь поступившей кредитной заявкой.

Цель работы – создание модуля рекомендаций на основе алгоритмов машинного обучения для решения задачи бинарной классификации на основе предоставленного набора данных с кредитными заявками из магазинов электроник.

4. Содержание работы

4.1. Задачи, подлежащие реализации:

- Анализ предметной области.
- Анализ предоставленного набора данных.
- Постановка задачи машинного обучения и определение алгоритма, который лучше всего справляется с поставленной задачей.
- Настройка рабочего окружения для развертывания Django и библиотек машинного обучения.

- Разработка модуля рекомендаций.
- Разработка программного модуля.
- Экономический расчет.

4.2. Требования к функциональным характеристикам

Модуль рекомендаций должен обладать следующим функционалом:

- Подготовка данных – полученные данные из базы данных кредитных заявок обрабатывать и преобразовывать в формат, пригодный для использования в математической модели.
- Загрузка модели – обученную математическую модель загружать в оперативную память для дальнейшего использования.
- Предсказание – по преобразованным данным предсказывать вероятность возврата клиента в банк.
- Сохранение и возврат прогноза – сохранение в базу данных предсказанных значений конкретной заявки и возвращение в основную систему предсказанных результатов.

4.3. Требования к составу программных средств

Клиентская часть является графическим браузером Firefox 53.0.3, под управлением операционной системы Windows 10.

Серверная часть разрабатываемого проекта предусмотрена для работы в среде, состоящей из следующих компонентов:

- CentOS 6.8
- Docker 1.13
- uWSGI 2.0.13.1
- Nginx 1.11.10
- PerconaDB 5.6
- Python 3.5
- Django 1.8

- Pandas 0.19.2
- Sklearn 0.18.1
- Numpy 1.12.1
- Matplotlib 2.0.0
- Scipy 0.19.0

4.4. Требования к программным компонентам

Основным требованием к программным компонентам является корректное исполнение функций, заявленных в пункте 4.2.

5. Требования к входным и выходным данным

5.1. Требования к входным и выходным данным:

Входные данные для модуля рекомендаций:

- Кредитные заявки.

Выходные данные для модуля рекомендаций:

- Предсказанный класс.
- Предсказанная вероятность отнесения к классу 1.

5.3. Требования к архитектуре системы

Модуль рекомендаций должен быть разработан для сайтов построенных на основе Django 1.8, при этом графическая часть в модуле не предусмотрена. Соответственно, обработка и представление выходных данных модуля должны быть реализованы в зависимости от сайта. Вся программная часть должна находиться внутри одного docker-контейнера на виртуальном сервере.

Клиентская часть представлена стандартным веб-браузером.

5.4. Требования к составу технических средств

Клиентская часть модуля рекомендаций предполагает реализацию на персональном компьютере с конфигурацией, приведенной ниже:

- процессор Intel, 2.5 GHz и выше,
- RAM не менее 4 Гбайт,
- монитор,
- клавиатура,

● мышь или другой манипулятор,

● оборудование для доступа к сети Интернет.

Серверная часть проекта выполняется на арендованном виртуальном сервере со следующей конфигурацией:

● процессор 2xXeon X5675,

● RAM не менее 2 Гбайт,

● дисковая подсистемы со свободным пространством не менее не менее 10 Гбайт.

● оборудование для доступа к сети Интернет.

6. Этапы разработки

Таблица 1 – Этапы разработки

Наименование этапа	Дата выполнения
Разработка технического задания	до 2017-04-03
Разработка эскизного проекта	до 2017-04-15
Разработка технического проекта	до 2017-05-15
Разработка программы	до 2017-05-31
Отладка программы	до 2017-06-08
Разработка программной документации	до 2017-06-20
Защита квалификационной работы	до 2017-06-30

7. Техническая документация, предъявляемая по окончании работы

По окончании работы предъявляются следующие документы:

1. Техническое задание.
2. Расчётно-пояснительная записка.
3. Программа и методика испытаний.
4. Руководство пользователя.
5. Текст программы.

6. Графический материал по проекту в формате листов А1.

8. Порядок приемки работы

Приемка работы осуществляется в соответствии с документом "Программа и методика испытаний".

9. Дополнительные условия

Техническое задание может уточняться в установленном порядке.

