Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

«Определение метрики качества для известного типа задач.»

Автор: Шведов Денис Владимирович _____
Направление подготовки (специальность): 01.03.02 Прикладная математика и информатика
Квалификация: Бакалавр
Руководитель: Фильченков А.А., канд. физ.-мат. наук _____

К защите допустить
Зав. кафедрой Васильев В.Н., докт. техн. наук, проф. _____

«___» _____ 20___ г.

Санкт-Петербург, 2017 г.

Студент Шведов Д.В. **Группа** М3436 **Кафедра** компьютерных технологий **Факультет** информационных технологий и программирования

Направленность (профиль), специализация Математические модели и алгоритмы разработки программного обеспечения

Консультанты:

а) Соколов В.В., ведущий программист ООО «ВИИРОУТЕ РНД» _____

Квалификационная работа выполнена с оценкой _____

Дата защиты «___» _____ 20___ г.

Секретарь ГЭК *Павлова О.Н.* Принято: «___» _____ 20___ г.

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

УТВЕРЖДАЮ
Зав. каф. компьютерных технологий
докт. техн. наук, проф.
_____ Васильев В.Н.
«___» _____ 20___ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Студент Шведов Д.В. **Группа** М3436 **Кафедра** компьютерных технологий **Факультет** информационных технологий и программирования **Руководитель** Фильченков Андрей Александрович, канд. физ.-мат. наук, доцент кафедры компьютерных технологий

1 Наименование темы: Определение метрики качества для известного типа задач.

Направление подготовки (специальность): 01.03.02 Прикладная математика и информатика

Направленность (профиль): Математические модели и алгоритмы разработки программного обеспечения

Квалификация: Бакалавр

2 Срок сдачи студентом законченной работы: «31» мая 2017 г.

3 Техническое задание и исходные данные к работе.

По известному множеству наборов данных необходимо научиться классифицировать поступающие новые наборы данных, чтобы понимать к какой из уже известных категорий они относятся.

4 Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов)

Пояснительная записка должна демонстрировать наиболее правильный подход к решению этой задачи, а также его плюсы и минусы по сравнению с другими методами. Должно быть произведено сравнение оптимальности ответов, времени работы, а также других метрик для всех рассмотренных способов классификации. При этом должны быть рассмотрено, какие признаки являются ключевыми и влияют на результат.

5 Перечень графического материала (с указанием обязательного материала)

Не предусмотрено

6 Исходные материалы и пособия

а) Weka Documentation;
б) VeeRoute Internal Documents;
в) Peter Harrington. Machine Learning in Action.

7 Календарный план

№№ пп.	Наименование этапов выпускной квалификационной работы	Срок выполнения этапов работы	Отметка о выполнении, подпись руков.
1	Ознакомление с предметной областью	до 31.12.2016	
2	Проработка идеи решения	до 31.01.2017	
3	Работа с исходными наборами данных	до 13.02.2017	
4	Реализация основных признаков	до 31.03.2017	
5	Проведение расчетов и сравнение результатов различных классификаторов	до 31.04.2017	
6	Написание пояснительной записки	до 31.05.2017	
7	Представление ВКР на кафедре	до 05.06.2017	

8 Дата выдачи задания: «01» сентября 2016 г.

Руководитель _____

Задание принял к исполнению _____ «01» сентября 2016 г.

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Студент: Шведов Денис Владимирович

Наименование темы работы: Определение метрики качества для известного типа задач.

Наименование организации, где выполнена работа: Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования: Научиться более оптимально и корректно решать задачу классификации новых датасетов.

2 Задачи, решаемые в работе:

а) Провести исследование описанной задачи;
б) Привести исходные наборы данных к более читаемому программному виду;
в) Подобрать максимальное количество признаков (features);
г) Внедрить различные классификаторы и выделить наиболее лучший из них;
д) Разобраться, какие признаки являются ключевыми при определении принадлежности к классу.

3 Число источников, использованных при составлении обзора: _____

4 Полное число источников, использованных в работе: 8

5 В том числе источников по годам

Отечественных			Иностранных		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет

6 Использование информационных ресурсов Internet: _____

7 Использование современных пакетов компьютерных программ и технологий: Для реализации решения задачи был использован язык программирования Java 1.8 с дополнительно установленной библиотекой Weka для обучения и тестирования различных классификаторов. Также использовался Mission Control Center для перевода файлов из формата excel в json. Еще была использована программа «Graphviz» для визуализации графа из текстового формата (dot) в графический.

