Санкт-Петербургский государственный университет

Кафедра теории систем управления электрофизической аппаратурой

Хачатрян Альберт Гагикович

Магистерская диссертация

Методы интеллектуального анализа данных в задаче построения безопасного маршрута

Направление 03.04.01
Прикладная математика и физика
Магистерская программа математические и информационные технологии

Научный руководитель,
Кандидат технических наук
Блеканов Иван Станиславович

Санкт-Петербург
2018

Содержание

Введение

Актуальность

Актуальность повышения уровня безопасности на дорогах связана с тем, что дорожно транспортные происшествия (ДТП) являются частой причиной ущерба гражданам и угрозе их жизни и здоровью. Безопасность дорожного движения является одной из важных социально -экономических и демографических задач Российской Федерации.

Согласно данным государственной инспекции безопасности дорожного движения (ГИБДД), за 2017 год на территории Санкт-Петербурга и Ленинградской области произошло 6943 аварии. Из этого следует, что в прошлом году в среднем в день происходило около 19 аварий, что значительно меньше, чем показатели за 2016 год – 21 аварий. Однако получается, что практически каждый час на улицах Санкт-Петербурга происходит одна авария.

На сегодняшний день на рынке существует большое количество сервисов и мобильных приложений, позволяющих строить маршрут, наиболее популярные это сервисы Яндекс.Карты, Google Maps, 2GIS и OpenStreetMap. Но в этих проектах не используются для построения маршрутов открытые данные об аварийности, например представленные на официальном сайте ГИБДД. Вышеуказанные сервисы строят наикратчайший маршрут или самый быстрый с учетом загруженности дорог, но информацией о загруженности дорог обладают некоторые из них. Также существует ряд ограничений для использования сервисов в своих проектах, таких как суточное ограничение на количество запросов к сервису. Пользователю выдается очень мало информации о построенном маршруте, время, расстояние и в некоторых информация о том, что маршрут проходит через платный участок.

Для решения вышеописанных проблем былва поставлена задача разработать метод и общедостыпный программный комплекс (веб-сервис) который строит маршрут для водителей, детально анализируя его и классифицируя как опасный или безопасный, опираясь на статистические данные об аварийности. Если существует несколько вариантов маршрутов, то сервис предложит наименее опасный из них. Также программа сообщит об особо опасных участках дороги, которые встретятся пользователю на его пути.

Данное решение позволит пользователям (водителям) получаеть не только стандартную информацию, которую могут предложить современные карты или сервисы, но и детально разобранный по отрезкам (от перекрестка до перекрестка) маршрут с информацией об аварийности на каждом участке и классе всего маршрута (опасный или безопасный). В этом

заключается новизна данной магистерской диссертации.

Сервис в работе опирается на уже существующие программные решения в этой области(такие как Google Maps и Яндекс карты). С помощью этих технологий строится маршрут, при этом пользователь видит только обычную карту с проложенным маршрутом, в то время как программа, используя данную технологию, получает координаты перекрестков, длину каждого участка, название улицы, а также общие сведения о маршруте – загруженность дорог и общее время пути. Используя полученные данные, программа проводит заложенные в нее вычисления, а именно считается аварийность маршрута на 1 км и риск попасть в аварию на этом маршруте; в качестве ответа выдает класс построенного маршрута и выделяет наиболее опасные участки дороги.

Цель работы

Основная цель работы, создание методов интеллектуального анализа данных, ориентированных на повышение уровня безопасности на дорогах общего пользования, путем предоставления водителям менее аварийного маршрута, и выявления наиболее аварийных участков.

Интеграция созданных методов анализа данных в общедоступный веб-сервис, позволит пользователям в режиме реального времени получать, через устройства подключенные к интернету, следующую информацию:

- Безопасный маршрут по заданным адресам и детальную информацию об аварийности построенного маршрута (наглядную демонстрацию маршрута на карте, показатели аварийности каждого участка, расстояние участка и краткую инструкцию);
- Статистику аварийности в выбранном регионе за указанный период времени (количество аварий, общее количество участников, количество раненых, погибших и т.д);
- Персонализированную информацию (сохраненые пользователем маршруты и добавленные в избранное адреса)

Показатели аварийности участка используются для поиска наиболее аварийных отрезков маршрута, к аварийным отрезкам маршрута относятся, те на которых количество аварийных ситуаций критично относительно найденных ранее показателей, что является следствием некорректной организации дорожного движения или некачественного дорожного покрытия. Устранение вышеописанных недостатков на аварийных участках, позволит добиться улучшения уровня безопасности на дорогах общего пользования.

Основные задачи

Для достижения выше описанной цели были поставлены следующие задачи:

1. Обзор существующих сервисов, инструментов и методов построения маршрута;
2. Сбор и автоматизация получения необходимых данных для анализа безопасности;
3. Разработка метода для оценки уровня безопасности построенного маршрута;
4. Разработка архитектуры программного комплекса (веб-сервиса) для хранения, анализа и построения маршрута;
5. Программная реализация общедоступного веб сервиса для построения безопасного маршрута и получения общей статистики об аварийности;
6. Апробация разработанного комплекса и внедрение продукта в сторонние сервисы;

Глава 1. Обзор существующих сервисов, инструментов и методов

При исследовании проектов, в основной функционал которых входит построение маршрутов для проезда на автомобиле и получение информации по постороннему маршруту, были выявлены наиболее распространенные на сегодняшний день веб-сервисы:

- Картографический проект OpenStreetMap [8].

  Веб-сервис, позволяющий просматривать карту всего мира, искать географические объекты на карте по заданному адресу или по заданным координатам, строить маршруты и получать информацию о географических объектах.

  Данные сервис получает силами сообщества участников (пользователей интернета), для создания карт использовались различные источники: GPS-трекеров, аэрофотографии, видеозаписи, спутниковые снимки и панорамы улиц.

  Обновление информации (добавление или удаление географических объектов) на карте происходит по принципу вики. Каждый пользователь, зарегистрированный в системе, может вносить свои изменения на карту, что может послужить причиной некорректного построения маршрута. Также данный сервис не владеет информацией о текущей загруженности улиц, что является основной причиной некорректного подсчета времени которое пользователь затратит на маршрут.

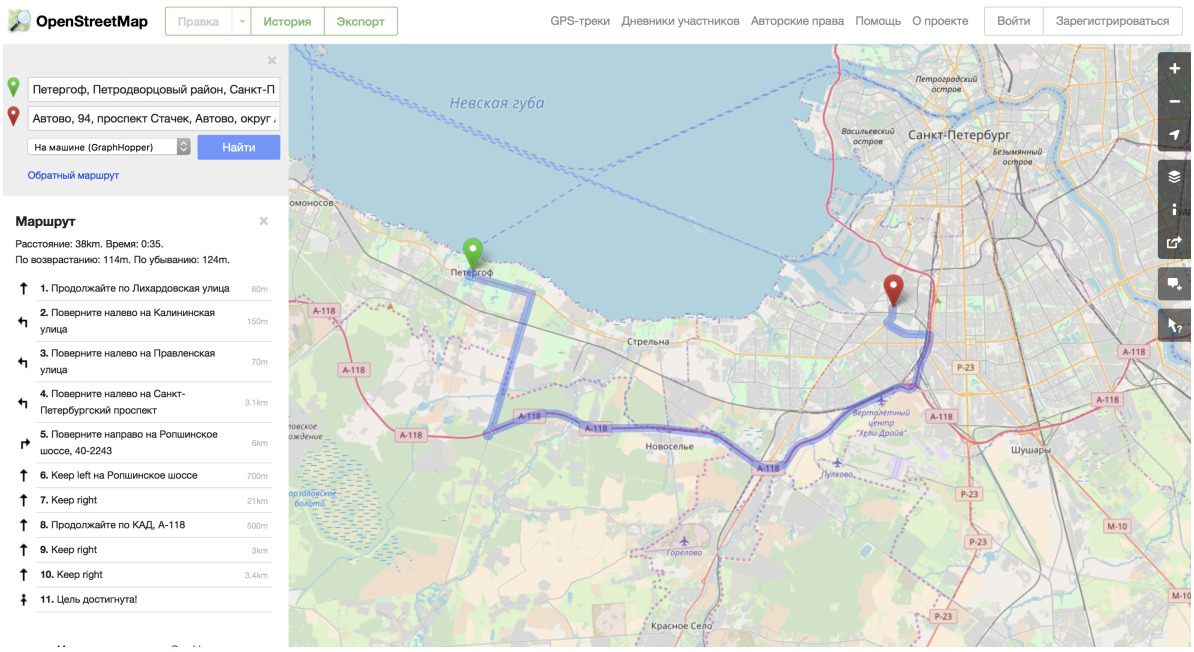

Рис. 1.1. Маршрут построенный сервисом OSM

- Международная картографическая компания 2GIS ("ДубльГИС") [9]

По данным на 2016 год в линейку продуктов 2GIS входят: онлайн-версия, мобильная версия, версия для персонального компьютера, 2GIS Dialer, 2GIS для браузеров, 2GIS API.

Рассмотрим онлайн-версию, она включает в себя линейку для измерения расстояний, поиск проезда на общественном и личном транспорте, справочник организаций, отображение пробок в некоторых городах.

Обновлением данных в компании занимается выделеный отдел специалистов, выверяющие карты на местности и контакт-центр, актуализирующий информацию в справочнике. Частота актуализации четыре раза в год.

При построении маршрута для проезда на личном транспорте, прокладывается самый короткий маршрут, но не самый быстрый, так как сервис владеет неполной информацией о загруженности дорог.

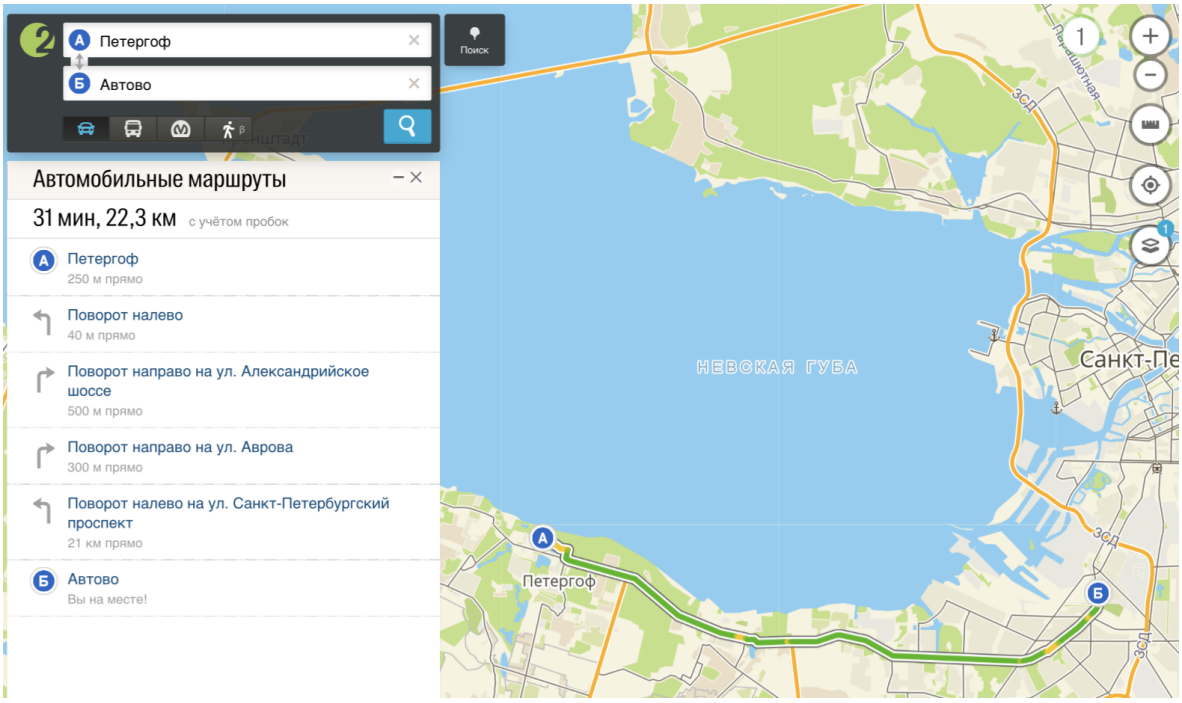

Рис. 1.2. Маршрут построенный сервисом 2GIS

- Карты Google (Google Maps) [10]

  Картографический сервис, включающий в себя набор приложений и технологий, реализованных компаний Google. Сервис поддерживает несколько режимов отображения: карта (картографические данные), ландшафт (отображение физических элементов, например растительность и горы), спутник (спутниковые и аэрофотоснимки)

  С сервисом интегрирован бизнес-справочник и карта автомобильных дорог с поиском маршрутов. При построении маршрута сервис предлагает на выбор 3 маршрута, с различной длинной и временем. Сервис обладает свежими данными о загруженности улиц, однако для использования технологий Google Maps в своих проектах существует особые

ограничения. Цены и количество разрешенных запросов в сервис регулируется компанией, на текущий период доступно 2500 запросов в сутки на построение маршрута.

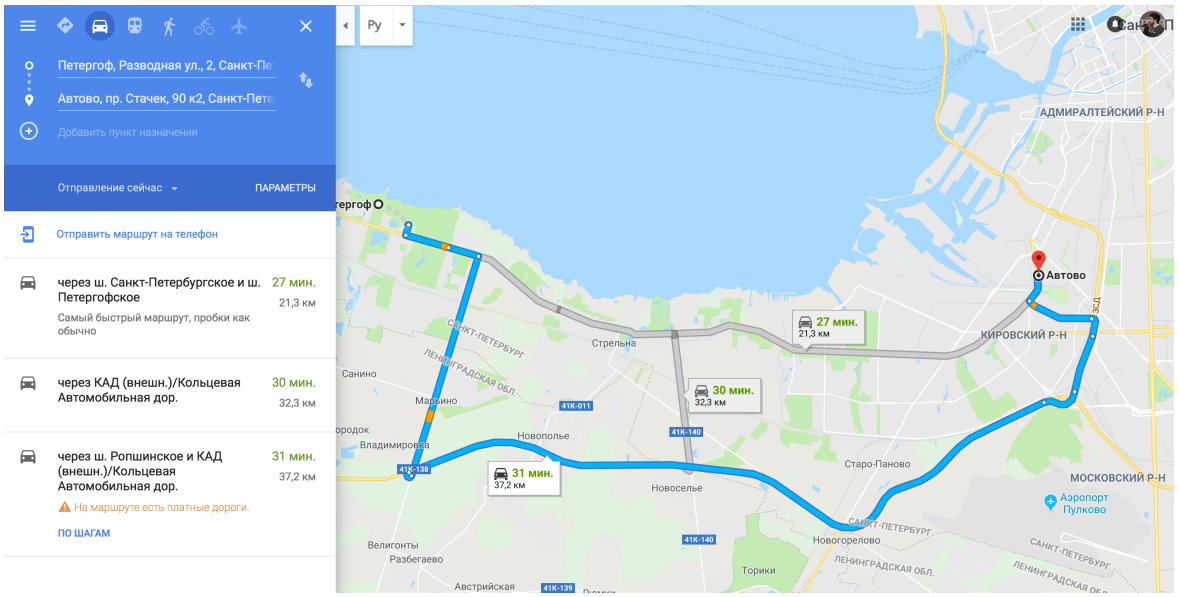

Рис. 1.3. Маршруты построенные с помощью Google Maps

- Поисково-информационная картографическая служба Яндекс.Карт [11]

  В сервисе реализован поиск по карте, прокладка маршрута, информация о пробках и панорамы улиц крупных городов. Яндекс использует только собственные карты, которые обновляет раз в две недели.

  Карты доступны в четырёх вариантах: спутниковые снимки, схемы, народная карта, спутниковые снимки с надписями и условными обозначениями (гибрид). Функционал по работе с картами доступен в полном объеме только для ограниченого числа городов.

  Сервис обладает полной информацией о загруженности дорог, так как большинство автомобилистов используют в качестве навигаторов продукт от Яндекс, и компания, получая от пользователей самую актуальную информацию, может точно информировать вас о пробках и при построении маршрута предложить менее затратный по времени маршрут. На выбор предоставляется 3 маршрута, с различными показателями по длине и времени и пользователь в праве выбрать наиболее подходящий для него маршрут. Как и в случае с картами Google Яндекс карты также имеют ограничение на количество сторонних запросов в сервис, за сутки в Яндекс можно обратиться 25000 раз, что в 10 раз превышает показатели Google, поэтому в качестве сервиса через который мы будем получать маршруты выбран Яндекс. Таким образом мы из всех предложенных маршрутов от Яндекс карт выберем самый безопасный, а через Google Maps будем получать варианты объезда наиболее опасных участков.

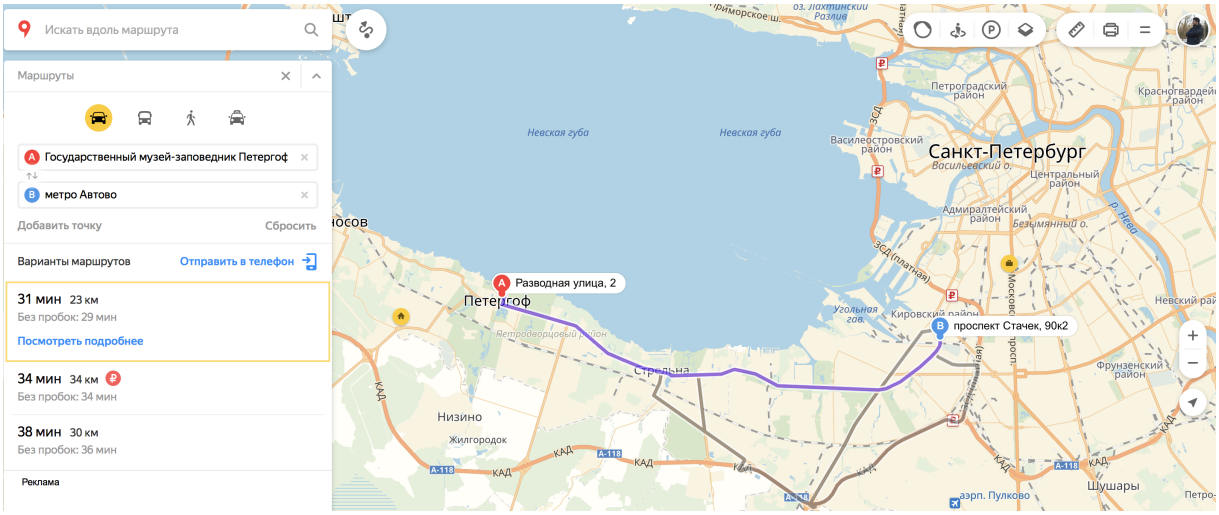

Рис. 1.3. Маршруты построенные с помощью Яндекс.Карт

Яндекс опубликовал свою технологию маршрутизации [12]. Маршрут рассчитывается по алгоритму Дейкстры. С его помощью система вычисляет самый быстрый вариант проезда — исходя из длины каждого отрезка графа и скорости движения на этом участке. Если пользователь строит маршрут проезда без учёта пробок, то алгоритм использует среднюю скорость движения на участке. А если пользователь хочет знать, как быстрее всего добраться до места с учётом ситуации на дороге, то алгоритм задействует данные о текущей ситуации на дороге.

В различных странах с безопасностью на дорогах общего пользования борятся по разному, большую популярность получила шведская программа по повышению безопасности дорожного движения и снижению смертности в ДТП - Vision Zero [13]

Базовым принципом программы является недопустимость дорожно-транспортных происшествий со смертельным исходом. Данный принцип также называют принципом «нулевой терпимости», согласно ему нельзя относиться к смертям на дороге как к неизбежному последствию, связанному с автомобилизацией.

Основной подход программы к этой проблеме призван снять с водителей основную вину за смертельные происшествия на дорогах, сделать так, чтобы в решении проблемы участвовали и те, кто строит и обслуживает дороги, производители автомобилей. Разработчики программы понимают, что водители — обычные люди и будут ошибаться всегда. Однако необходимо организовать дорожное движение таким образом, чтобы ошибки людей не приводили к смертельным исходам.

На рынке существуют различные инструменты позволяющие получить статистику по авариям за различный период в конкретном регионе, самый распространенный из них раздел статистики на официальном сайте ГИБДД [1]. На сайте представлена карта России со всеми субъектами и есть возможность посмотреть общую статистику по регионам, также полу-

чить список аварий за выбранный период по конкретному региону. Фильтр на сайте не позволяет строить сравнительную характеристику по регионам, получать статистику по смертельным исходам, видам аварий и различным комбинациям параметров.

С помощью сайта ГИБДД были получены все аварии произошедшие на территории Санкт-Петербурга в период с 2015-2017 год.

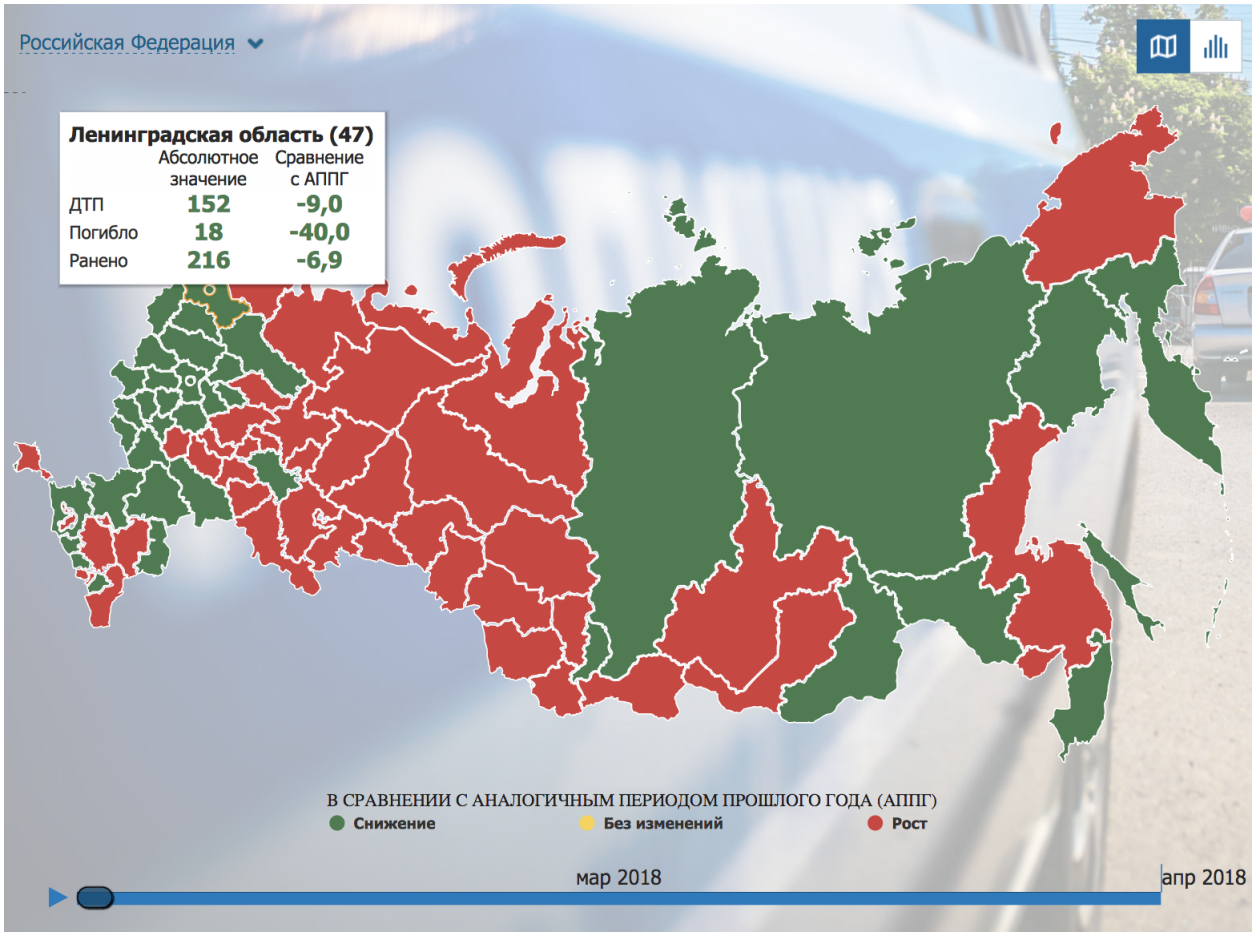

Рис. 1.4. Карта с общей статистикой ДТП

Еще один популярный сервис с возможностью получения статистики – Безопасныедороги.РФ [14]

Веб-сервис позволяет просматривать статистику в удобных и понятных графиках, задавать различные комбинации данных для отображения на графиках. Для зарегистрированных пользователей есть возможность добавить аварию или редактировать существующую, заявка на изменение или добавление отправляется администраторам сервиса.

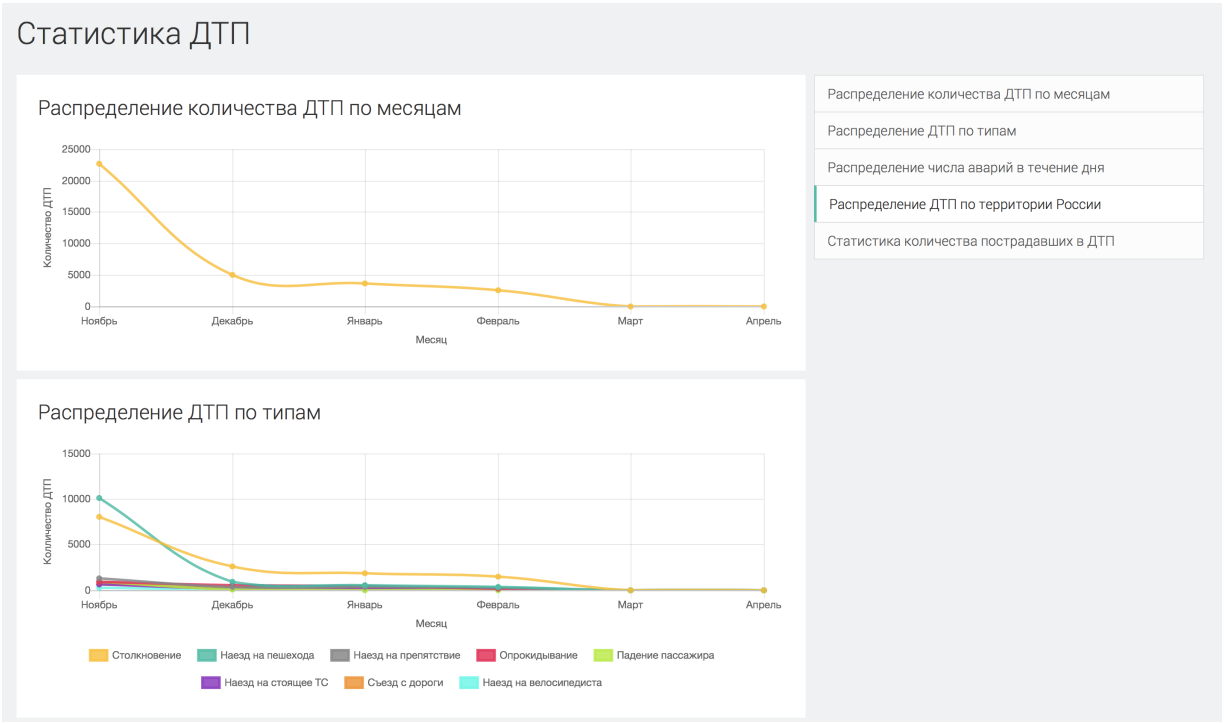

Рис. 1.5. Статистика ДТП в сервисе Безопасныедороги.РФ

Реализована возможность просмотра аварий, прикрепленных к улицам на карте, и получения информации о каждой аварии.

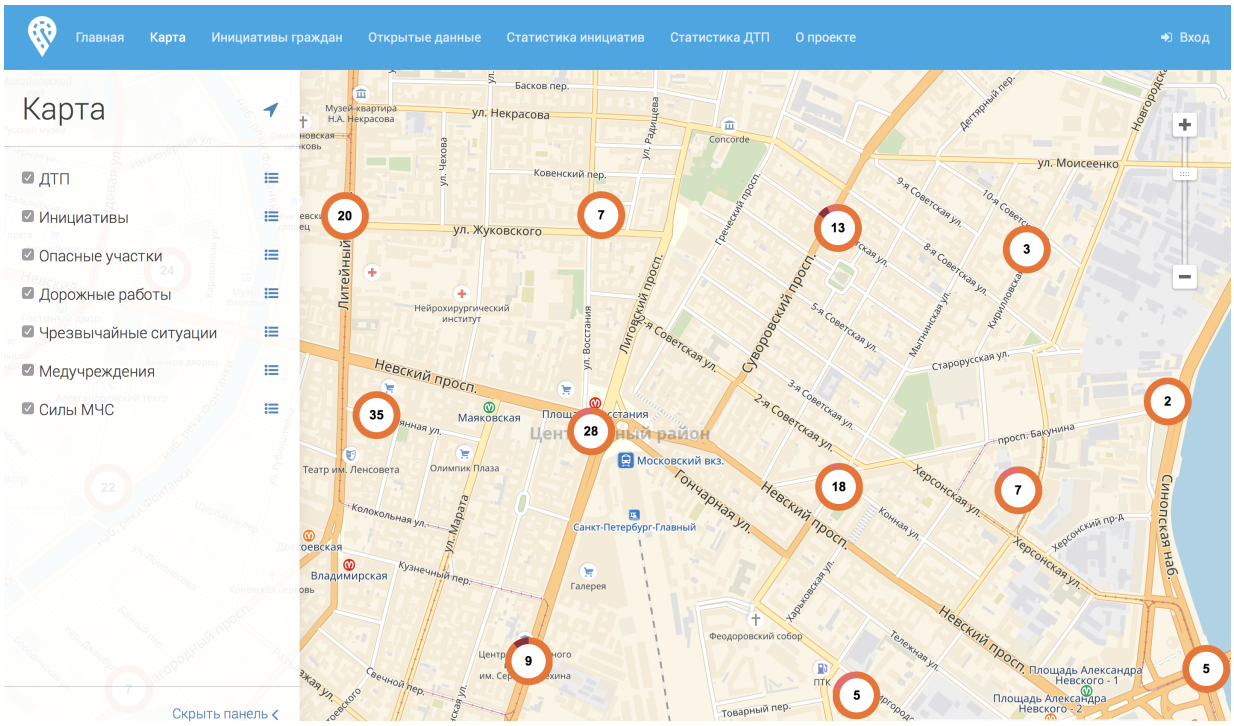

Рис. 1.6. Карта ДТП в сервисе Безопасныедороги.РФ

В ходе исследования были найдены различные методы для построения безопасного маршрута, один из них представлен в работе «SocRoutes: safe routes based on tweet sentiments» [7]. В статье предлагается новый тип маршрутной навигации, основанный на региональном контексте - прежде всего на чувствах. Система, направлена на то, чтобы найти более безопасный, дружелюбный и более приятный маршрут, основанный на чув-

ствах, исходящих из сообщений в режиме реального времени с геотегами из Twitter.

Сервис адаптирует маршруты, избегая мест с крайне негативными настроениями, тем самым потенциально обнаруживая более безопасный и более приятный маршрут с предельным увеличением общего расстояния по сравнению с самым коротким путем. Поддерживается три типа режимов движения: ходьба, езда на велосипеде и вождение. Также существует значительная корреляция между региональными настроениями, связанными с Twitter, и уровнем преступности, в частности для районов с высоким уровнем преступности и высокой отрицательной чувствительности.

Таким образом сервис используя исключительно социальное настроение в средствах массовой информации, может найти маршруты которые обходят криминальные районы.

В работе «Методика прогнозирования аварийности на регулируемом перекрестке» [6] исследуются различные модели прогнозирования аварийности на регулируемых перекрестках, для организации корректного дорожного движения, что позволяет избежать конфликтных ситуаций «транспорт – пешеход».

Взаимодействие пешеходных и транзитных транспортных потоков характеризуется высокой степенью опасности. Для того чтобы снизить число аварий и тяжесть их последствий в рассматриваемом конфликте, необходимо еще на стадии проектирования объекта и разработки мероприятия оценить предлагаемые решения, т. е. спрогнозировать аварийность. Приведены исследования специфики взаимодействия транспортных и пешеходных потоков, а также анализ формирования пространственных конфликтных точек и создания конфликтных зон в исследуемом конфликте между транспортными средствами и пешеходами на регулируемых пешеходных переходах, расположенных в зоне перекрестков. Создана методика прогнозирования аварийности по методу конфликтных зон для различных режимов движения на перекрестках. Установлены зависимости приведенной (по тяжести последствий) аварийности от потенциальной опасности конфликтов в различных режимах движения и для разных условий конфликтного взаимодействия.

Для корректной реализации метода интеллектуального анализа данных и создания общедоступного веб-сервиса были исследованы следующие работы:

- Работа "Теория вероятностей и математическая статистика" [2].

  Данная книга содержит основные разделы курса теории вероятностей: свойство вероятностной меры, формулы элементарных вероятностей, классическое определение вероятности, геометрическая вероятность, виды сходимостей случайных величин, характеристические функции

и др. В данной дипломной работе была использована теория изложенная в гл. 5 (Случайные величины).

- Работа "Математика случая: Вероятность и статистика – основные факты" [3].

  В данной книге рассматриваются вероятностно-статистические основы современных статистических методов. Изложены основные понятия, которые используются при применении статистических методов. Особое внимание уделено непараметрическим подходам, статистике нечисловых данных и другим перспективным элементам высоких статистических технологий. Для данной работы использовался материал изложенный в гл. 5 (Выборочные характеристики).

- Разработка веб-приложений [4].

  В данной книге рассказывается о создании веб-приложений. Затронуты основные проблемы, которые возникают при их создании и изложены основные технологии, которые используются при создании веб-приложений. В книге основной уклон сделан в сторону языка Python, однако, технологии и методы, которые там используются, можно перенести практически на любой язык программирования. Сервис представленный в данной работе, собран по данной книге, особое внимание было уделено разделу 3 (Понимание моделей, представлений и шаблонов), разделу 4 (Преимущества ORM) и разделу 5 (Адреса URL, механизмы HTTP и представления).

- Python и анализ данных [5]

  В работе рассматриваются вопросы переформатирования, очистки и обработки данных на Python. Подробно на примерах демонстрируется подход к разработке научных приложений, ориентированных, главным образом на обработку данных. В книги описываются различные библиотеки для языка Python, позволяющие эффективно решать аналитические задачи широкого круга.

После изучения существующих сервисов, инструментов и метод в задачах построения безопасного маршрута, можно сделать следующий вывод. На текущий момент не существует сервиса для построения безопасного маршрута с учетом последней информации о загруженности на дорогах общего пользования. Сервисы позволяющие просматривать статистику не обладают функцией прокладывания маршрута для пользователя.

Далее в работе будет описан сервис который позволит и просматривать статистику по авариям и строить безопасный маршрут, опираясь на маршруты предложенные сервисом Яндекс карт. Таким образом пользователь получит не только информацию о времени потраченном в пути и

длину маршрута, а информацию об аварийности маршрута с описанием каждого участка построенного маршрута.

Глава 2. Реализация программного комплекса для анализа безопасности маршрута

2.1. Сбор необходимых данных

Информация об авариях с привязкой к точному адрессу в открытом доступе впервые появилась в 2016 году, на официальном сайте ГИБДД. На текущий день инофрмации об аварийях стало больше и теперь доступна следующая информация: точный адрес, координаты, вид аварии (наезд на пешехода, столкновение), дата и время, количество транспортных средств с кратской информацией, количество раненых, количество погибщих, информация о дорожном покрытии и тд. Но существует большая проблема, вышеописанную информацию можно скачать в виде csv таблтицы только по одному муниципалитету. Муниципалитет — это, например, один район Санкт-Петрбурга.

Получение данных в ручном режиме трудозатратно, а точнее не подходит в рамках текущего проекта, по следующей причине. В России 85 регионов и насчитывается 2423 муниципалитета, для того чтобы получить информацию по одному муниципалитету нужно совершить 6 кликов мышкой, итого получается 14538 кликов мышкой нужно совершить для получения информации по всем муниципалитетам. И такую процедуру нужно проводить каждый раз после обновления, поэтому было принято решение научиться получать данные автоматически и в програмночитаемом виде (xml или json).

После иссследования сайта ГИБДД было выявлено, что данные на сайте получаются путем отправления на сервер POST запросов. POST — один из многих методов запроса, поддерживаемых HTTP протоколом, используемым во Всемирной паутине.

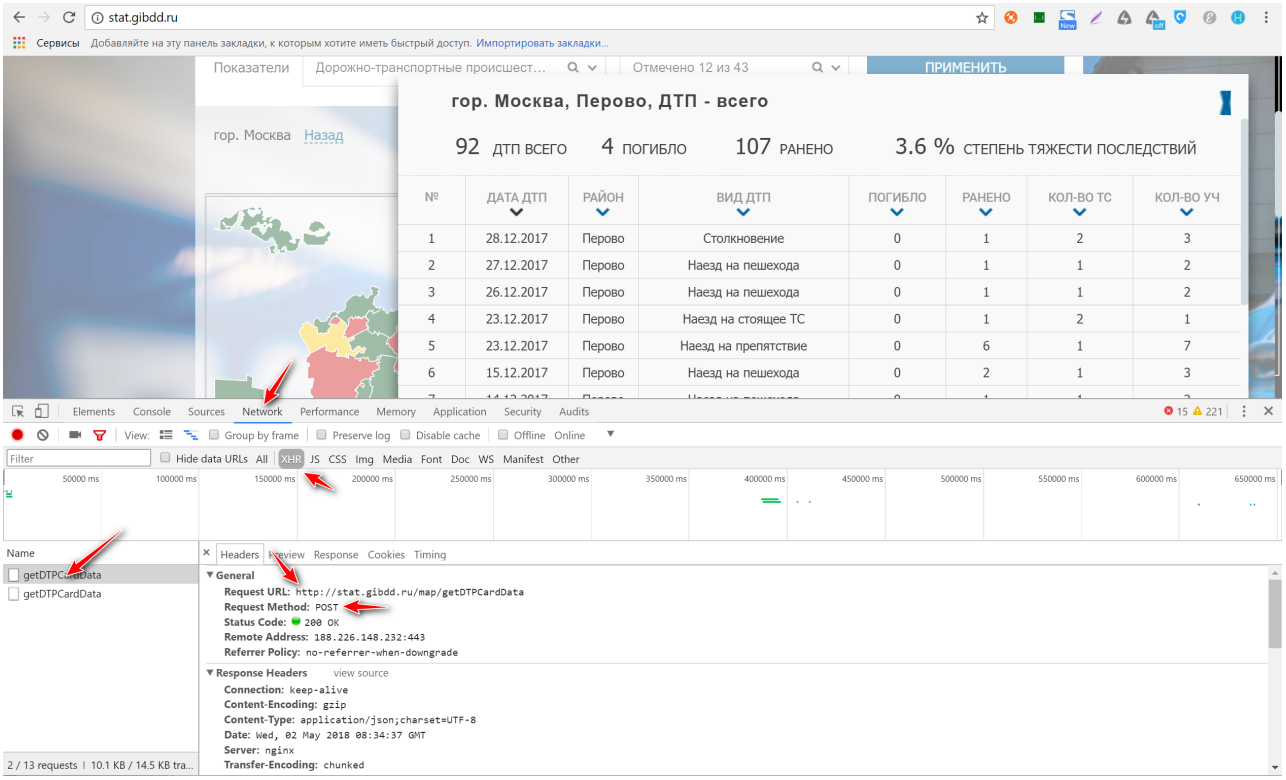

Рис. 2.1.1. Отправления запросов на сервер

На рисунке выше показано по какому адресу отправляется запрос и так же можно посмотреть параметры которые запрашиваются, в итоге получаем следующее:

- Адрес запроса для получения аварий:
  `http://stat.gibdd.ru/map/getDTPCardData`
- Заголовок запроса:

```
HEADERS = {
    'Content-Type': 'application/json; charset=UTF-8'
}
```

- Тело запроса:

```
body = {
        'date': dates, # Список дат ["MONTHS:1.2017", ...]
        'ParReg': str(region), # id региона ("41" - Санкт-Петербург ...)
        'order': {'type': '1', 'fieldName': 'dat'},
        'reg': district, # район ("45263583" - Петродворцовый район ...)
        'ind': '1',
        'st': '1',
        'en': '15000'
    }
```

В конечном итоге мы можем единаразово получить все регионы, по регионам получить районы и для каждого района в автоматизированном режиме получать данные, отправляя POST запросы на сайт ГИБДД.

Описанная процедура была проделана для города Санкт-Петербург и было получено более 19000 аварий за последние 3 года меньше чем за 10 минут, что намного быстрее чем скачивание данных в ручном режиме, также важно то, что программа сама следит за появлением новых данных, и в случае обновления аварий на сайте ГИБДД они будут автоматически загруженны в нашу базу.

2.2. Метод для оценки уровня безопасности построенного маршрута

Пусть из пункта A в пункт B существует несколько маршрутов, назовем их $K_1, K_2, K_3, ..., K_n$.

Каждый маршрут состоит из нескольких участков - отрезков (от перекрестка до перекрестка), перекрестком будем считать смену одной улицы на другую. Обозначим участки маршрута следующим образом k_{ij}:

$$K_1 = (k_{11}, k_{12}, ..., k_{1m_i}), K_2 = (k_{21}, k_{22}, ..., k_{2m_i}), ..., K_n = (k_{n1}, k_{n2}, ..., k_{nm_i}),$$

где i принимает значения от 1 до n и отвечает за выбор определенного маршрута, j принимает значения от 1 до m_i и отвечает за выбор конкретного отрезка i-го маршрута.

Обозначим за l_{ij} длину каждого участка k_{ij}, измеряемую в километрах.

Введем переменную d, которая будет обозначать день недели, d может принимать значения от 0 до 6, 0 - понедельник, 1 - вторник и т.д.

Обозначим за A^d количество аварий, которое произошло в Санкт-Петербурге в d-ый день недели. И введем переменную a_{ij}^d - количество аварий, которые произошли в d-ый день недели на j-ом отрезке i-го маршрута.

Определение 1. Уровнем аварийности на участке k_{ij} в день d будем называть частоту аварий, которая приходится на этот участок, в общем числе аварий произошедших в Санкт-Петербурге в этот день. Обозначим аварийность буквой D_{ij}^d и будем считать по следующей формуле

$$D_{ij}^d = \frac{a_{ij}^d}{A^d}.$$

Таким образом, получаем частоту аварий в d-ый день недели на j-ом отрезке i-го маршрута.

Определение 2. Под уровнем аварийности приходящейся на 1 км участка k_{ij} будем называть величину x_{ij}^d, которая считается следующим образом

$$x_{ij}^d = \frac{D_{ij}^d}{l_{ij}}.$$

Теперь для каждого маршрута можно составить выборку из случайных величин, которую будем обозначать X_i^d

$$X_i^d = (x_{i1}^d, x_{i2}^d, ..., x_{im_i}^d).$$

Определение 3. Под уровнем аварийности i-го маршрута будем называть выборочное среднее X_i^d, которое считается следующим образом

$$\bar{x}_i^d = \frac{1}{m_i} \sum_{j=1}^{m_i} x_{ij}^d.$$

Определение 4. Риском попасть в аварию будем называть выборочное среднее квадратическое отклонение, вычисленное для i-го маршрута. Обозначим как s_i^d и будем вычислять по следующей формуле

$$s_i^d = \sqrt{\frac{1}{m_i} \sum_{j=1}^{m_i} (x_{ij}^d - \bar{x}_i^d)^2}.$$

Опираясь на вычисления приведенные выше, а, именно, анализируя уровень аварийности маршрута и риск попасть в аварию, маршрут классифицируется как опасный или безопасный. Если маршрутов несколько, то пользователю предлагается наименее опасный из них.

Анализируя уровень аварийности на участке k_{ij}, выявляются самые аварийные участки дорог, и пользователю предлагается по возможности их объехать.

2.3. Обучение программного комплекса

При обучении системы для классификации маршрутов использовался следующий подход состоящий из нескольких этапов:

- Парсинг адресов через которые будут прокладываться маршруты. Самым частым маршрутом, которым ездит среднестатистический пользователь является маршрут от дома до работы и обратно. Поэтому было принято решение найти адреса бизнес центров в Санкт-Петербурге, а также адреса парикмахерских, так как большинство парикмахерских расположены в жилых домах. Данные успешно удалось спарить с портала Yell.ru. Это сервис отзывов и рекомендаций, который помогает людям искать, находить и выбирать лучшие компании, места и заведения. На сайте самые актуальные данные, так как обновление происходит ежедневно. Таким образом удалось получить свыше 600 адресов парикмахерских и бизнес центров

- Построение и получение информации о показателях аварийности каждого маршрута

  Был реализован скрипт, который по полученным адресам, прокладывает маршрут от парикмахерских до бизнес центров, получает информацию об аварийности, далее заносит ее в базу для дальнейшего анализа.

```
1   with open(os.path.join(os.path.dirname(__file__), "barbershop.json")
        ) as file_s:
2       start_list = json.loads(json.loads(file_s.read()))
3
4   with open(os.path.join(os.path.dirname(__file__), "business_center.
        json")) as file_f:
5       end_list = json.loads(json.loads(file_f.read()))
6
7   for _ in range(min(len(start_list), len(end_list))):
8       start_address = start_list.pop()
9       end_address = end_list.pop()
10
11      # Метод возвращает детальную информацию о маршруте
12      data = get_data_route(start_address, end_address, None)
13
14      r = Route()
15      r.start = data["start"]
16      r.finish = data["end"]
17      r.danger = data["danger"]
18      r.risk = data["risk"]
19      r.property = json.dumps(data)
20      r.save()
```

- Формирование обучающей выборки

  После построения маршрутов, необходимо было их разделить на опасные и безопасные, для первого деления было принято использовать метод k-средних.

  Алгоритм k-средних выполняет поиск заранее заданного количества кластеров в не- маркированном многомерном наборе данных. Достигается это с помощью простого представления о том, что такое оптимальная кластеризация.

  – «Центр кластера» — арифметическое среднее всех точек, относящихся к этому кластеру.
  – Каждая точка ближе к центру своего кластера, чем к центрам других кластеров.

  Эти два допущения составляют основу модели метода k-средних. Наш

набор данных он двумерный и его необходимо разделить на два класса. Ниже приведен скрипт который делит наши данные.

```
from sqlalchemy import create_engine
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np


engine = create_engine("postgresql://safe_route:password@94
    .250.250.220/safe_route")
conn = engine.connect()

res = conn.execute(""" SELECT "danger", "risk" FROM "route" """)
x = []
for _ in res:
    x.append(list(_))

X = np.array(x)

kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

y_kmeans = kmeans.predict(X)
```

Далее после разделения на классы, было принято решение пройтись вручную по распределенным маршрутам и скорректировать их, а так же спорные маршруты удалить из выборки, для достижения более точного деления.