Приложение Б

Федеральное государственное бюджетное образовательное учреждение
высшего образования

***«Московский государственный технический университет
имени Н.Э. Баумана»
(национальный исследовательский университет)***

УТВЕРЖДАЮ	СОГЛАСОВАНО
_____	_____
"___" _____ 2017 г.	"___" _____ 2017 г.

ПРОГРАММНЫЙ МОДУЛЬ ФОРМИРОВАНИЯ РЕКОМЕНДАЦИЙ ПО ВЫДАЧЕ БАНКОВСКОГО КРЕДИТА

Порядок и методика испытаний
(вид документа)

Листы А4
(вид носителя)

_____4_____
(количество листов)

Исполнитель: студент группы ВИУ5-81

_____ М. О. Таран

"_____"_____ 2017 г.

Москва, 2017

Порядок и методика испытаний

1. Цель испытаний

Цель испытаний состоит в проверке работоспособности программного модуля и соответствия выполняемых функций требованиям документа “Техническое задание”.

2. Состав предъявляемой документации

Перед проведением испытаний предъявляются следующие документы:

- Техническое задание;
- Порядок и методика испытаний.

3. Технические требования

3.1. Требования к программной документации

Состав программной документации должен удовлетворять требованиям документа “Техническая документация”, предъявляемого по окончании работы.

Программная документация должна быть оформлена в соответствии с ГОСТ и ЕСПД по составлению и оформлению документов на программное изделие.

3.2. Требования к составу технических средств

3.2.1. Требования к программному обеспечению

Серверная часть должна выполняться в docker-контейнере со следующим ПО:

- CentOS 6.8
- uWSGI 2.0.13.1
- Nginx 1.11.10
- PerconaDB 5.6
- Python 3.5
- Django 1.8
- Pandas 0.19.2
- Sklearn 0.18.1

- Numpy 1.12.1
- Matplotlib 2.0.0
- Scipy 0.19.0

Клиентская часть является графическим браузером Firefox 53.0.3, под управлением операционной системы windows 10.

3.2.2. Требования к аппаратному обеспечению

Серверная часть подсистемы модуля должна функционировать на x64 совместимой архитектуре со следующими требованиями к аппаратной конфигурации:

1. Процессор 2xXeon X5675;

2. ОЗУ 2 Гб;

3. Место на жестком диске не менее 10 Гб;

Рекомендуемая аппаратная конфигурация:

4. Сетевая карта с пропускной способностью не менее 100 Mbps.

Клиентская часть подсистемы модуля должна функционировать на X32/X64 совместимой архитектуре со следующими требованиями к аппаратной конфигурации:

1. Процессор x64 архитектурой с тактовой частотой от 2000 МГц;

2. ОЗУ не менее 2Гбайт;

3. Дисковая подсистема со свободным пространством, необходимым для установки ОС и web-браузера;

4. Наличие манипулятора типа “мышь”;

5. Графический адаптер и совместимый с ним монитор, поддерживающие разрешение экрана не менее 1280x720;

6. Доступ в Интернет и необходимое для этого оборудование.

4. Порядок проведения испытаний

Испытание системы должно проводиться в следующей последовательности:

1. Запуск программы-клиента;

2. Проведение испытаний в соответствии с требованиями документа “Техническое задание”;

3. Завершение работы.

5. Программа испытаний

Таблица 1 – программа испытаний

№	Действие	Результат	Примечания
1	Запустить web-браузер и ввести в адресной строке URL-адрес сайта	Выводится форма авторизации на сайте	
2	Ввести логин и пароль: demo demo	Отобразится страница со списком кредитных заявок	
3	Нажать на ссылку с номером заявки	Откроется модальное окно с описанием заявки	
4	Нажать на кнопку “Новое” у заявки в колонке статус заявки	Откроется список доступных статусов заявки	
5	Выбрать статус “Одобрен” или “Отклонен”	В колонке статус отобразится выбранный статус	
6	Перейти на страницу 4. У любой заявки нажать на кнопку “Обновить”	Через пару секунд в колонке прогноз появится значение, если его там не было, или обновится, если было.	

7	Нажать на кнопку “Статистика” в верхнем меню	Откроется страница с графиками	

6. Результат испытания

Основой испытаний является осуществление всех возможных пользовательских действий. Испытание считается пройденным успешно, если в его процессе не была выведена страница с системной ошибкой.

Приложение В

Федеральное государственное бюджетное образовательное учреждение
высшего образования

***«Московский государственный технический университет
имени Н.Э. Баумана»
(национальный исследовательский университет)***

УТВЕРЖДАЮ

_____

"___" _____ 2017 г.

СОГЛАСОВАНО

_____

"___" _____ 2017 г.

ПРОГРАММНЫЙ МОДУЛЬ ФОРМИРОВАНИЯ РЕКОМЕНДАЦИЙ ПО ВЫДАЧЕ БАНКОВСКОГО КРЕДИТА

Руководство пользователя
(вид документа)

Листы А4
(вид носителя)

_____5_____
(количество листов)

Исполнитель: студент группы ВИУ5-81

_____ М. О. Таран

"_____"_____ 2017 г.

Москва, 2017

Руководство пользователя программного модуля

После включения модуля рекомендаций на сайте появляется возможность просматривать прогноз по каждой кредитной заявке.

Данное руководство предназначено для специалиста банка, который зарегистрирован на сайте и имеет доступ к кредитным заявкам.

1. Вход на сайт

Чтобы зайти на сайт необходимо в адресной строке браузера ввести <адрес сайта> и нажать enter. После перехода на сайт откроется страница с формой для ввода логина и пароля, если пользователь не был авторизован ранее. (рисунок 1).

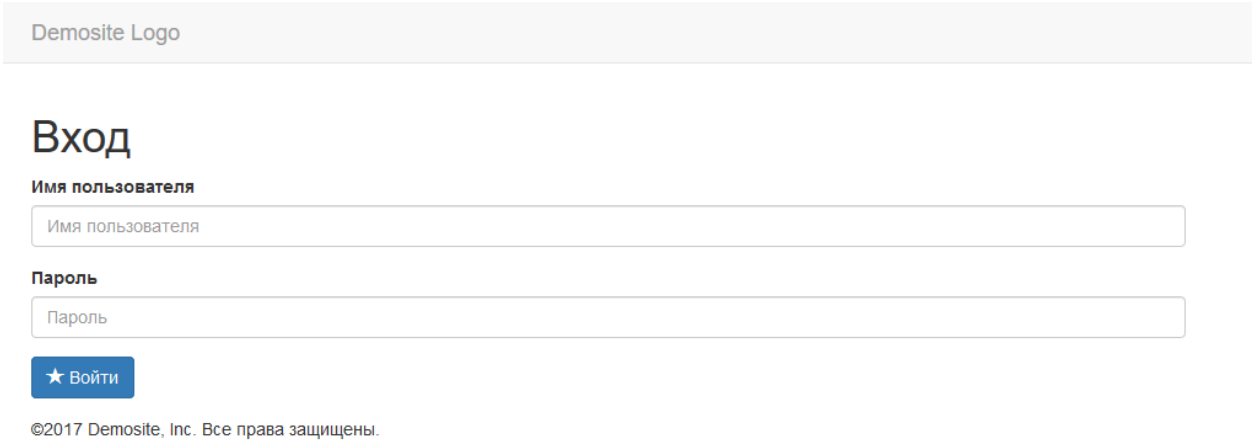
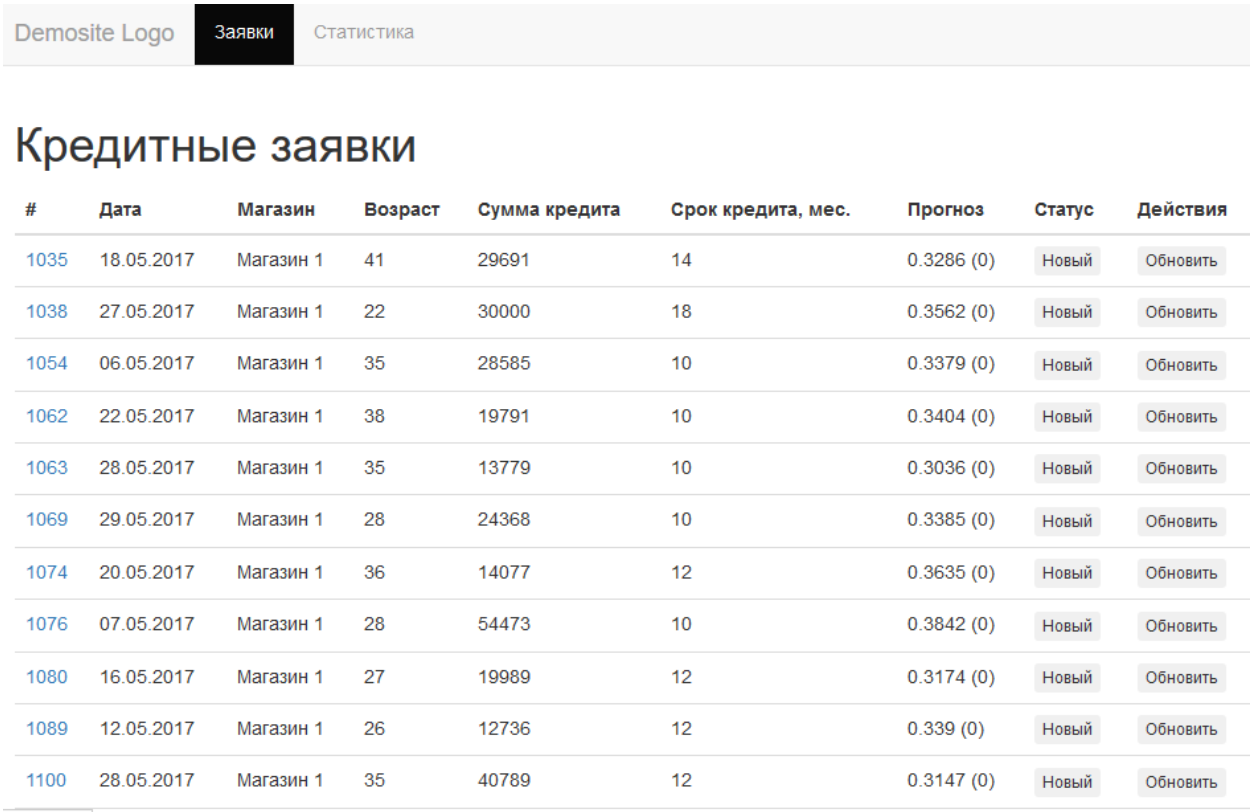


Рисунок 1 – Вход на сайт

2. Кредитные заявки

После авторизации пользователь попадает на страницу со списком заявок. На ней он сможет посмотреть все заявки, которые приходят из разных магазинов (рисунок 2).



#	Дата	Магазин	Возраст	Сумма кредита	Срок кредита, мес.	Прогноз	Статус	Действия
1035	18.05.2017	Магазин 1	41	29691	14	0.3286 (0)	Новый	Обновить
1038	27.05.2017	Магазин 1	22	30000	18	0.3562 (0)	Новый	Обновить
1054	06.05.2017	Магазин 1	35	28585	10	0.3379 (0)	Новый	Обновить
1062	22.05.2017	Магазин 1	38	19791	10	0.3404 (0)	Новый	Обновить
1063	28.05.2017	Магазин 1	35	13779	10	0.3036 (0)	Новый	Обновить
1069	29.05.2017	Магазин 1	28	24368	10	0.3385 (0)	Новый	Обновить
1074	20.05.2017	Магазин 1	36	14077	12	0.3635 (0)	Новый	Обновить
1076	07.05.2017	Магазин 1	28	54473	10	0.3842 (0)	Новый	Обновить
1080	16.05.2017	Магазин 1	27	19989	12	0.3174 (0)	Новый	Обновить
1089	12.05.2017	Магазин 1	26	12736	12	0.339 (0)	Новый	Обновить
1100	28.05.2017	Магазин 1	35	40789	12	0.3147 (0)	Новый	Обновить

Рисунок 2 – Кредитные заявки

В таблице представлены следующие основные данные:

- Номер заявки.
- Дата поступления заявки.
- Магазин.
- Возраст клиента.
- Сумма кредита.
- Срок кредита.
- Прогноз модуля рекомендаций.
- Статус заявки.
- Действия для запуска прогнозирования в ручном режиме.

3. Экранная форма подробной информации о заявке

Оператор может посмотреть подробную информацию по заявке, если нажмет на ссылку с номером заявки (рисунок 3).

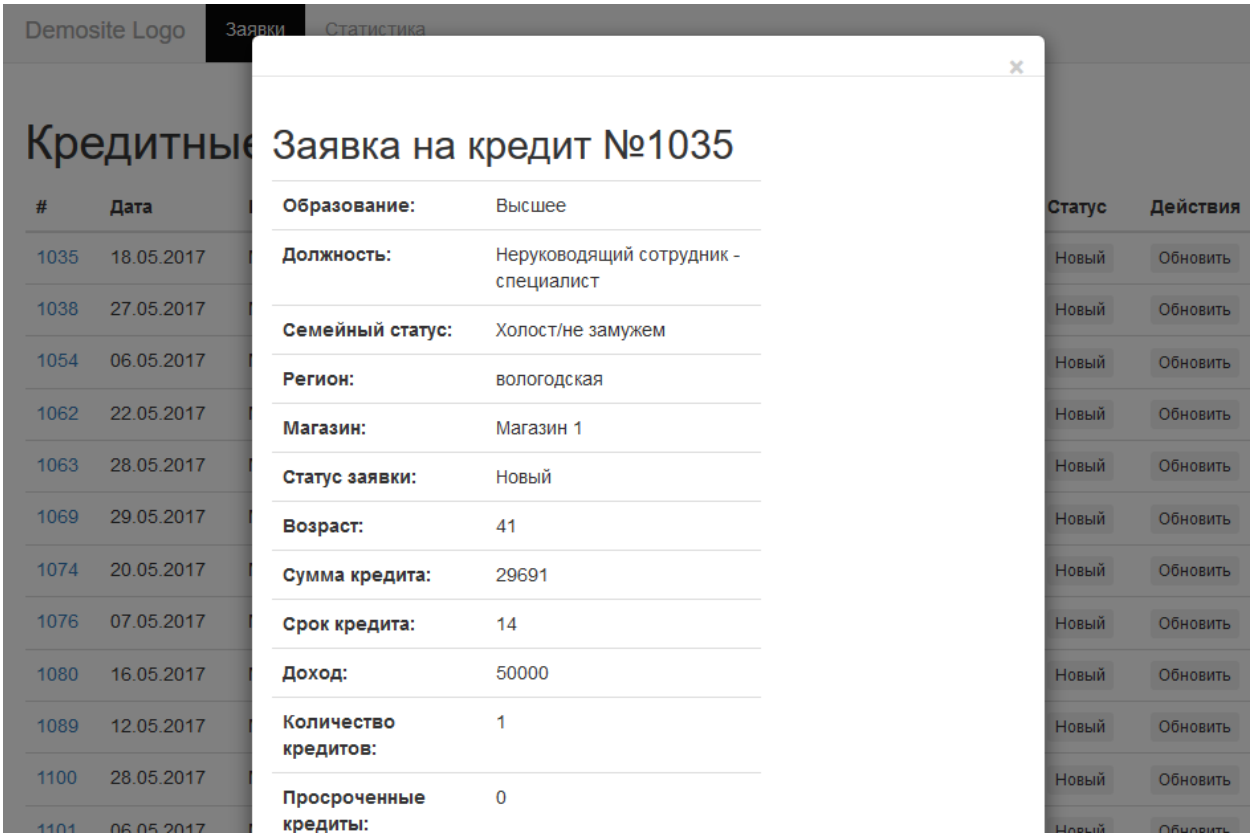


Рисунок 3 – Кредитная заявка

Ему будет продемонстрирована всплывающее окно, в котором будут описана следующая информация:

- Образование.
- Должность.
- Семейный статус.
- Регион.
- Магазин.
- Статус заявки.
- Возраст.

- Сумма кредита.
- Доход.
- Количество кредитов.
- Просроченные кредиты.

Модальное окно с информацией по заявке можно закрыть, нажав на крестик или кликнув мышкой за пределами этой всплывающей формы.

4. Статистика

Чтобы посмотреть на графики со статистикой заявок нужно перейти на страницу <адрес сайта>/stats/. Оператору будут доступны несколько видов отчетов (рисунок 4), а именно:

- Количество заявок по дням.
- Средняя сумма кредита в день.
- Количество заявок по полу клиентов.
- Количество заявок по статусу заявки.
- Количество заявок по возрасту.

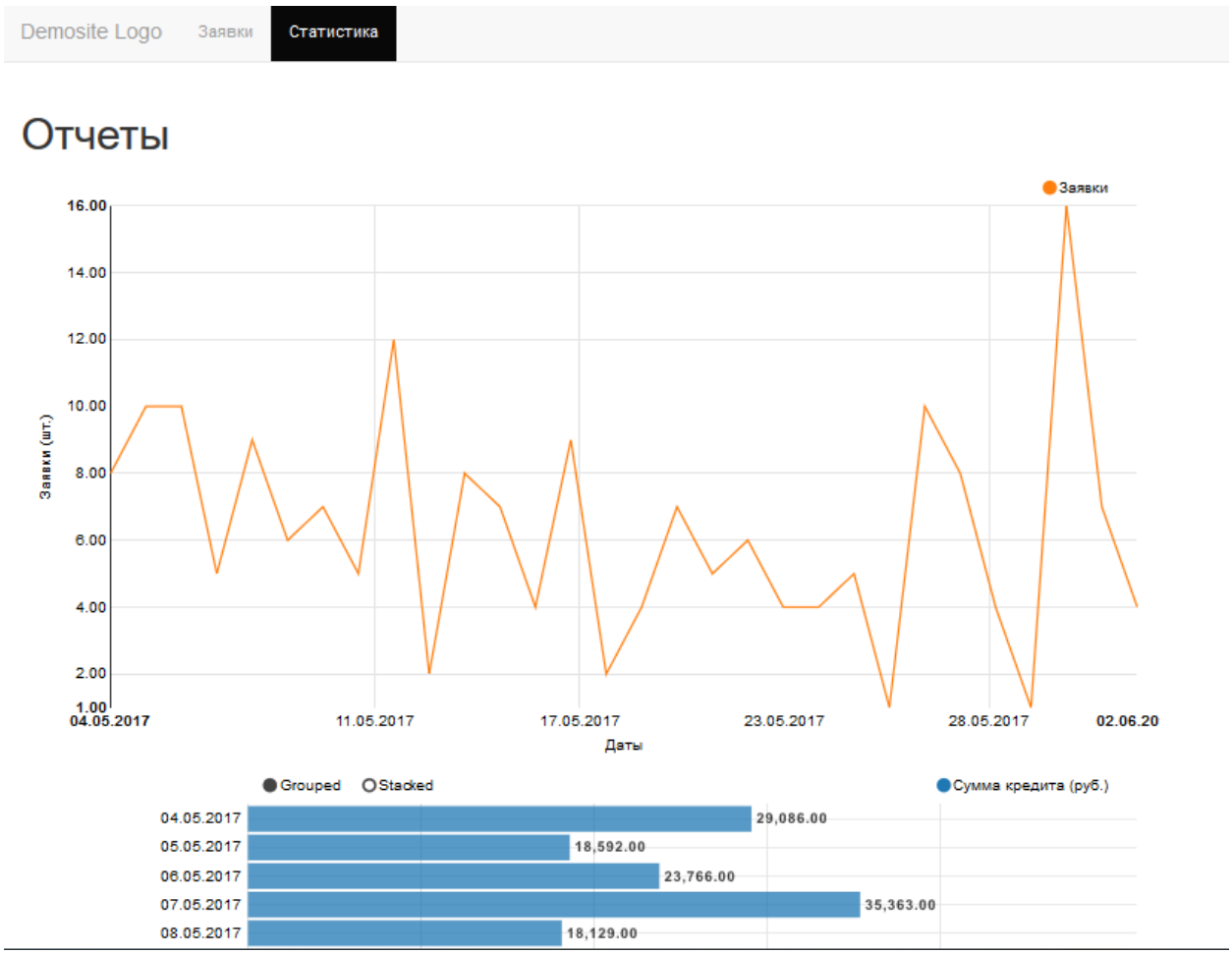


Рисунок 4 – Страница со статистикой

Приложение Г

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Московский государственный технический университет имени Н.Э. Баумана»
(национальный исследовательский университет)

УТВЕРЖДАЮ

_____

"___" _____ 2017 г.

СОГЛАСОВАНО

_____

"___" _____ 2017 г.

ПРОГРАММНЫЙ МОДУЛЬ ФОРМИРОВАНИЯ РЕКОМЕНДАЦИЙ ПО ВЫДАЧЕ БАНКОВСКОГО КРЕДИТА

Текст программы
(вид документа)

Листы А4
(вид носителя)

19
(количество листов)

Исполнитель: студент группы ВИУ5-81

_____ М. О. Таран

"_____"_____ 2017 г.

Москва, 2017

Текст программы

Приложение Д

Федеральное государственное бюджетное образовательное учреждение
высшего образования

***«Московский государственный технический университет
имени Н.Э. Баумана»
(национальный исследовательский университет)***

УТВЕРЖДАЮ	СОГЛАСОВАНО
_____	_____
"___" _____ 2017 г.	"___" _____ 2017 г.

ПРОГРАММНЫЙ МОДУЛЬ ФОРМИРОВАНИЯ РЕКОМЕНДАЦИЙ ПО ВЫДАЧЕ БАНКОВСКОГО КРЕДИТА

Графические материалы
(вид документа)

Листы А4
(вид носителя)

_____6_____
(количество листов)

Исполнитель: студент группы ВИУ5-81

_____ М. О. Таран

"_____"_____ 2017 г.

Москва, 2017

ПРОГРАММНЫЙ МОДУЛЬ ФОРМИРОВАНИЯ РЕКОМЕНДАЦИЙ ПО ВЫДАЧЕ БАНКОВСКОГО КРЕДИТА

Цель:
Разработать модуль, который будет прогнозировать взятие кредита в целевом банке на основе предоставленного набора данных.

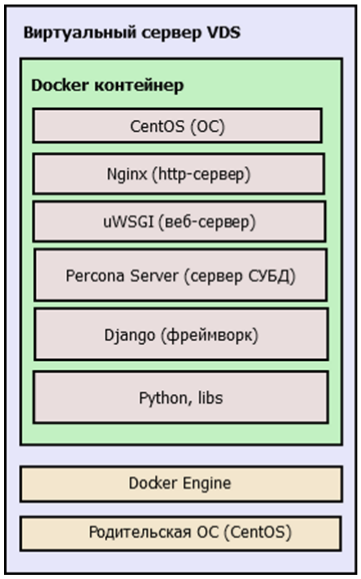
Задачи:

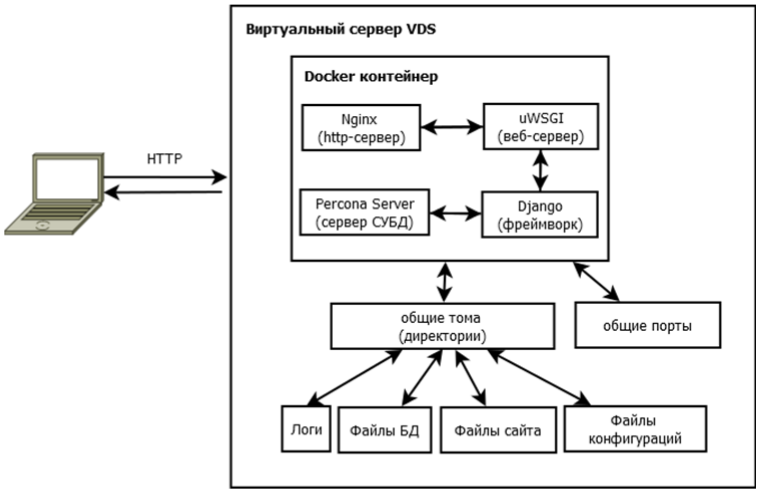
- Анализ предметной области.
- Анализ предоставленного набора данных.
- Постановка задачи машинного обучения и определение алгоритма, который лучше всего справляется с поставленной задачей.
- Настройка рабочего окружения для развертывания Django и библиотек машинного обучения.
- Разработка модуля рекомендаций.

Сравнение аналогов

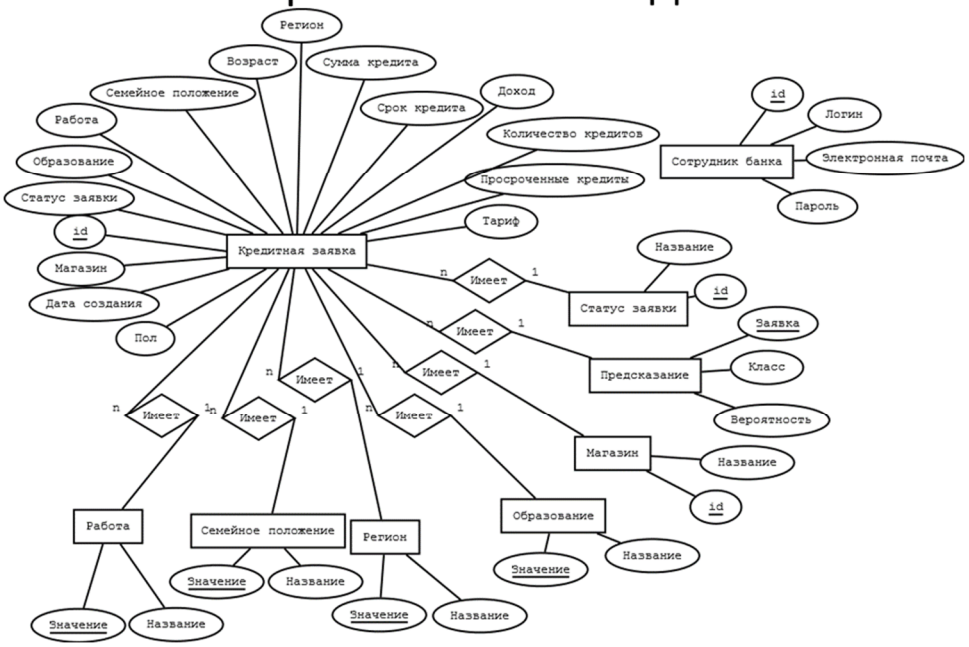
Критерии сравнения	Весовой коэффициент	LMS	Кредит-Конвейер	First Loan Bank	Модуль рекомендаций
Интерфейс	0,1	5	7	7	6
Наглядность	0,1	4	8	7	7
Интеграция	0,2	6	9	8	9
Функциональ-ность	0,1	5	8	7	7
Машинное обучение	0,3	0	2	4	8
Использование собственных данных банка	0,2	3	4	4	8
Интегральный показатель	1	3,2	5,5	5,7	7,8

Архитектура системы



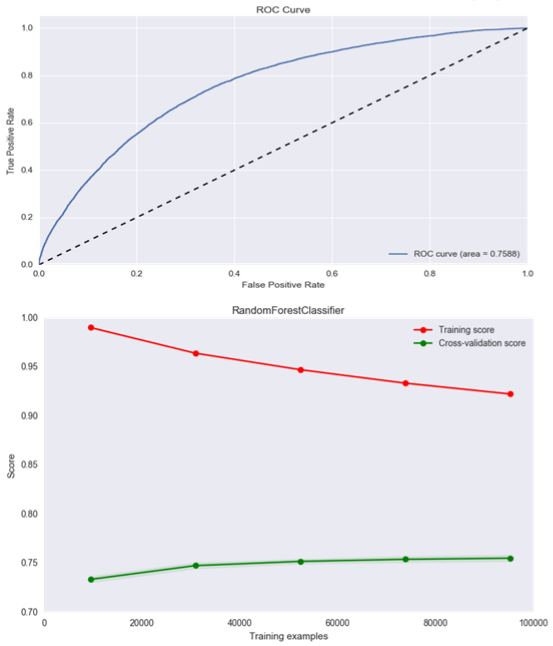


Инфологическая модель



Даталогическая модель

Исследовательская часть



Название модели	Параметры	AUC ROC
Логистическая регрессия	C=0.5, tol=0.00001, solver= liblinear.	0.7198
К-ближайших соседей	n_neighbors=60.	0.7249
Случайный лес	**max_features=27, max_depth=18, n_estimators=250.**	**0.7594**
Многослойная нейронная сеть	Первый слой. Нейроны – 300, dropout - 0.2. Второй слой. Нейроны – 150, dropout - 0.25. Третий слой. Нейроны – 150, dropout - 0.2.	0.7573
Сверточная нейронная сеть	Conv -> ReLU -> Conv -> ReLU -> Conv -> ReLU -> MaxPooling -> Conv -> ReLU-> Conv -> ReLU -> Conv -> ReLU -> GlobalAveragePooling -> Dropout -> Dense.	0.7356

Экранные формы