8 Краткая характеристика полученных результатов: Результаты, полученные в данной работе могут быть использованы в программном обеспечении компанией VeeRoute для классификации новых поступающих наборов данных.

9 Гранты, полученные при выполнении работы: При выполнении работы грантов получено не было.
10 Наличие публикаций и выступлений на конференциях по теме работы: По теме данной работы публикаций и выступлений на конференциях нет.

Выпусник: Шведов Д.В. _____

Руководитель: Фильченков А.А. _____

«___» _____ 20___ г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

0.1. Краткое описание

В последнее время очень важной становится задача обработки новых данных. С течением времени у каждой компании появляются определенные наборы данных. Какие-то получаются на основании полученных результатов, какие-то пишут тестировщики вручную. Каждые такие наборы чаще всего можно объединить с какими-то другими, получается некое множество различных категорий. Потом компания получает какой-то новый набор данных, и она хочет узнать, к какой категории из имеющихся можно отнести эти новые данные, поскольку, скорее всего, компания уже имеет какой-то план действий, как поступать с информацией того или иного типа.

Для решения этой задачи будет использоваться машинное обучение. Из имеющихся данных, предоставленных компанией VeeRoute, будут сформированы различные признаки (features), по которым можно будет грамотно определить принадлежность к тому или иному классу. Далее следует выделить обучающуюся и тестовую выборки. На основании известных данных будет обучаться классификатор. Типов классификаторов в машинном обучении имеется достаточно множество, в приведенной работе будут использоваться некоторые с помощью библиотеки Weka для Java. Получившаяся модель будет тестироваться на новых данных, чтобы понять, насколько выбранный тип классификации оптимально определяет класс объектов.

0.2. Актуальность

Актуальность исходной задачи проявляется в необходимости многих компаний работать с новыми данными, на основании того, к какому классу из имеющихся они относятся (или не относятся ни к какому).

Разумеется эта работа будет очень полезна для компании VeeRoute, поскольку основная задача этой компании — оптимизация логистических процессов для внутригородского транспорта, будь то перевозка грузов или просто такси. Компании приходится сталкиваться с постоянным притоком новых наборов данных (datasets). Таким образом, очень важно быстро и качественно уметь классифицировать новые datasets, чтобы ускорить работу всех отраслей компании в целом.

Каждый набор данных — какая-то бизнес-задача, с которой обращаются в компанию заказчики. VeeRoute уже имеет представление, что делать с известными ей данными и какие к ним алгоритмы применять для построения или оптимизации маршрута. Поэтому, если уметь грамотно определять принадлежность новых данных к какому-то старому известному классу, то можно применять известный алгоритм. В противном случае, придется проводить дополнительные исследования, что конечно займет неопределенное время. В этом и заключается актуальность этой работы для компании VeeRoute

0.3. Новизна

Ранее написанных классификаций информации в компании не существовало. Поскольку с момента основания компании (2013 — 2014) данных стало намного больше, то сильно возросла необходимость в грамотном определении того, к чему относится тот или иной набор данных.

С точки зрения научной новизны, тяжело сказать что-то новое, поскольку никаких новых способ или алгоритмов не изобретается. Работа представляет из себя инженерную разработку. Единственное,что можно сказать — поднимается вопрос того, как корректнее определять принадлежность к тому или иному классу в условиях не очень большой тестовой выборки — основная задача машинного обучения.

0.4. Структура работы

В главе 1 представлен обзор предметной области. Говорится о подробностях нашей задачи, что она из себя представляет с инженерной точки зрения. Также подробным образом объясняется, какую именно бизнес-задачу решает данное исследование. И насколько это будет полезно для компании VeeRoute. Подробно изучена структура исходных наборов данных. Расписано, как они были преобразованы из excel в json-формат. Упоминается про группы данных, которые мы будем использовать в машинном обучении в рассматриваемой задачи.

В главе 2 представлено примерное решение исходной задачи. Указаны features, которые были использованы для обучения классификатора и их (признаков) подробное описание. Как внедряется машинное обучение, какие типы классификаторов используются и их описание. Сво-

дится ли многоклассовая классификация к бинарной и тому подобные вопросы возникающие при решении данной задачи.

В главе 3 представлены программные результаты работы на различных данных и на разных моделях. Показаны, какие признаки влияли на классификацию и какие выводы вообще можно сделать в итогам исследования.

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

В этой главе рассказывается про суть задачи, вводятся нужные определения, описываются входные данные и их типы, а также поднимается вопрос того, какую бизнес-задачу решает компания и зачем ей нужна эта работа.

1.1. Краткое описание задачи

Как уже было сказано во введении, определение типа новых наборов данных — важнейшая задача в инженерной разработке. Грамотное определение принадлежности к определенной категории многократно увеличивает скорость обработки данных в компании. В данном случае эта задача решается для компании, специализирующийся на оптимизации логистических процессов для внутригородского транспорта.

Решение этой задачи заключается в машинном обучении, а точнее в создании грамотного программного алгоритма, который и будет классифицировать наборы данных.

Для решения задачи будут использоваться данные, предоставленные компанией VeeRoute. Опираясь на них, будут сформированы признаки, по которым далее будет происходить машинное обучение. По признакам будет построен классификатор, который и будет решать поставленную задачу. Основная первоначальная цель — достигнуть хорошего результата на предоставленных данных.

1.2. Какую обширную задачу решает себе компания данным исследованием?

Теперь несколько слов о важности этой работы для компании VeeRoute. Выше уже говорилось, что это такое место, которое производит enterprise software по управлению логистики. В основном, это, конечно же, доставки различных вещей и грузов.

Очень важно понимать разницу между бухгалтерией и логистикой. Это совсем разные понятия. В бухгалтерии существуют определенные законы и регламенты, все делается, руководствуясь этими вещами. В логистики же нет ничего такого, нет никаких четких правил.

В число бизнес-процессов логистики входят:

а) планирование движения продукта;

б) доставка продукта от производителя или поставщика;
в) ведение складского учета полученного груза ;
г) доставка товара в магазины;
д) контроль за движением товарных потоков.

Следует также понимать,что бизнес-процессы логистики в любой крупной бизнеса всегда уникальны. В каждой компании все происходит абсолютно по-разному. Так как в любой организации есть уникальные особенности бизнес-процесса , то нет какого-то единого универсального бизнес-решения. Это значит, что наборы данных, по факту, у каждой компании могут быть полностью не похожими на другие. То есть бизнес-логика, решение и требование к нему, данные у каждой крупной организации абсолютны уникальны. Это приводит к тому, что и входные наборы данных (и их формат) у каждой компании свои собственные, исключительные. Сказанное выше приводит к тому, что перед тем, как запустить поиск какого-то решения на клиентском датасете, необходимо валидировать эти данные на их корректность, поскольку нету единого стандарта.

В чем состоит исходная задача. Уже имеются известные наборы решений на датасеты, которые тоже уже известны. Появляется новый клиент со своими собственными данными. Компании нужно понять:

а) валидные ли данные или нет;
б) есть ли уже похожие данные, под которые есть готовые решения.

Под вторым пунктом подразумевается готовый набор конфигурационных файлов, учитывающих специфику бизнес-процессов, и особенность входных данных. Суть в том, что, имея такой алгоритм, при добавлении изначального датасета каждого нового клиента и configuration файла, можно сделать самообучающуюся систему, которая в зависимости от того, какой пришел новый клиент:

а) проверит валидность данных;
б) подберет похожий набор конфигурационных файлов, который уже решает такую задачу.

На выходе получается очень важная бизнес-задача. В данный момент она все-таки чуть менее привлекают компанию VeeRoute. Потому что это достаточно трудоемкая бизнес-задача и данных пока не так много.

Пока еще это все можно сделать вручную, но по мере роста компании вопрос станет все более и более актуальным. Чем больше будет приток клиентов, тем тяжелее компании будет с этим справляться без специального софта.

Конечно, можно сказать, что всегда в логистических организациях решается задача коммивояжера или ее производная, но можно привести вот такие примеры (специфики):

а) у одной компании доставка по большому городу (например, