САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА технологии программирования

Рубенко Виктор Эдуардович

Выпускная квалификационная работа бакалавра

Адаптивный алгоритм прогнозирования времени прибытия общественного транспорта

Направление 010400

Прикладная математика, фундаментальная информатика и программирование

Научный руководитель,
старший преподаватель
Мишенин А.Н.

Санкт-Петербург

2016

Оглавление

Введение

В наши дни общественный транспорт является одним из важнейших компонентов городской инфраструктуры. Большая часть городского населения использует различные виды общественного транспорта для своих ежедневных поездок. Так, за 2015 год только автобусный пассажиропоток в Санкт-Петербурге составил 319 миллионов человек, что является показателем для городских планировщиков о необходимости инвестирования ресурсов в создание эффективных транспортных систем. Для этого транспортные операторы и специалисты по планированию обращаются к технологиям как аппаратного, так и программного обеспечения. И цель здесь не только в создании эффективной транспортной системы, но также и в улучшении и упрощении жизни населения.

Конечно, в большинстве мегаполисов транспортные системы достаточно хорошо развиты. На самом деле, большинство операторов общественного транспорта уже сделали доступными расписания или график своих услуг для пассажиров в интернете, с помощью мобильных приложений или табло на станциях или остановках. Однако, часто случается так, что в течение дня в городе изменяется динамика машинопотока, из-за чего возможны непредвиденные задержки, и ожидающим пассажирам будут причинены неудобства, поскольку они не будут знать, где в настоящий момент находится нужным им транспорт. Данная ситуация возникает из-за использования не очень точных методов для нахождения местоположения по полученным данным из аппаратных средств, таких, как GPS/Глонасс-приёмников.

В наше время, развитие GPS/Глонасс-устройств позволяют получать отчет о местоположении с довольно высокой степенью точности, что и может быть использовано для прогнозирования времени прибытия транспорта. Такие технологии, то есть установленные GPS/Глонасс-приемники, уже

используются различными операторами для контроля и управления их парка в таких городах, как Москва, Санкт-Петербург, Калининград и т.д., что позволяет получать данные о местоположении транспорта в режиме реального времени, если информация приходит с высокой частотой. Соответственно можно обработать и проанализировать приходящие данные.

Постановка задачи

Учитывая актуальность проблемы предсказания прибытия общественного транспорта на остановки с ростом развития городов и их инфраструктуры, были поставлены следующие задачи.

Во-первых, собрать и подготовить данные для дальнейших исследований: получить GPS/Глонасс координаты транспорта, в частности 300 автобусного маршрута ГУП «Пассажиравтотранс», дорожных знаках и остановках, информацию о погоде и машинном трафике.

Во-вторых, рассмотреть несколько методов машинного обучения для построения регрессионных моделей предсказания на разных наборах данных, найти оптимальные параметры и выборку.

В-третьих, предложить адаптивный алгоритм предсказания прибытия общественного транспорта в режиме реального времени, основывая на анализе получаемых данных.

Глава 1. Подготовка входных данных

Методы машинного обучения требуют подготовки входных данных: необходимо тренировочное множество для создания своих моделей и тестовое для проверки на точность и производительность. Наборы данных содержат экземпляры или строки, которые состоят из значений различных атрибутов, то есть наборы очень похожи на таблицы баз данных. В качестве среды разработки использовался PyCharm Community Edition 5.0.5 для языка Python 2.7.

1.1 Источник данных

Большую часть данных мы будем получать с Портала общественного транспорта Санкт-Петербурга http://transport.orgp.spb.ru/, который предоставляет информацию о транспорте 24 перевозчиков, таких как СПб ГУП «Пассажиравтотранс», ГУП «Горэлектротранс», ООО «ПИТЕРАВТО» и др., в формате GTFS и GTFS-Realtime[5].

Спецификация General Transit Feed Specification (GTFS) – открытый формат данных, который содержит статическую информацию о расписании движения общественного транспорта и сопутствующий географической информации: маршруты, остановки, время прибытия.

Спецификация General Transit Feed Specification-Realtime (GTFS-Realtime) – открытый формат данных, который содержит динамическую информацию, позволяющий в режиме реального времени предоставлять транспортным компаниям информацию о текущем местоположении транспорта.

Также на движение общественного транспорта гипотетически могут влиять погодные условия. Например, в дождливую погоду количество людей на остановках, возможно, будет меньше, люди будут быстрее заходить на посадку, ещё погода оказывает влияние на состояние дорожного полотна, из-за чего возможны изменения в скорости движения транспорта. Данные о

погоде будем получать с сайта http://weather.rshu.ru/, предоставляемым Российским государственным гидрометеорологическим университетом. Данные будут включать в себя температуру, осадки, ветер и влажность.

И третьим компонентом будет информация о трафике. Если в городе происходит какое-то крупное мероприятие, то большое количество людей будет сначала стремится попасть на него, а потом вернуться домой, что вызывает пробки на дорогах, а соответственно и задержки в движении общественного транспорта. Данные о трафике будем записывать, основываясь на средней скорости транспорта на определенном промежутке дороги в определенное время в различные дни.

Вся информация для дальнейших исследований была сохранена локально, для обеспечения необходимой производительности для последующего анализа.

1.2 Предобработка данных

Данные собирались по движению 300 автобусного маршрута в период с 05.04.2016 по 03.05.2016. За день приходило около 2000 откликов о местоположении. Также были дополнительно собраны данные за несколько дней по другим маршрутам для алгоритма поиска остановок.

Маршрут был линеаризован[2], то есть разбит на сегменты, представляющие собой линии от одной контрольной точки до другой, где контрольными точками считаются места с высокой плотностью докладов транспорта о своём местоположении относительно остального пути[3]. Высокой плотностью будут обладать светофоры, знаки, повороты, перекрестки и остановки. Координаты остановок обычно известны, иначе их требуется найти самостоятельно перед линеаризацией. Длина сегменты вычисляется по следующей формуле:

$$L = \sqrt{(x_2 - x_1) + (y_2 - y_1)}, \quad (1)$$

где L – длина сегмента;

$(x_1, y_1) -$ координаты первой точки;

$(x_2, y_2) -$ координаты второй точки

После линеаризации маршрута, на каждом сегменте был получен путевой угол и средняя скорость:

$$\alpha = \begin{cases} \cos^{-1}\frac{y_2 - y_1}{L}, \text{если } x_2 \geq x_1 \\ 360° - \cos^{-1}\frac{y_2 - y_1}{L}, \text{если } x_2 < x_1 \end{cases}, \quad (2)$$

где α – путевой угол;

$(x_1, y_1) -$ координаты первой точки;

$(x_2, y_2) -$ координаты второй точки.

$$v_{c_{i_k}} = \frac{\sum_{j=1}^{n_{i_k}} v_{i_j}}{n_{i_k}}, k = 1,2, i = 1, \dots, N, \quad (3)$$

где $v_{c_{i_k}}$ – средняя скорость движения на i-ом сегменте в k-ом направлении (совпадающем путевом угле);

n_{i_k} – количество точек в i-ом сегменте в k-ом направлении;

v_{i_j} – скорость в j-ой точке i-го сегмента в k-ом направлении

$\boldsymbol{N}$ – количество сегментов.

Далее полученная информация была разбита на несколько частей по следующим критериям:

1. Длина: начиная от нескольких часов, дней, недель и заканчивая месяцем.

2. День: некоторые множества были образованы в зависимости от дня недели.

Список данных, разбитых на 10 групп, можно увидеть в Таблице 1.

ID	***Описание***	***Количество наборов***
1a	*Обучение на одном часу, тест на следующем часу.*	*30*
1b	*Обучение на двух часах, тест на следующем часу.*	*30*
1c	*Обучение на трёх часах, тест на следующем часу.*	*30*
1d	*Обучение на четырёх часах, тест на следующем часу.*	*30*
2	*Обучение на Понедельнике-Пятнице, тест на каждой дне следующей недели*	*4*
3	*Обучение на Понедельнике-Воскресенье, тест на каждой дне следующей недели*	*4*
4	*Обучение на одном дне недели, тест на таком же дне следующей недели*	*4*
5	*Обучение на двух одинаковых днях недели, тест на таком же следующей недели*	*3*
6	*Обучение на трёх одинаковых днях недели, тест на таком же следующей недели*	*2*
7	*Обучение на четырёх одинаковых днях недели, тест на таком же следующей недели*	*1*
8	*Обучение на двух неделях, тест на следующей*	*3*
9	*Обучение на трёх неделях, тест на следующей*	*2*
10	*Обучение на четырёх неделях, тест на следующей*	*1*

Таблица 1. Разбиение данных на 10 групп, 1 группа включает в себя 4 подгруппы.

1.3 Реализация методов

В данной работе предсказания будут строиться на регрессионных моделях метода ближайших соседей и метода опорных векторов.

Метод ближайших соседей (K-Nearest Neighbors) - метод, основанный на оценивании сходства между объектами.

Метод (машина) опорных векторов (Support Vector Machine) – метод, основанный на поиске разделяющей гиперплоскости с максимальным зазором для классификации.

Была использована библиотека scikit-learn (v0.17.1)[4], поскольку она проста в установке и использовании, и уже включает в себя реализацию необходимых методов. Идея заключается в нахождении зависимости времени прохождения транспортом контрольных точек маршрута от предикторов, получаемых от транспорта.

1.4 Оценка

Важно помнить, что предсказание времени прибытия транспорта создается для удобства пользователей, чтобы дать им знать, когда нужно идти или сколько стоять на остановке. Средняя абсолютная ошибка (САО) показывает среднюю абсолютную ошибку предсказания в секундах, в то время как среднеквадратическое отклонение ошибки предсказания (СОП) характеризует стабильность предсказания. Несмотря на важность низкой САО для пользователя, стабильность предсказания гораздо важнее, поскольку показывает, можно ли доверять этой системе. Таким образом, оценка системы будет представлять из себя сумму средней абсолютной ошибки и среднеквадратичного отклонения ошибки.

Глава 2. Нахождение остановок

Как уже было сказано, возможно, что данные об остановках могут отсутствовать, в таком случае придется искать координаты остановок самостоятельно.

2.1 Алгоритм

Определение 1. Точками будем называть записанное местоположение транспорта в определённый момент времени в виде (широта, долгота).

Определение 2. Мини-кластером будем называть кластер, сформированный точками, которые находятся на расстоянии 5 метров друг от друга.

Расстояние в 5 метров было выбрано эмпирически. Каждая точка принадлежит одному мини-кластеру. Такие мини-кластеры можно легко найти через матрицу расстояний.

Определение 3. $C_{макс}$ – величина, равная размеру мини-кластера, с наибольшей плотностью на пути следования транспорта.

Максимальный размер кластера находится путем сравнения количества точек в каждом мини-кластере. Мини-кластеры содержат информацию, с помощью которой можно отличить остановку от других точек. Используя мини-кластеры, мы объединяем их в кластерные множества размером 20м. Таким образом, получим определение транспортной остановки в пределах 20 метров.

Определение 4. Кластерное множество – множество в виде квадрата с координатами (x, x’, y, y’), где x’-x = 20м и y’-y = 20м.

Значение в 20 метров выбирается так же эмпирически, поскольку ошибка GPS/Глонасс устройств составляет около 20 метров.

Интуитивно понятно, что кроме остановок, нам могут попадаться множества с высокой плотностью, если они являются дорожными знаками или светофорами. Однако, эти объекты имеют иное распределение, которое мы можем использовать, чтобы отличать их от реальных остановок: водителям не всегда приходится полностью останавливаются перед знаками или светофорами в течение длительного времени, они либо останавливаются на короткие промежутки, либо просто замедляют движение. Другими словами, транспорт не останавливается перед светофором строго в определенной точке: он может остановиться на расстоянии в 40м, 20м или 5м. В то время как на остановках водители общественного транспорта обязаны останавливаться в более-менее определенной месте.

Далее, используя плотность мини-кластеров и распределения точек внутри кластерных множеств, выделим 13 классификационных признаков, и обучим несколько моделей машинного обучения для идентификации остановок.

2.2 Гистограммы распределения

Гистограммы вычисляются для каждого кластерного множества на пути следования транспорта. Гистограммы генерируются, используя распределение точек в каждом из мини-кластеров определенного кластерного множества. Гистограмма содержит 10 интервалов, каждый интервал собой часть от максимального размера кластеры $C_{макс}$.

По сути, каждый интервал гистограммы, за исключением последнего, содержит 5% приращения от $C_{макс}$. Так, первый интервал содержит количество мини-кластеров в данном кластерном множестве с количеством точек <=5% от $C_{макс}$, второй интервал содержит количество мини-кластеров данного кластерного множества с количеством точек <=10% от $C_{макс}$, третий – 15% и так далее. Последний же интервал содержит количество мини-кластеров данного кластерного множества с количеством точек > 45% от $C_{макс}$.

Для каждой гистограммы каждый столбец, показывающий частоту представлен в виде доли от общего количества точек, представленных в этой гистограмме. Таким образом, отношение между столбцами сохраняется, но их величина нормализуется. Заметим, что после нормализации сумма всех столбцов частот для каждой гистограммы равна 1.

Как было сказано, мы будем использовать 13 признаков, 10 мы уже задали выше, перейдем к оставшимся 3:

1. Для каждой точки вычисляем скорость транспорта. Поскольку точки представлены в виде пары долгота и широта, а сервер отправляет их

каждые 45 секунд (в нашем случае), мы можем найти скорость. Затем находим точки, скорость на которых меньше 2м/с, и определяем их как низкоскоростные. Подсчитываем количество таких точек в каждом кластерном множестве и вычисляем вес низкой скорости, равный отношению количества низкоскоростных точек к общему их общему числу в кластерном множестве.

2. Вычисляем среднюю скорость на кластере. Поскольку мы можем вычислить скорости точек в кластерном множестве, то можем найти и среднюю.
3. Сумма первого и второго столбца в гистограмме. Как уже оговаривалось, на знаках и светофорах наблюдается большая, но недостаточная плотность точек. Соответственно, данный признак должен позволить избегать данных объектов.

2.3 Машинное обучение

Имея данные о движения транспорта за несколько дней, мы можем построить тренировочное множество. По алгоритму, описанному ранее, мы находим 13 признаков для классификации для каждого кластерного множества, на который разбит путь движения. Классификация будет происходить на 2 класса:

1. Остановка
2. Не остановка

Основываясь на гистограммах, для обычных точек, знаков и светофоров частоты в первых двух столбцах (<=5% и 10% от $C_{макс}$) высоки, это значит, что количество точек гораздо меньше $C_{макс}$. На остановках, из-за посадки/высадки пассажиров появляется большая плотность точек. В таком случае первые столбцы гистограммы пусты или мало информативны, а остальные - активны.

Для классификации будет использоваться метод Random Forest[8], основанный на использовании комитета решающих деревьев. Реализация RF

уже есть модуле scikit-learn. Для проверки точности моделей использовали кросс-валидацию 10 раз. Обучившись на 153 точках, из которых 74 являются транспортными остановками, были получены следующие результаты, представленные в таблице 2.

	Верно	*Не верно*	*Точность*
Остановка	*59*	*15*	*79.7%*
Не остановка	*60*	*19*	*75.9%*
Среднее			*77.8%*

Таблица 2. Результат Random Forest

Таким образом, мы достигли точности почти в 78% с помощью метода Random Forest. Далее полученные остановки используются в качестве контрольных точек для линеаризации.

Глава 3. Оптимизация параметров

kNN и SVM являются методами, зависящими от данных, а поскольку не существует никакого общего правила о том, как настраивать методы для конкретной задачи – оптимальные параметры, как правило, находятся эмпирически, а не аналитически. Значит, необходимо оптимизировать параметры для наших входных данных.

3.1 Параметры регрессии k-ближайших соседей (kNN)

В реализации kNN в библиотеке scikit-learn есть четыре важных параметра: сколько использовать соседей (k), весовая функция, алгоритм, используемый для нахождения k ближайших соседей, и метрика. Алгоритмы тоже содержат свои собственные параметры, которые должны быть проверены. В scikit-learn представлено три алгоритма ближайших соседей: Brute Force, K-D Tree и BallTree[4][6].

Brute Force – самый простой алгоритм поиска соседей, который вычисляет перебором расстояния между всеми парами точек в наборе данных.

K-D Tree – идея заключается в разбиении пространства точек на бинарное дерево ограничивающих параллелепипедов, которые вложены друг в друга (иерархическая структура). Сокращение вычислений дистанций происходит на предположении, что если точка A очень далеко от B (в разных узлах), а точка C очень близко к B (в одном), то A очень далеко от C.

Ball Tree – алгоритм, похожий на K-D Tree, но узлами являются не прямоугольники, а гиперсферы, вложенные друг в друга.

Несколько тестов показали, что индивидуальные параметры для каждого алгоритмы не имеют особого значения для выборок, используемых в исследовании. Единственной настройкой, влияющая на предсказание, оказалось использование различных метрик в Brute Force. Были рассмотрены три различные метрики: Евклидова, Чебышева и Менхэттена.

Евклидова метрика: $d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$

Метрика Чебышева: $d(x, y) = \max_{i=1,\ldots,n} |x_i - y_i|$

Метрика Манхэттена: $d(x, y) = \sum_{i=1}^{n}(|x_i| - |y_i|)$

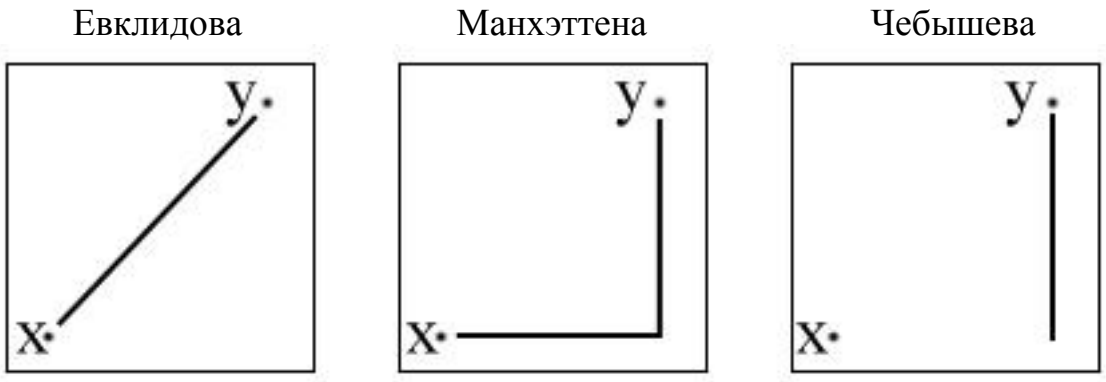

Рисунок 2. Графическое представление метрик.

Весовая функция представлена двумя видами: равномерная (единые весовые коэффициенты для каждого соседа), расстояние (обратно пропорциональная величине расстояние между точками).

Последним параметром остается величина k, которая может принимать значения от 1 до N (количество экземпляров в обучающем наборе). Значения K были выбрана как статические от 1 до 100, так и динамические начиная с N/500 и заканчивая N с шагом 10, а ближе к N шаг сокращался.

В итоге, регрессия kNN была запущен во всех рассмотренных комбинациях с различными значениями k.

Как видно из таблицы 4, наилучший результат показал kNN с алгоритмом Brute Force с метрикой Манхэттена, весовой функцией расстояние и значением k N/25.

Алгоритм	*Весовая функция*	*Значение K*	*CAO*	*СОП*	*Ошибка*
Brute Force с метрикой Манхэттена	*Расстояние*	*N/25*	*108.93*	*45.7*	*154.63*
Brute Force с метрикой Манхэттена	*Расстояние*	*N/26*	*108.89*	*45.8*	*154.69*
Brute Force с метрикой Манхэттена	*Расстояние*	*N/23*	*109.11*	*45.61*	*154.72*
Brute Force с метрикой Манхэттена	*Расстояние*	*N/31*	*109.41*	*45.31*	*154.72*
Brute Force с метрикой Манхэттена	*Расстояние*	*N/30*	*109.42*	*45.32*	*154.74*

Таблица 3. 5 лучших конфигураций kNN

3.2 Параметры регрессии опорных векторов (SVM)

Реализация SVM в библиотеке scikit-learn имеет два важных параметра: C и ядро. Наиболее часто используемыми являются такие ядра как,

полиномиальная функция, сигмоидальная и гауссовская (радиальная базисная функция RBF)[10]. Величина параметра C говорит о количестве неверно классифицированных точек, то есть дать позволить алгоритму ошибаться (штрафной параметр). Высокие значения параметра C делают классификацию более строгой, но и более затратной по времени.

Метод SVM был запушен со значением C от 0 до 10 с шагом 0.1. Тест показал, что с сигмовидным и полиномиальным ядром SVM работает медленнее, чем с RBF, а также ядро RBF показало наименьшее значение средней абсолютной ошибки.

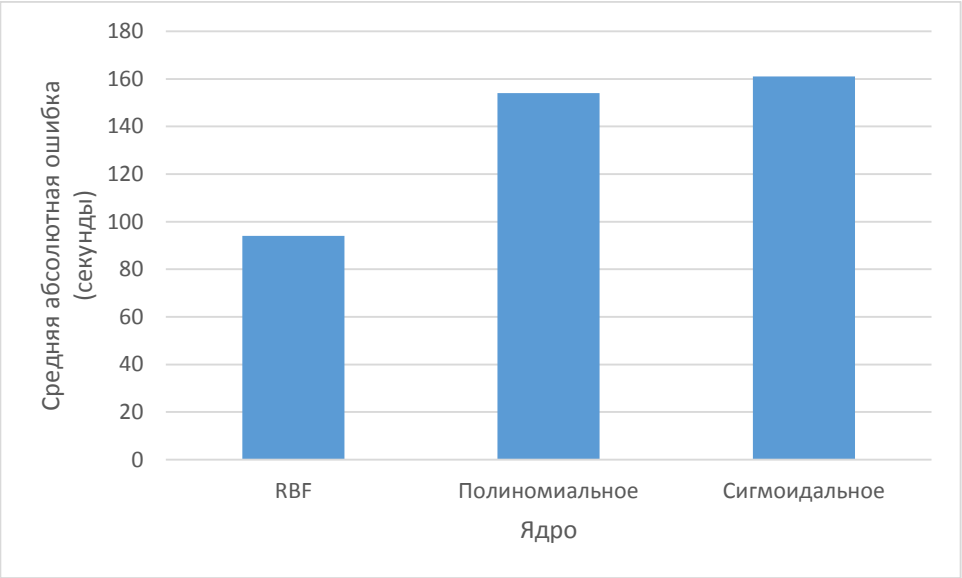

Рисунок 3. Значение средней абсолютной ошибки для каждого ядра.

Но, при использовании ядра RBF кроме параметра C значение имеет также параметр γ, которое тоже должно быть оптимизировано. Параметр γ – ширина RBF ядра, характеризует величину влияния одного примера обучения. Чем больше γ, тем выше точность регрессии для данных обучения. Правильный выбор значений параметров C и γ имеет критическое значение для производительности SVM.

Для нахождения оптимальных C и γ был используется метод ««поиск по сетке», реализованный в библиотеке scikit-learn. Алгоритм Grid Search получает данных в виде таблицы параметров, которые необходимо проверить (C, γ, ядро), и затем с помощью кросс-валидации находится лучшая комбинация C и γ. Полученные параметры: C = 5.841, γ = 0.0319

3.3 Сравнение стандартных и оптимизированных параметров

Как видно из рисунков 4 и 5, после оптимизации параметров регрессоры показали лучшие результаты на большинстве групп: kNN показал небольшое улучшение, в то время как у SVM средняя абсолютная ошибка была намного меньше и стабильнее. Стоит отметить, что параметры подбирались сразу под все данные, а не индивидуально под каждый, т.е. усреднялись, именно поэтому модели улучшили свои результаты не на всех группах. Этого достаточно для сравнения регрессоров между собой, но на практике оптимизация параметров должна быть применена индивидуально к рассматриваемому набору данных.

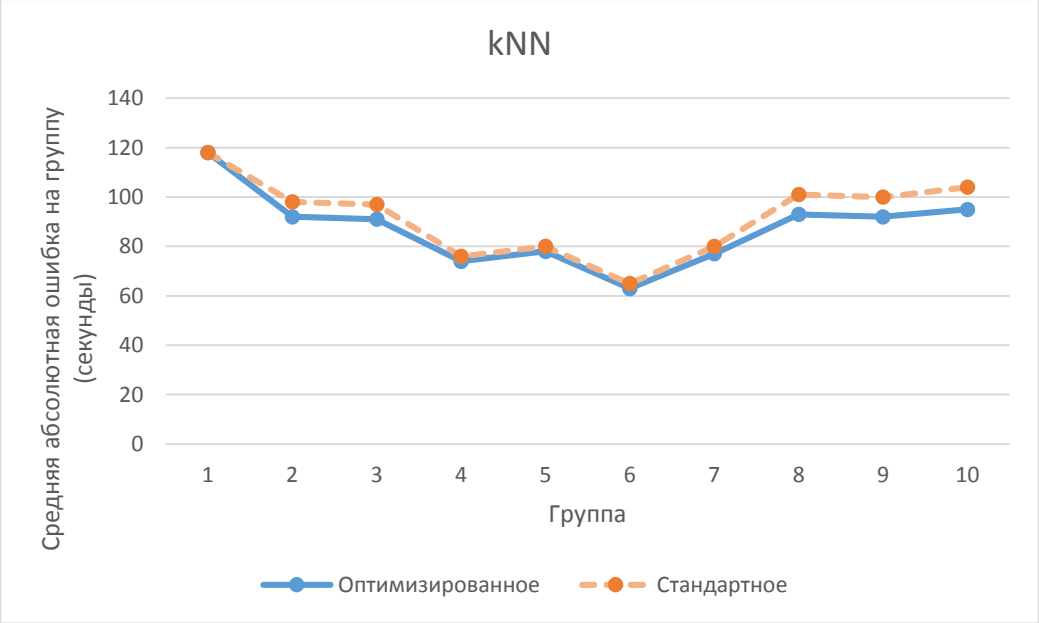

Рисунок 4. Результаты со стандартными и оптимизированными параметрами kNN по всем группам данных.

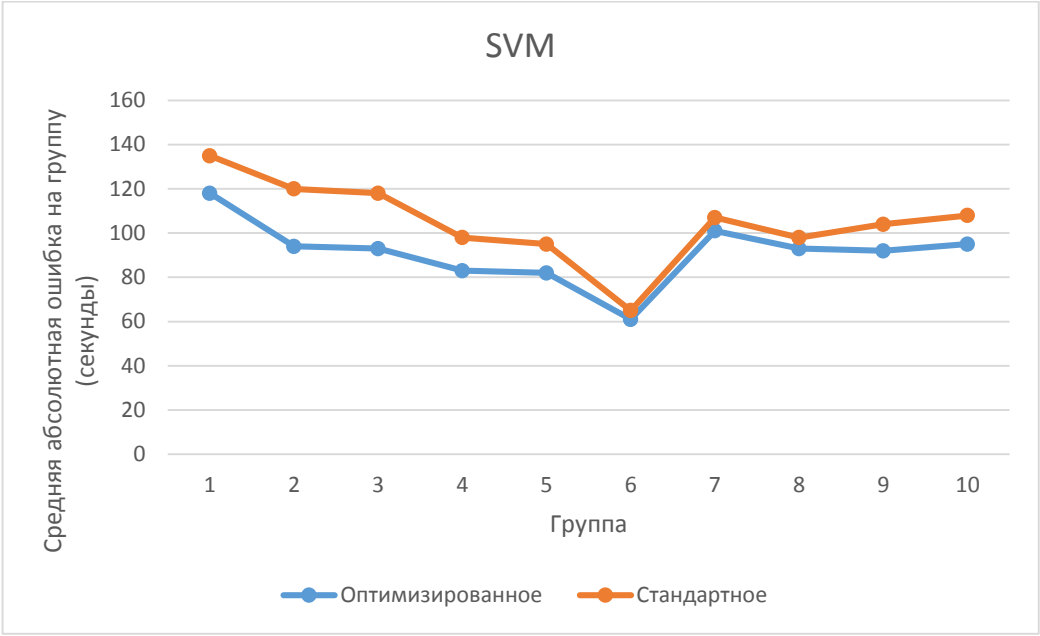

Рисунок 5. Результаты со стандартными и оптимизированными параметрами SVM по всем группам данных.

Глава 4. Анализ данных

В этой главе буду проанализированы результаты различных наборов данных, представленных Таблице 1 раздела 1.2, и выбрана оптимальная модель для предсказания. Рассматриваемые данные будут без учета информации о погоде.

4.1 Обучение на нескольких часах

Обучение на таких небольших наборах данных было сделано с целью проверки, насколько хорошо справятся алгоритмы при малом количестве информации. Результаты можно увидеть на рисунке 6. Видно, что значение абсолютной средней ошибки становилось меньше с каждым дополнительным часом в обучении, причем kNN показал результат лучше, чем SVM. Это говорит о том, что на малых наборах данных точность будет тем лучше, чем больше учебных часов, т.е. обучение на небольших отрезках времени не дает

достаточно информации для создания хорошей модели, поскольку даже один поврежденный экземпляр будет сильно сказываться на всем остальном наборе.

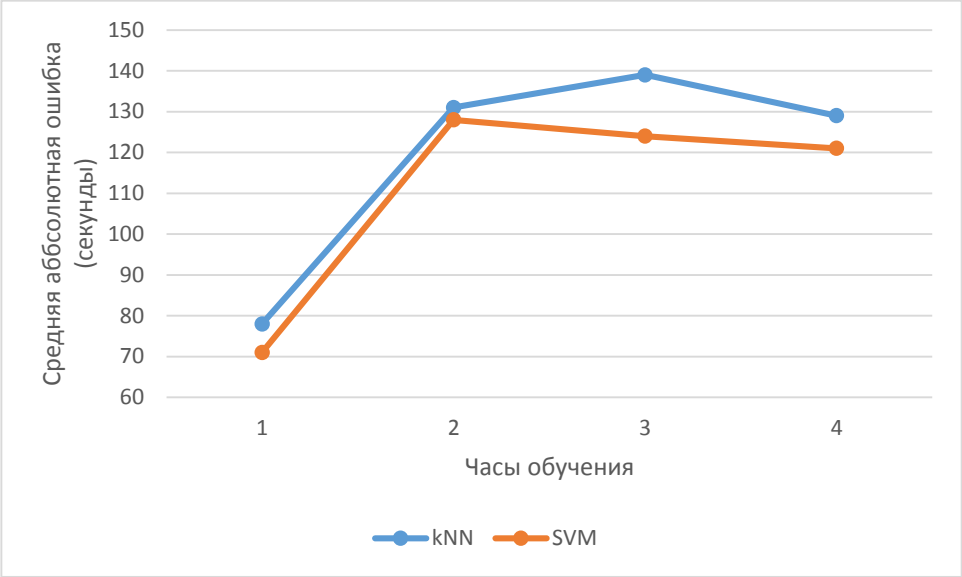

Рисунок 6. Результаты kNN и SVM при обучении на 1-4 часов и тестировании на следующем часу.

4.2 Обучение на неделе

Сначала регрессоры были протестированы на 2 группе данных: обучение на Понедельнике-Четверге и тест на Пятнице той же недели. Было это сделано для проверки предположения, что каждый день имеет свою, отличную от других, схему движения, путем обучения на четырёх днях и теста на пятом. Однако, средняя абсолютная ошибка в 92 секунды (у SVM), показывает, что движение транспорта достаточно похоже, чтобы делать предсказания между днями.

Далее, были тесты на 3 и 4 группе: обучение только на будних днях и обучение на всех днях недели соответственно и тестирование на каждом дне следующей недели. Сделано это было для того, чтобы, во-первых, проверить точность прогноза для одного дня при обучении на нескольких днях, а во-вторых, проверить влияние будних дней и выходных на структуру движения и на точность предсказания. На рисунке 7 показано, что разница в обучении

оказывает минимальное влияние на результат, т.е. обучение на 5 и на 7 днях имеет практически одну и ту же среднюю абсолютную ошибку, но всё-таки обучение без выходных даёт немного лучший результат для будних дней следующей недели, чем с выходными, это показывает, что движение в Субботу и Воскресенье изменяется. Модели SVM и kNN имеют очень схожие предсказания, кроме понедельника.

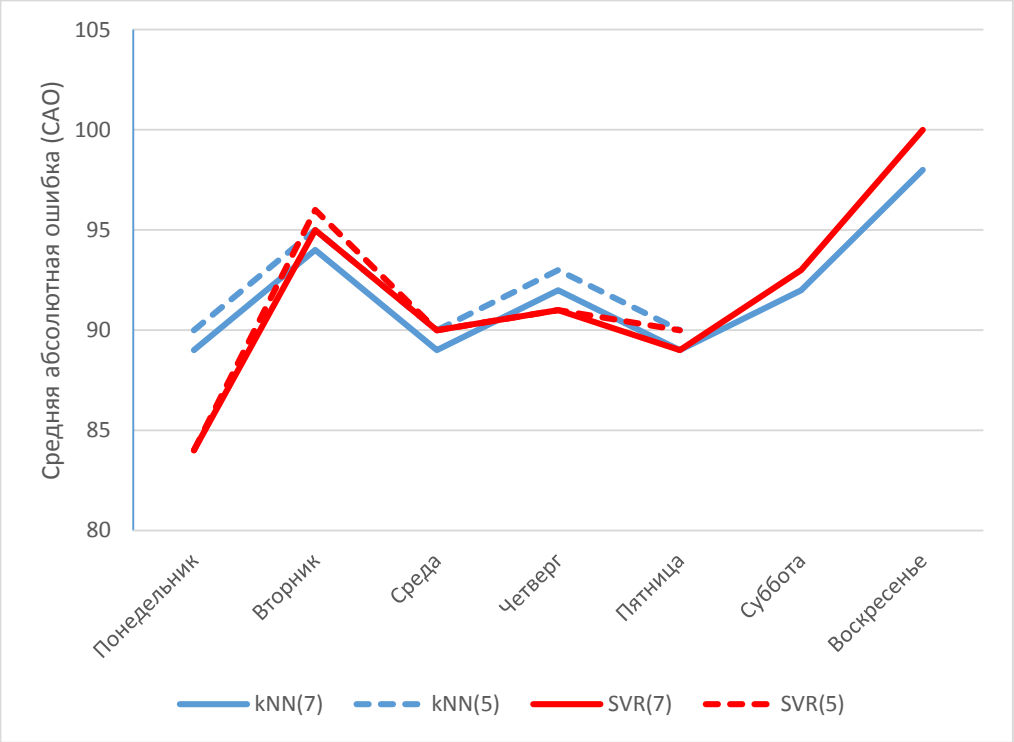

Рисунок 7. Обучение kNN и SVM на целой неделе (7) и только на буднях (5), тестирование на днях следующей неделе.

4.3 Обучение на повторяющихся днях

Данные наборы данных были рассмотрены руководствуясь предположением, что одни и те же дни обладают одинаковой структурой движения, другими словами, чтобы предсказать расписание прибытия для понедельника лучше всего обучаться только на понедельниках. Это предположение было оценено путем обучения на одном, двух, трёх и четырёх последовательных одинаковых днях и проверки на следующем: если, например, обучение было по четырём понедельникам, то проверка будет

проходить на пятом. Данная операция была произведена для каждого дня недели.

На рисунке 8 показаны результаты. Как видно из усреднённых графиков на двух и четырёх дня произошло увеличение ошибки, в то время как на трёх неожиданное понижение. Скорее всего четвертую неделю просто было легче предсказать, хотя, возможно, это говорит о том, что трёх дней достаточно, чтобы построить оптимальную модель, а затем идёт осложнение.

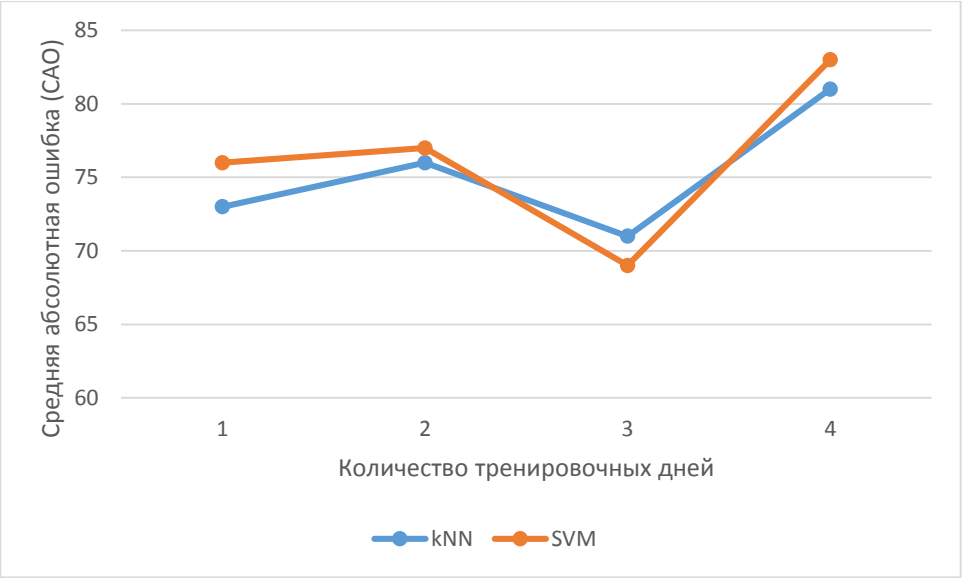

Рисунок 8. Результаты kNN и SVM при обучении на 1-4 одинаковых днях недели и тестировании на следующем дне.

4.4 Обучение на нескольких неделях

При обучении на данных наборах данных регрессоры имели как более длинный период времени, так и большее количество рассматриваемой информации, благодаря чему должна повыситься точность. Кроме того, проверим предположение, что тестирование на неделе, следующей сразу после обучения, даст более низкую среднюю абсолютную ошибку, чем на последующих.

Как видно из рисунка 9 обучение на двух неделях и тестирование на следующих после них дало лучшие результаты, чем на трёх и четырёх. Это

может быть объяснено тем, что добавление еще одной недели в обучение делало модель слишком сложной, что и повлекло увеличение ошибки. Также можно заметить, что SVM справляется немного лучше kNN на больших наборах данных. Также видно, что, как и ожидалось, с появлением временного разрыва между обучающей и тестовой выборкой, произошло увеличение ошибки.

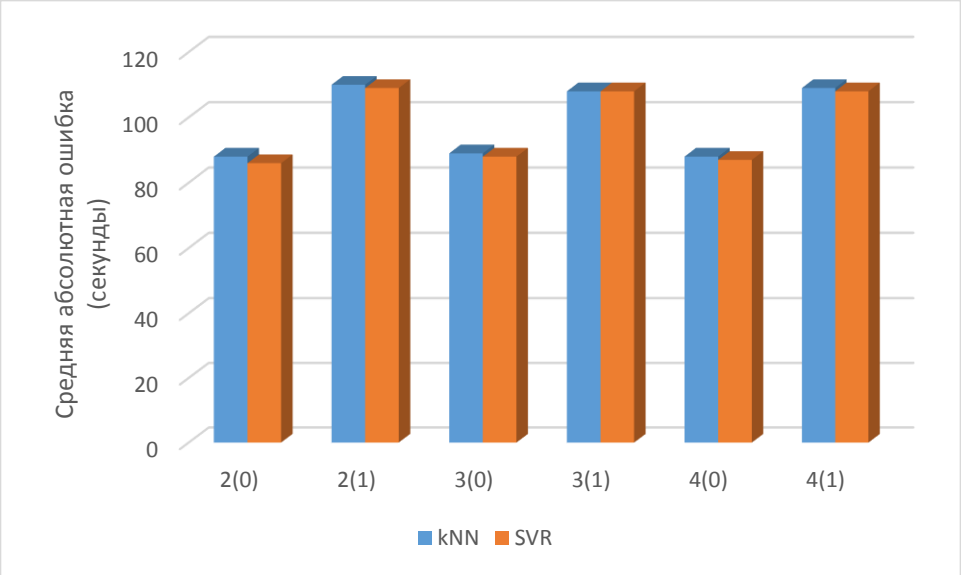

Рисунок 9. Результаты kNN и SVM после обучения на 2-4 неделях и тестировании на следующей и последующей неделе. 2(0) означает, что обучение было произведено по 2 неделям, а тестирование на наделе, следующей сразу после обучающих.

4.5 Дополнительный атрибут

Как уже было сказано в начале главы, данные в параграфах 4.1-4.4 рассматривались без информации о погоде. Теперь, получив результаты, сравним поможет ли информация о погоде сделать модель более точной. Для этого рассмотрим обучение на последовательно повторяющихся днях, поскольку на них были получены наилучшие результаты.

Результаты представлены на рисунке 10. Оказалось, что дополнительная информация о погоде никак не сказалась на предсказании SVM, в то время как kNN только ухудшил результаты.

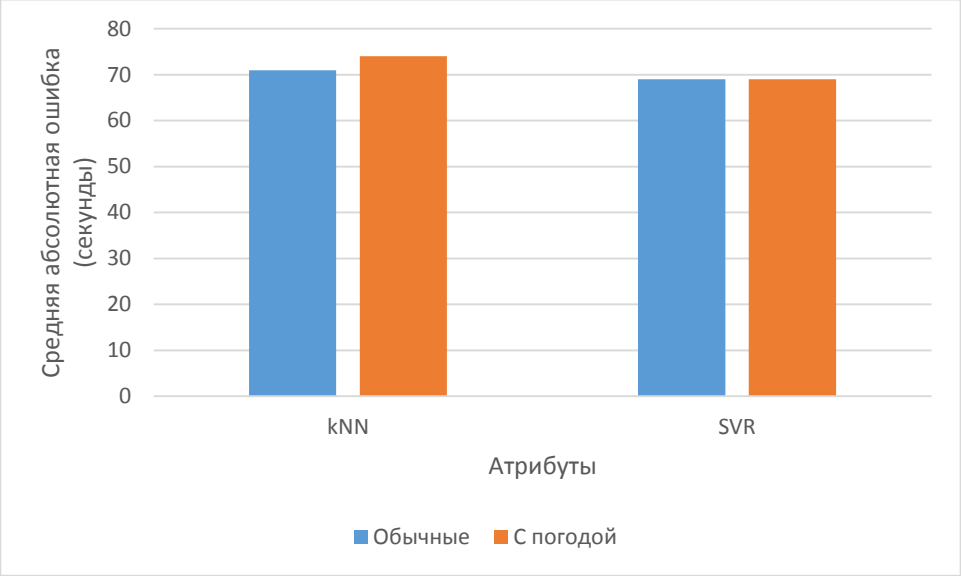

Рисунок 10. Результаты моделей kNN и SVM обученные по трём последовательно повторяющимся дням недели и протестированные на четвертом с учётом и без погодных данных.

Таким образом, необходимость в использовании погодных данных отсутствует.

4.6 Итоги

В таблице 4 представлены результаты анализа данных.

Параграф	*Обучающая выборка*	*Наименьшая средняя абсолютная ошибка*	*Метод регрессии*
4.1	*Несколько часов*	*72 секунды*	*SVM*
4.2	*Неделя*	*84 секунды*	*SVM*
4.3	*Повторяющиеся дни*	*69 секунд*	*kNN*
4.4	*Несколько недель*	*88 секунд*	*SVM*

Таблица 4. Результаты предсказания для разных групп данных.

Как видно из таблицы, разница в результате между выборками обучения не высока. Средняя абсолютная ошибка по всем данным составила 95 секунд, а среднеквадратичное отклонение 151 секунду, что является удовлетворительным для предсказания.

Стоит выделить, что наилучший результат показало обучение на выборке из последовательно повторяющихся дней недели, причем, как видно из рисунка 8, оптимальным является обучение по 3 дням: 1-2 дня дают слишком мало информации для построения оптимальной модели, а 4 – могут осложнить модель из-за изменяющейся схемы движения.

Также модели были проверены с учетом информации о погодных условиях, и, как оказалось, результаты не только не улучшились, а в случае kNN даже ухудшились. Такой результат дала выборка, состоящая из четырёх недель (3 – обучение, 1 – тест), возможно, что при рассмотрении более длительного периода времени, где изменения погоды будут более контрастными, данные могут улучшиться.

kNN и SVM алгоритмы показали схожие результаты, и когда одна из моделей показывала лучший результат средней абсолютной ошибки, другая находилась всегда рядом с ней. kNN работал примерно в пять раз быстрее, чем SVM, когда оба были с оптимизированными параметрами. Также kNN показал лучший результат со стандартными параметрами, причем разница в ошибке после оптимизации была не велика, следовательно, kNN менее зависим от оптимизации параметров, как SVM.

Глава 5. Предсказание в режиме реального времени

В предыдущих главах рассматривались различные наборы данных, средние ошибки на которых редко были ниже 80 секунд. Тем не менее, за счет учета информации в текущий момент времени, такой как время прохождения автобусом остановок, предполагаемое время прибытия на остановку может быть улучшено по мере прохождения автобусом маршрута, и получен более точный алгоритм предсказания прогнозирования общественного транспорта.

5.1 Набор данных

Поскольку в главе 4 наилучший результат предсказания удалось достичь при обучении на выборке, состоящей из трёх последовательно повторяющихся днях, этот набор данных и будет использоваться для предсказания в реальном времени. Кроме того, информация по трём дням была дополнительно проанализирована:

1. Разбита по времени отправления автобуса с начальной остановки (в часах) для каждого из двух возможных направлений движения автобуса.
2. Для каждого часа было посчитано среднее время движения между контрольными точками.
3. Найдено время среднее время начала прохождения для каждого маршрута автобуса за день, т.е. получено расписание для начальных остановок.

5.2 Алгоритм

Время прибытия транспорта на остановку будет вычислять по следующей формуле:

$$AT_i = \begin{cases} PT_{i-1} + STB_i, \text{если } i > 1 \\ \begin{cases} ST + STBi, \text{если предсказания нет} \\ PTi, \text{если предсказание есть} \end{cases}, \text{если } i = 1 \end{cases}, i = 1, \dots, N \quad (4)$$

где AT_i – время прибытия на i контрольную точку;

PT_i – предсказываемое время прибытия на i контрольную точку;

STB_i – время движения от i-1 до i контрольной точки;

ST – время начала движения по расписанию;

N – количество контрольных точек на маршруте.

Разберем формулу (4) более наглядно на рисунке 11. Пусть известно, что автобус начинает движение из точки 0 в 07:00, и пусть время движения между

каждой точкой равно 10 минутам. Тогда можно последовательно вычислить время прохождения каждой контрольной точки: 1 – 07:10, 2 – 07:20, 3 – 07:30. Как только система получает достаточную информацию с GPS/Глонасс устройства транспорта, происходит предсказание прибытия автобуса на 1 контрольную точку, например, 07:13, тогда соответственно сдвигается время прибытия на каждую следующую остановку: 2 – 07:23, 3 – 07:33. Данная процедура происходит для каждой контрольной точки, пока автобус не достигнет конечной остановки.

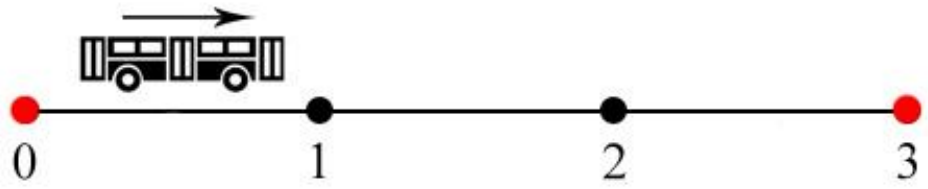

Рисунок 11. Движения автобуса по маршруту. Маршрут состоит из 4 контрольных точек 0-3, красным помечены остановки (0,3).

5.3 Эксперименты

Рассматривать будем тот же маршрут, что и в главах 3-4, поскольку у нас уже имеются найденные оптимизированные параметры. Эксперимент будет проходить на днях с 04.05.16 по 10.05.16 включительно. Соответственно для каждого дня недели были сформированы наборы данных, включающие три таких же предыдущих дня. Поскольку данные обновляются каждые 45 секунд, а для вычисления скорости необходимо минимум 2 точки, предсказание будет обновляться каждые 90 секунд.

Полученные результаты можно увидеть на рисунке 12. В виде графиков представлены результаты: средняя абсолютная ошибка для каждой остановки за неделю. Как и ожидалось, SVM показал результаты на несколько секунд лучше, чем kNN. Также удалось снизить ошибку: 58 и 60 в день для kNN и

SVM соответственно, в то время как в параграфе 4.3 результатами были 71 и 69 секунд.

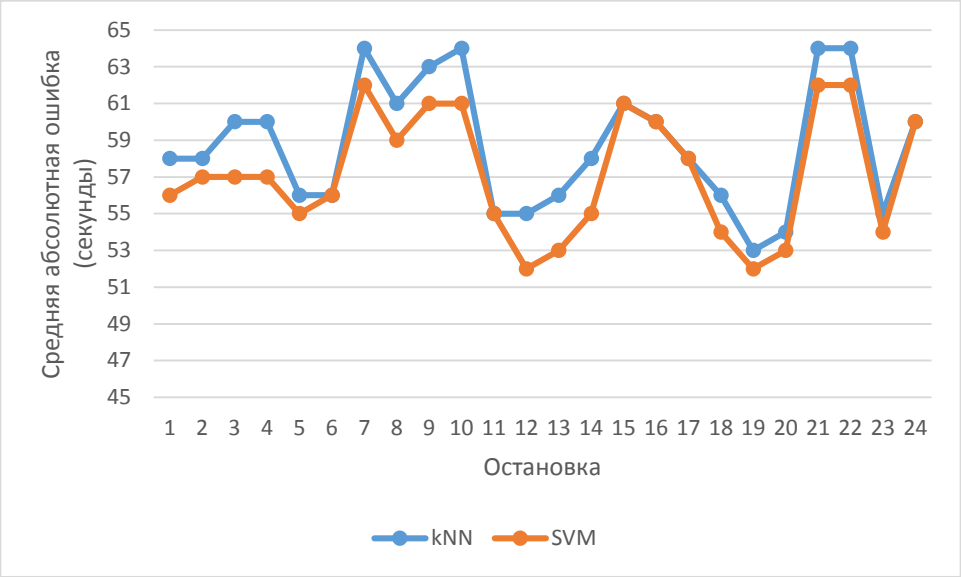

Рисунок 12. Средняя абсолютная ошибка моделей для каждой остановки за неделю.

Заключение

В настоящей работе был представлен адаптивный алгоритм прогнозирования прибытия общественного транспорта, основанный на регрессионных моделях.

Была произведена оптимизация параметров регрессионных методов для повышения точности прогнозирования. Средняя абсолютная ошибка была понижена, это показывает, что оптимизация играет важную роль в подготовке предсказания времени прибытия. Алгоритм kNN был гораздо устойчивее к настройке параметров, быстрее в работе и уступает SVM лишь несколько секунд в ошибке, следовательно, он больше подходит для предсказания в режиме реального времени в условиях постоянно меняющегося движения.

Анализ набора данных показал, что лучший способом предсказания на конкретный день является обучение на трёх предыдущих днях того же типа. Это означает, что если цель состоит в предсказании на понедельник, то регрессор должен быть обучен на трёх предыдущих понедельниках. Также было выяснено, что добавление информации о погоде не улучшает прогнозирование.

Список литературы

1. Christopher Bishop, Pattern Recognition and Machine Learning, 2006
2. Погребной В.Ю. Алгоритмизация прогнозирования времени прибытия пассажирского транспорта города Томска на остановку с использованием модели, основанной на исторических и реальных данных / В.Ю. Погребной, А.С. Фадеев // Интернет журнал «Науковедение», № 6 (19), 2013. С. 1–16
3. Wei-Hua Lin, Jian Zeng, Experimental study of real-time bus arrival time prediction with GPS data // Transportation Research Record: Journal of the Transportation Research Board, No. 1666, 1999. P. 101-109
4. Scikit-learn documentation http://scikit-learn.org/stable/documentation.html
5. GTFS documentation https://developers.google.com/transit/gtfs/
6. Nitin Bhatia, Survey of Nearest Neighbor Techniques // International Journal of Computer Science and Information Security vol. 8, No. 2, 2010. P. 302-305
7. Farhan, Ali. Bus arrival time prediction for dynamic operations control and passenger information systems, 82nd Annual Meeting of the Transportation Research Board, National Research Council, 2002.
8. Leon Stenneth, Philip S. Yu, Monitoring and mining GPS traces in transit space, SIAM International Conference on Data Mining, 2013.
9. Вьюгин В.В. Математические основы машинного обучения и прогнозирования. – М.: МЦНМО, 2013. С. 390
10. Ямшанов М.Л. Оптимизация выбора параметров SVM-классификатора с ядром RBF для задач классификации текстовых документов // Вестник ВятГГУ, 2006. №15.