В конечном итоге получилось следующее деление (Рис. 2.3.1)

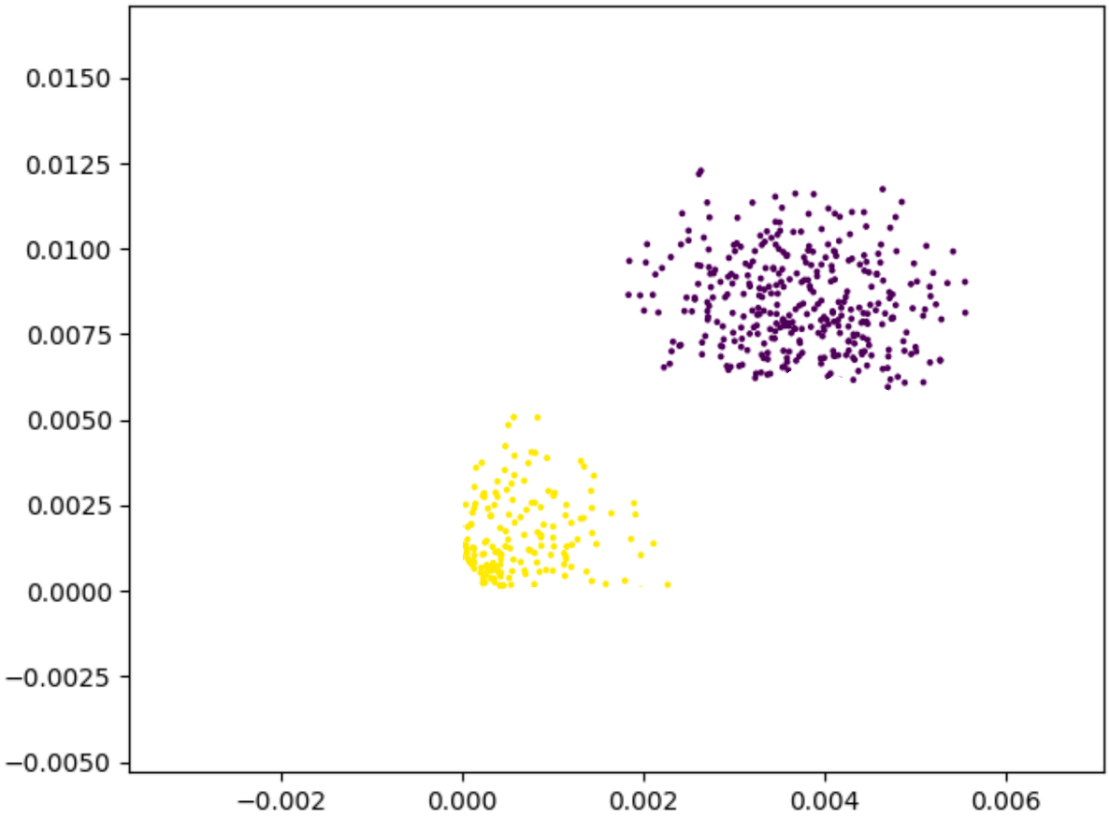

Рис. 2.3.1. Распределение маршрутов на классы

- Метод опорных векторов для класификации маршрутов

  Полученная выборка использовалась в качестве обучающей для метода опорных векторов.

  Метод опорных векторов (support vector machines, SVMs) — набор схожих алгоритмов обучения с учителем, использующихся для задач классификации и регрессионного анализа. Принадлежит семейству линейных классификаторов. Особым свойством метода опорных векторов является непрерывное уменьшение эмпирической ошибки классификации и увеличение зазора, поэтому метод также известен как метод классификатора с максимальным зазором.

  Основная идея метода — перевод исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с максимальным зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. Разделяющей гиперплоскостью будет гиперплоскость, максимизирующая расстояние до двух параллельных гиперплоскостей. Алгоритм работает в предположении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора.

  Одна из сильных сторон модели SVM - невосприимчивость к тому, как именно ведут себя удаленные точки. Они не вносят вклада в использу-

емую для обучения модели функцию потерь, так что их расположение и количество не имеет значения.

```
1
2  def plot_svc_decision_function(model, ax=None, plot_support=True):
3      """Строим график решающеи? функции для двумернои? SVC"""
4      if ax is None:
5          ax = plt.gca()
6      xlim = ax.get_xlim()
7      ylim = ax.get_ylim()
8      x = np.linspace(xlim[0], xlim[1], 30)
9      y = np.linspace(ylim[0], ylim[1], 30)
10     Y, X = np.meshgrid(y, x)
11     xy = np.vstack([X.ravel(), Y.ravel()]).T
12     P = model.decision_function(xy).reshape(X.shape)
13     ax.contour(X, Y, P, colors='k',
14     levels = [-1, 0, 1], alpha = 0.5, linestyles = ['--', '-', '--'])
15
16     if plot_support:
17         ax.scatter(model.support_vectors_[:, 0],
18                    model.support_vectors_[:, 1],
19                    s=300, linewidth=1, facecolors='none');
20     ax.set_xlim(xlim)
21     ax.set_ylim(ylim)
22
23
24 plt.scatter(X[:, 0], X[:, 1], c=y_true, s=50, cmap='autumn')
25 plot_svc_decision_function(model)
26 plt.show()
```

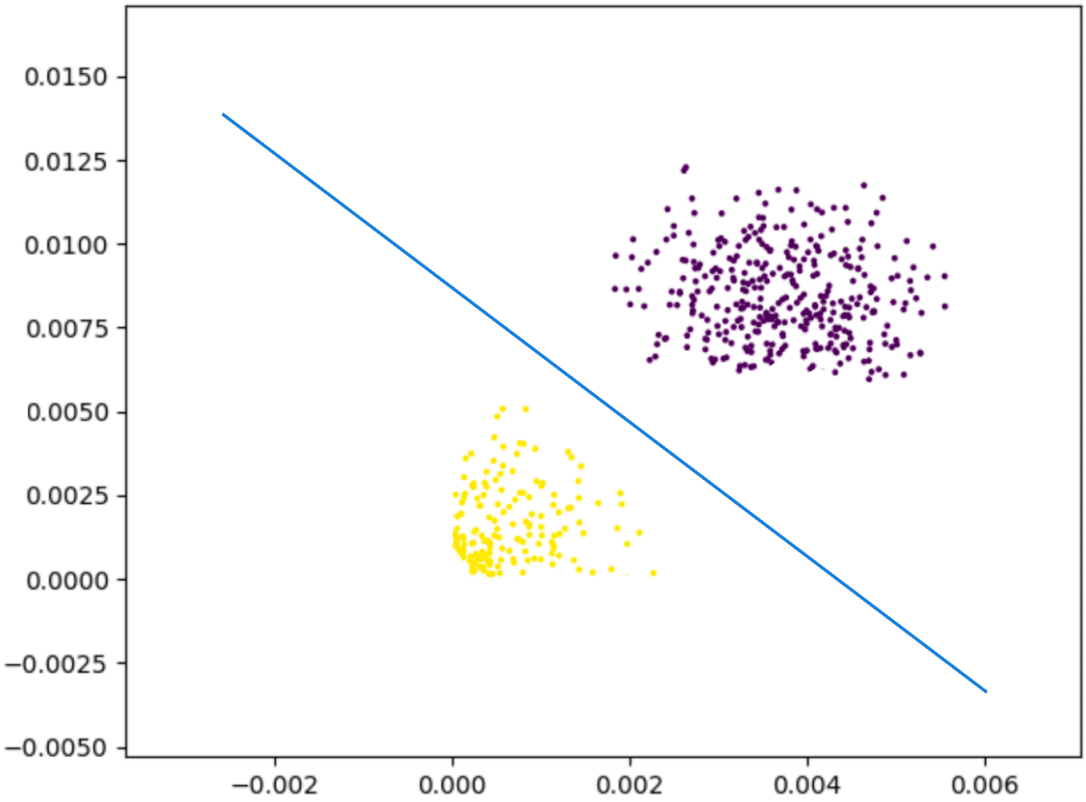

Рис. 2.3.2 Результат SVM

2.4. Архитектура программного комплекса

Пресследуя цель уменьшения трудозатрат на разработку сложного программного комплекса, было принято решение, что необходимо использовать готовые унифицированные архитекурные шаблоны. Наиболее популярные из них:

1. Шаблон MVC (Model-View-Controller) схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, модификация каждого компонента может осуществляться независимо.

    - Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние;
    - Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели;
    - Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

2. Шаблон MVP (Model-View-Presenter) шаблон проектирования, производный от MVC, который используется в основном для построения пользовательского интерфейса.

- Модель (Model) — хранит в себе всю бизнес-логику, при необходимости получает данные из хранилища;
- Вид (View) — реализует отображение данных (из Модели), обращается к Presenter за обновлениями;
- Представитель (Presenter) — реализует взаимодействие между моделью и представлением.

3. Шаблон MVVM (Model-View-ViewModel) используется для разделения модели и её представления, что необходимо для изменения их отдельно друг от друга. Например, разработчик задает логику работы с данными, а дизайнер соответственно работает с пользовательским интерфейсом

Для текущего проекта подходит класический шаблон MVC. Разберем его более подробно на примере функционала нашего сервиса.

У нас есть программа которая строит и анализирует маршрут, и есть пользователь, который хочет получить безопасный маршрут, это схемотически показано на рисунке 2.4.1

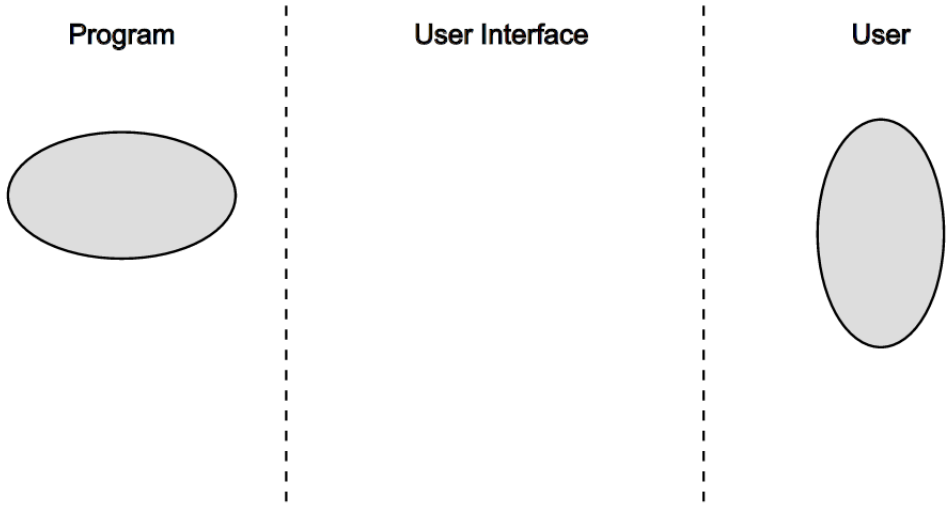

Рис. 2.4.1 Схема архитектуры сервиса

Для того, чтобы вызывались основные функции нашей программы нам нужно понимать, что конкретно хочет получить пользователь, поэтому в User interface появляется блок, который принимает решение, что именно должна сделать наша программа (Рис. 2.4.2).

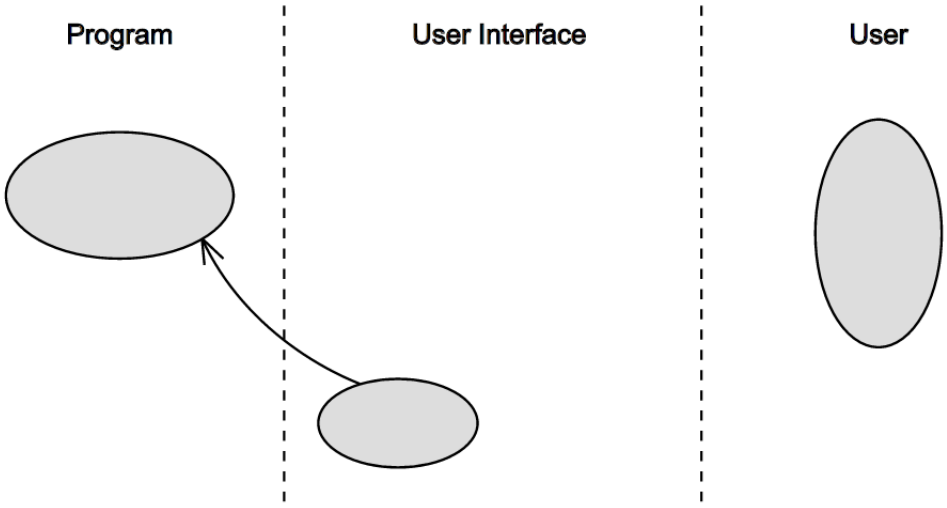

Рис. 2.4.2 Схема архитектуры сервиса

Следовательно взаиможействие с сервисом описывается следующим образом, пользователь запрашивает построение маршрута, запрос пользователя обрабатывает блок, который принимает решение, далее этот блок вызывает нужные функции основной программы и уже после пользователь видит результат своего запроса (Рис. 2.4.3).

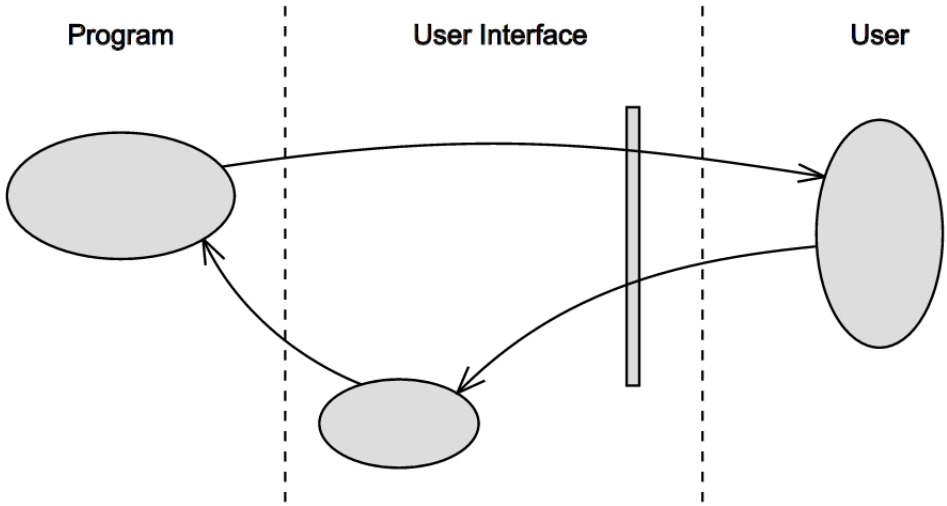

Рис. 2.4.3 Схема архитектуры сервиса

Описанная выше схема является архитиктурным шаблоном проектирования MVC. Остается только правильно расставить названия блоков (Рис. 2.4.4).

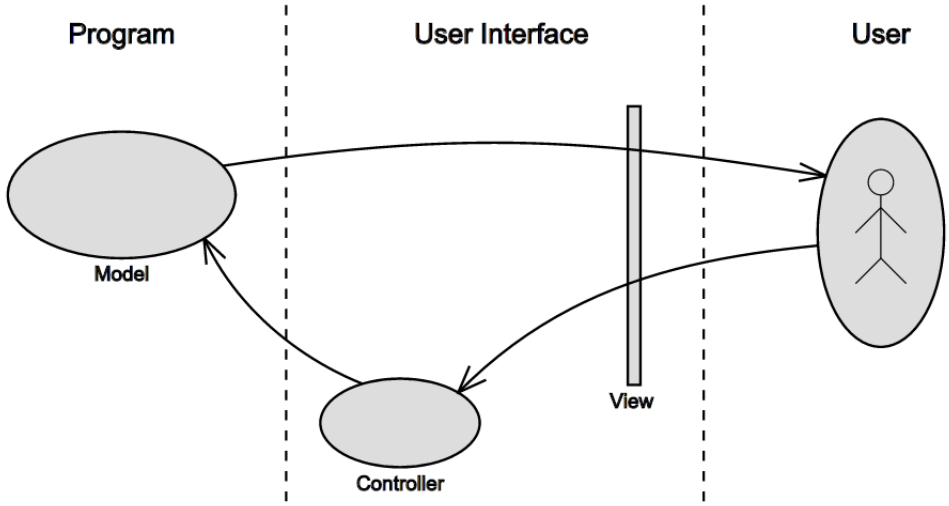

Рис. 2.4.4 Схема архитектуры сервиса

Основные разделы веб сервиса:

- Страница авторизации и регистрации (для создания личного акаунта в системе);
- Главная страница с общей статистикой по авариям в городе и информацией об аварийности последнего построенного пользователем маршрута;
- Раздел с общей статистикой, подробная статистика в графиках по авариям за выбранный промежуток времени;
- Страница с возможностью построения безопасного маршрута с подробной информацией об аварийности каждого участка;
- Раздел профиля, для сохранения маршуртов, добавления часто используемых адресов и видения статистики.

2.5. Программная реализация общедоступного веб сервиса

Для решений нашей поставленной задачи в качестве хостинга для веб- приложения был приобретен VPS - виртуальный выделенный сервер.

Виртуальный выделенный сервер эмулирует работу отдельного физического сервера. На одной машине может быть запущено множество виртуальных серверов. Помимо некоторых очевидных ограничений, каждый виртуальный сервер предоставляет полный и независимый контроль и управление, как предоставляет его обычный выделенный сервер.

Каждый виртуальный сервер имеет свои процессы, ресурсы, конфигурацию и отдельное администрирование. Обычно, в качестве виртуального сервера используются свободно распространяемые версии операционных систем UNIX и GNU/Linux. Для эмуляции обычно используются технологии виртуальных машин.

Администратор-владелец виртуального сервера может устанавливать любые приложения, работать с файлами и выполнять любые другие задачи, возможные на отдельной машине. Аренда виртуального сервера — популярный вид хостинга, так как предоставляет разумный баланс между ценой и возможностями для большинства владельцев интернет сайтов и приложений. Цена может сильно различаться в зависимости от пакета услуг поддержки и администрирования.

На VPS была установлена свободно распространяемая версия операционной системы GNU/Linux Ubuntu 14.04. Также были установлены и настроены для взаимодействия между собой следующие пакеты:

- Python 3 - Объекто-ориентированный язык програмиирования
- Django - бесплатный и свободный фреймворк для веб-приложений, написанный на Python. Фреймворк - это набор компонентов, которые помогают разрабатывать веб-сайты быстро и просто.
- PostgreSQL - свободная объектно-реляционная система управления базами данных
- Nginx - веб-сервер и почтовый прокси-сервер, работающий на Unix-подобных операционных системах

Общая схема работы программы по построению маршрута и анализу:

1. Запрос сущесвтующих маршрутов по указанным адресам. Запрашиваются маршруты у сервиса Яндекс карты и в ответ приходит от 1 до 3 маршуртов которые можно построить по этим адресам;
2. Получаем для каждого маршрута данные из нашей базы по авариям;
3. Каждый из маршрутов анализируется, и выбирается более безопасный маршрут;
4. Если все маршруты относятся к классу опасных, то на них ищутся критические участки и через сервис Google Maps получаем возможные пути их объезда и также среди них выбираем самый безопасный.

Рис. 2.5.1 Блок схема взаимодействия со сторонними сервисами

Разберем пункт с анализом маршрута более добробно, после того как мы запросили у яндекса маршруты, мы получаем маршруты поделенные

на участки от перекрестка до перекрестка с информацией об их длине и названии улицы (Рис. 2.5.2).

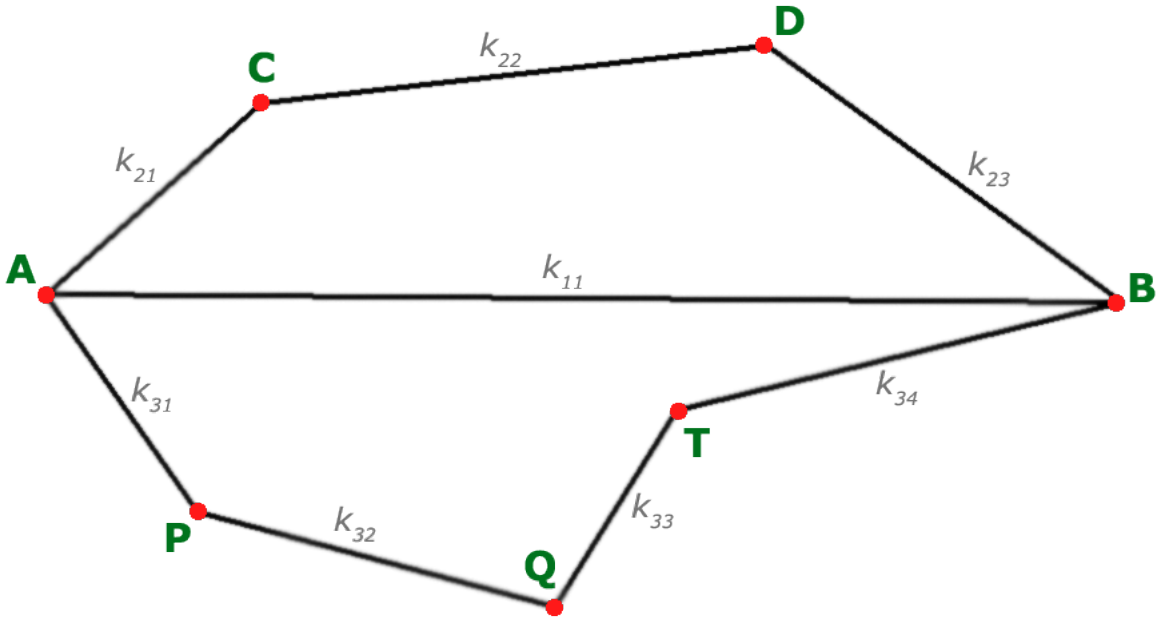

Рис. 2.5.2 Маршруты полученные от сервиса

Далее происходит следующее:

- Получаем количтво аварий в текущий день недели в этом месяце, то есть отбираем из базы аварии которые произошли в этом месяце в этот день недели;
- Из отобранных аварий получаем те которые произошли на нужной нам улице в пределах проезжающей части этой улицы;

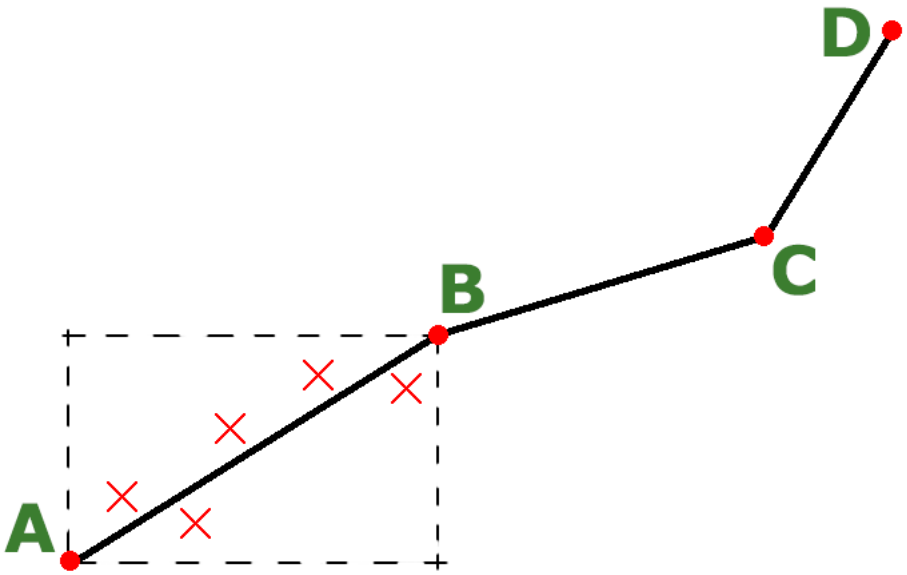

Рис. 2.5.3 Вхождение аварий в участок

- Далее считаем показатели аварийности, это аварийность на 1 км каждого отрезка и риск попасть в аварию на данноим участке;

- Получаем показатели аварийности всех ранее построенных маршрутов и добавляем к ним полученные только что показатели. Далее методом опорных векторов делим наши показатели на опасные и безопасные (Рис. 2.5.4);

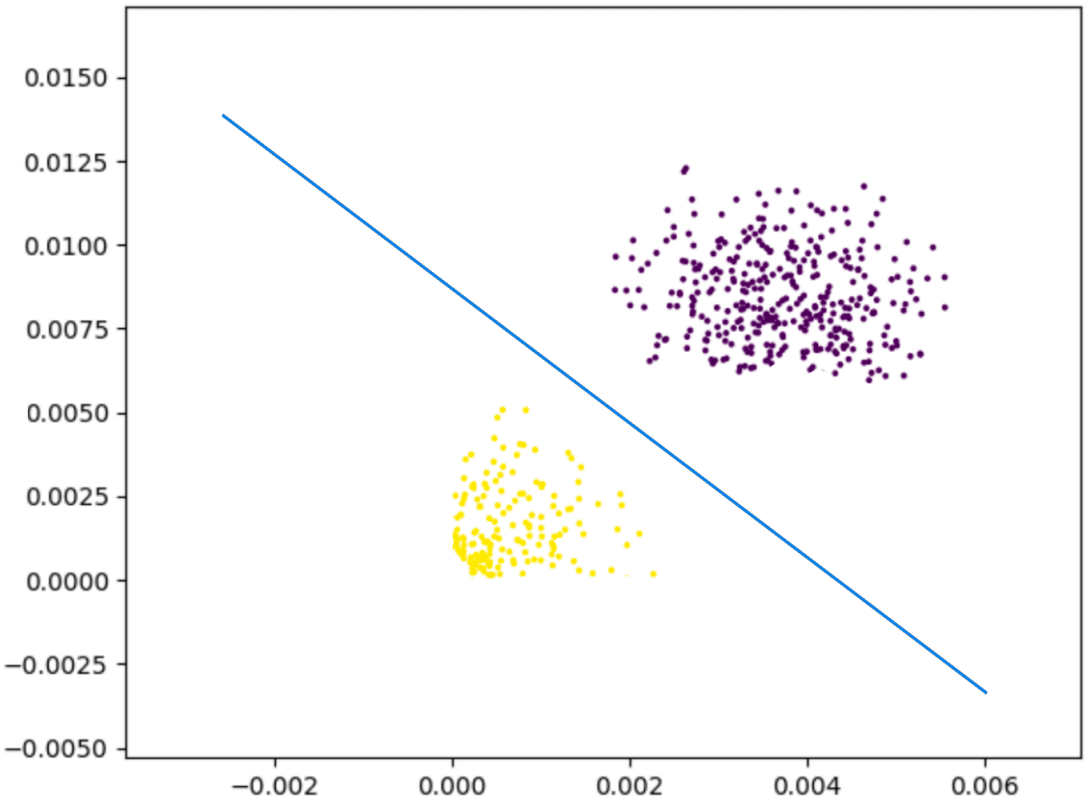

Рис. 2.5.4 Результат SVM

- Пользователю получает самый безопасный из построенных маршрутов, если все построенные маршруты были отнесены к классу опасных, то для наименее опасного из них происходит замена критических участков через Google Maps и пересчет показателей аварийности, в случае успешной замены будет выдан безопасный маршрут, к случае если сервис Google Maps не предложит вариантов или они будут аварийнее чем имеющиеся участки, то пользователю будет выдан безопасный маршрут относительно построенных яндексом, но опасный относительно имеющихся показателей аварийности.

2.6. Апробация разработанного комплекса программ

Описанный выше веб сервис работает в тестовом режиме и доступен по ip адресу `94.250.250.220`. А также разработанный метод анализа маршрута применяется в продукте СБИС компании Тензор. СБИС - сеть деловых коммуникаций и обмена электронными документами между компаниями, госорганами и обыкновенными людьми.

В продукте СБИС есть модуль управления выездными сотрудниками,

модуль работает следующим образом. Руководитель создает наряд, указывает время выполнения работ, заказчика, адрес, работы, исполнителей, материалы, запускает его в документооборот, в рабочем календаре исполнителей автоматически создаются мероприятия по наряду на указанную дату и время, далее инженер должен в указынные сроки выехать по наряду и выполнить заказ.

Маршрут, который предлагается изначально для выполнения наряда строится по описанному в этой работе алгоритму. Как это выглядит в действии продемонстрированно на рисунке ниже.

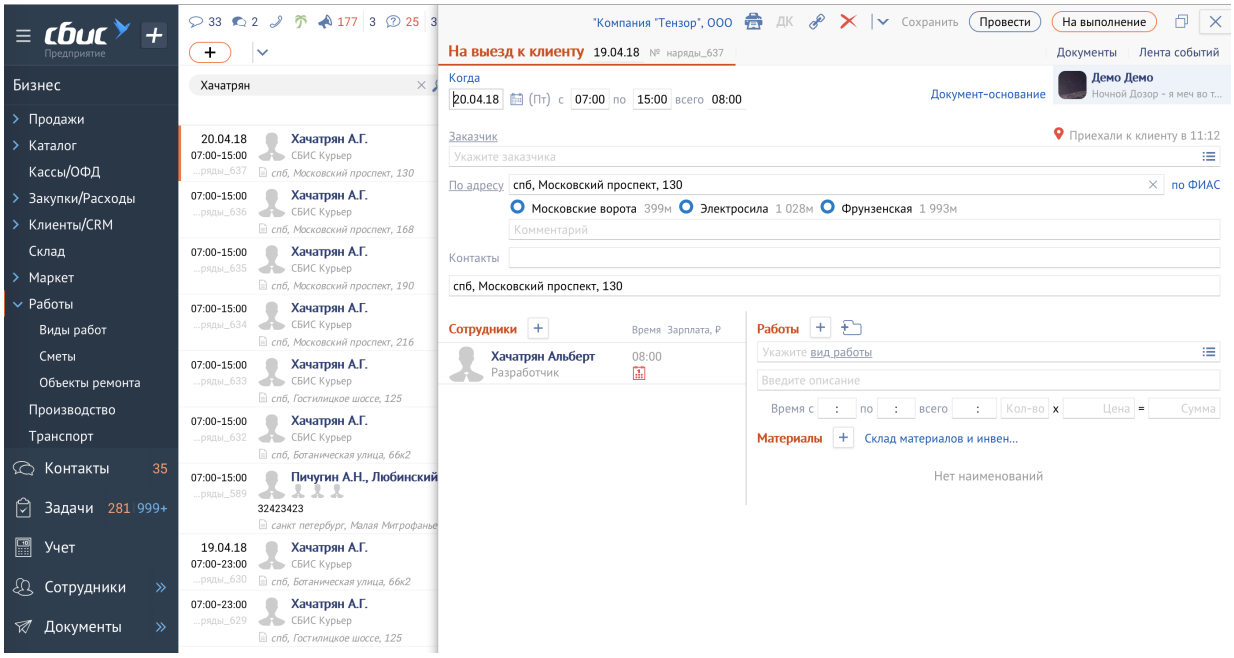

Рис. 2.6.1 Демострация модуля выездных сотрудников компании СБИС

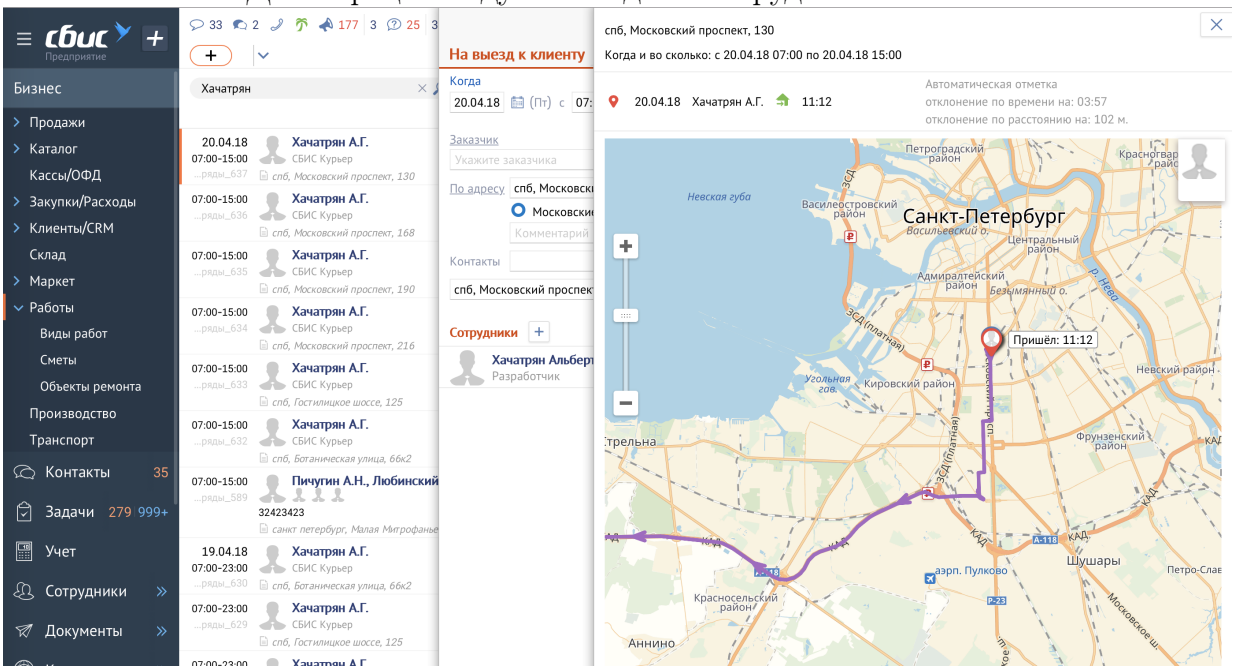

Рис. 2.6.2 Маршрут построенный для исполнителя

Заключение

Результаты

В результате проделанной работы удалось реализовать метод интеллектуального анализа данных, который позволяет оценить маршруты и классифицировать их на опасные и безопасные, а также позволяет выявить критические участки на дорогах общего пользования.

Также удалось реализовать общедоступный программный комплекс в виде веб-сервиса, который позволяет пользователям получить общую статистику об авариях по выбранным параметрам.

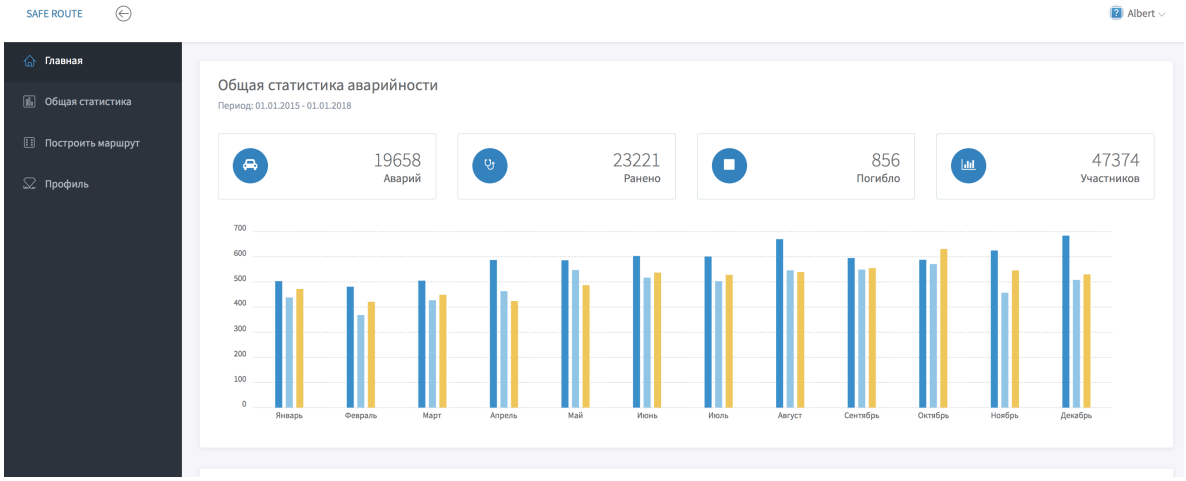

Рис. 3.1. Общая статистика аварийности

В результате интеграции метода анализа данных, реализована возможность построения безопасного маршрута для проезда пользователя по заданным адресам.

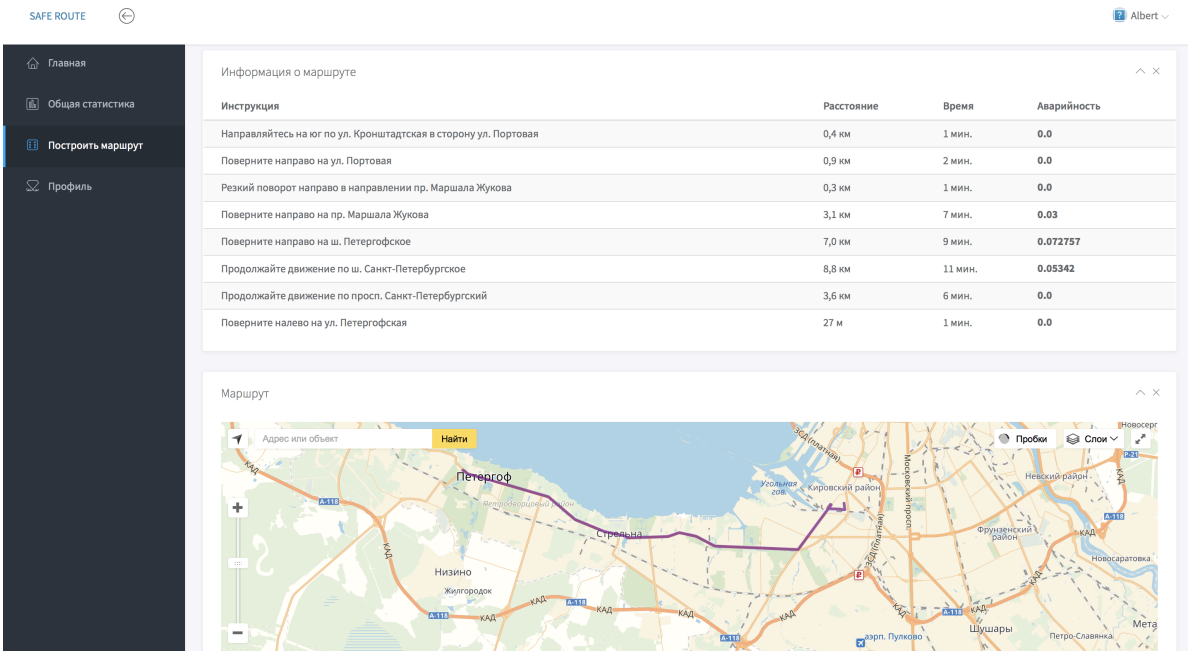

Рис. 3.2. Маршрут с информацией об аварийности каждого участка

Подключена возможность создания личного аккаунта в системе для

сохранения маршрутов и добавления часто используемых адресов в избранное.

Перспективы развития

Реализованный программный комплекс получая новые данные начинает более точно классифицировать маршруты, поэтому необходимо заняться привлечением пользователь в сервис, чтобы модель начала работать более точно.

Сервис полноценно функционирует для города Санкт-Петербург. Так как был автоматизирован процесс получения данных об аварийности по все России, то в ближайших перспективах расширить сервис для все России.

Список литературы

[1] Раздел статистики ГИБДД. http://stat.gibdd.ru

[2] Буре В. М., Парилина Е. М. Теория вероятностей и математическая статистика, издательство "Лань" , 2013. 416 с.

[3] Орлов А.И., Математика случая: Вероятность и статистика – основные факты, Учебное пособие. – М.: МЗ-Пресс, 2004. 176 с.

[4] Форсье Д., Биссекс П., Чан У., Разработка веб-приложений, издательство Символ-Плюс, 2010. 456 с.

[5] Маккинни У., Python и анализ данных, издательство ДМК Пресс, 2015. 482 с.

[6] Капский Д. В., Пегин П. А. Методика прогнозирования аварийности по методу конфликтных зон в конфликте «транзитный транспорт – пешеход» на основе моделей движения на регулируемом перекрестке // Наука и техника. – 2015. – № 5. - С. 46 - 52.

[7] Jaewoo K., Meeyoung C., Thomas S., SocRoutes: safe routes based on tweet sentiments // WWW '14 Companion Proceedings of the 23rd International Conference on World Wide Web, 2014. С. 179-182.

[8] OpenStreetMap. https://www.openstreetmap.org

[9] 2GIS. https://2gis.ru

[10] Google Maps. https://www.google.com/maps

[11] Яндекс.Карты. https://yandex.ru/maps

[12] Яндекс. Технология маршрутизации https://yandex.ru/company/technologies/routes

[13] Vision Zero. http://www1.nyc.gov/site/visionzero/index.page

[14] Безопасныедороги.РФ https://безопасныедороги.рф

Приложение

google.py

```
import urllib3
import json
from natasha import AddressExtractor

extractor = AddressExtractor()

key = ""
url = "https://maps.googleapis.com/maps/api/directions/json"


def write_point():
    return str(input("Начло маршрута: ")), str(input("Конец маршрута: "))


def get_steps(start_point, end_point):

    route = urllib3.PoolManager().request("GET",
                                          url,
                                          fields={"origin": start_point,
                                                  "destination": end_point,
                                                  "key": key,
                                                  "language": "ru"
                                                  }
                                          )
    route = json.loads(route.data)
    steps = route["routes"][0]["legs"][0]["steps"]

    return steps


def decode_polyline(point_str):
    """Decodes a polyline that has been encoded using Google's algorithm
    http://code.google.com/apis/maps/documentation/polylinealgorithm.html
    This is a generic method that returns a list of (latitude, longitude)
    tuples.
    :param point_str: Encoded polyline string.
    :type point_str: string
    :returns: List of 2-tuples where each tuple is (latitude, longitude)
    :rtype: list
    """
    # sone coordinate offset is represented by 4 to 5 binary chunks
    coord_chunks = [[]]
    for char in point_str:
        # convert each character to decimal from ascii
```

```
        value = ord(char) - 63
        # values that have a chunk following have an extra 1 on the left
        split_after = not (value & 0x20)
        value &= 0x1F
        coord_chunks[-1].append(value)
        if split_after:
            coord_chunks.append([])
    del coord_chunks[-1]
    coords = []
    for coord_chunk in coord_chunks:
        coord = 0
        for i, chunk in enumerate(coord_chunk):
            coord |= chunk << (i * 5)
            # there is a 1 on the right if the coord is negative
        if coord & 0x1:
            coord = ~coord # invert
        coord >>= 1
        coord /= 100000.0
        coords.append(coord)
    # convert the 1 dimensional list to a 2 dimensional list and offsets to
    # actual values
    points = []
    prev_x = 0
    prev_y = 0
    for i in range(0, len(coords) - 1, 2):
        if coords[i] == 0 and coords[i + 1] == 0:
            continue
        prev_x += coords[i + 1]
        prev_y += coords[i]
        # a round to 6 digits ensures that the floats are the same as when
        # they were encoded
        points.append((round(prev_x, 6), round(prev_y, 6)))

    return points


def rename_street(street):

    text = "Москва, {}, дом 7".format(street)
    matches = extractor(text)
    facts = [_.fact.as_json for _ in matches]
    res = json.loads(json.dumps(facts, indent=2, ensure_ascii=False))

    return res[0]["parts"][1]["name"]
```

region.py

```
import json
import urllib3
```

```
import datetime
from natasha import AddressExtractor


GIBDD = 'http://stat.gibdd.ru/map/'
HEADERS = {
    'Content-Type': 'application/json; charset=UTF-8'
}


class Region(object):
    path = "getMainMapData"

    def __init__(self, region_id):
        """ Инициализуирет районы выбранного региона
        :param region_id:
        """
        try:
            self.area = self.__fill_data(region_id)
        except:
            self.area = list()

    def __request(self, reg_id):
        """
        Получение районов у региона
        :param reg_id:
        :return:
        """
        url = GIBDD + self.path

        date = datetime.datetime.now().date()
        body = {
            "maptype": 1,
            "region": str(reg_id),
            "date": json.dumps(["MONTHS:{}.{}".format(date.month, date.year)
                ]),
            "pok": "1"
        }

        try:
            response = urllib3.PoolManager().request("POST", url, body=json.
                dumps(body), headers=HEADERS)
        except:
            response = None

        return response

    def __fill_data(self, region_id):
        """
```

```
50          Заполнение и сериализация полученных данных
51          :param region_id:
52          :return:
53          """
54          response = self.__request(region_id)
55
56          data = json.loads(response.data)
57          data = json.loads(data["metabase"])
58          data = json.loads(data[0]["maps"])
59
60          areas = []
61          for _ in data:
62              try:
63                  areas.append(Area(_))
64              except AssertionError:
65                  continue
66
67          return areas
68
69
70  class Area:
71
72      def __init__(self, area_info):
73          """
74          Инициализация района
75          :param area_info:
76          """
77          id_ = int(area_info.get("id", 0))
78          if id_ <= 0:
79              raise Exception
80
81          self.id = id_
82          self.name = area_info.get("name")
83          self.path = area_info.get("path")
84
85
86  def accident_data(region, dates):
87      """
88      Получение аварий по запрошеному региону и срезу
89      """
90      path = "getDTPCardData"
91
92      area = Region(region).area # Получение мунициполитетов запрошенного
            региона
93      district = 1
94      body = {
95          'date': dates, # Список дат ["MONTHS:1.2017", ...]
96          'ParReg': str(region), # id региона ("45" - город Москва, "41" -
            Санкт-Петербург ...)
```

```
        'order': {'type': '1', 'fieldName': 'dat'},
        'reg': district, # район ("45263583" - район Перово ...)
        'ind': '1',
        'st': '1',
        'en': '15000'
    }

    url = GIBDD + path
    accident = list()
    for _ in area:
        body_item = body.copy()
        body_item["reg"] = str(_.id)

        body_json = json.dumps(body_item).replace(" ", "")
        body_item = {"data": body_json}

        response = urllib3.PoolManager().request("POST", url, body=json.
            dumps(body_item), headers=HEADERS)

        data = json.loads(json.loads(response.data)["data"])
        for _ in data["tab"]:
            accident.append(_)

    return accident
```

models.py

```
import datetime

from django.db import models


district = {
    'Центральный район': 0,
    'Фрунзенский район': 0,
    'Пушкинский район': 0,
    'Петродворцовый район': 0,
    'Петроградский район': 0,
    'Невский район': 0,
    'Московский район': 0,
    'Курортный район': 0,
    'Кронштадтский район': 0,
    'Красносельский район': 0,
    'Красногвардейский район': 0,
    'Колпинский район': 0,
    'Кировский район': 0,
    'Калининский район': 0,
    'Приморский район': 0,
    'Выборгский район': 0,
```

```
    'Василеостровский район': 0,
    'Адмиралтейский район': 0
}


class Accident(models.Model):

    gibdd_id = models.IntegerField()
    date = models.DateField()
    time = models.TimeField()
    week = models.IntegerField()
    street = models.CharField(max_length=300)
    participants = models.IntegerField()
    death = models.IntegerField()
    wound = models.IntegerField()
    lat = models.FloatField()
    lon = models.FloatField()

    class Meta:
        db_table = "accident"

    def init_by_dict(self, data):
        """

        :return:
        """
        self.gibdd_id = data.get("KartId", None)
        date = datetime.datetime.strptime(data.get("date"), '%d.%m.%Y').date
            ()
        self.date = date
        self.time = datetime.datetime.strptime(data.get("Time"), "%H:%M").
            time()
        self.week = date.weekday()
        self.street = data["infoDtp"].get("street")
        self.participants = data.get("K_UCH")
        self.death = data.get("POG")
        self.wound = data.get("RAN")
        self.lat = data["infoDtp"].get("COORD_W")
        self.lon = data["infoDtp"].get("COORD_L")


class Route(models.Model):
    start = models.TextField()
    finish = models.TextField()
    danger = models.FloatField()
    risk = models.FloatField()
    property = models.TextField()

    class Meta:
```

```
70          db_table = "route"
```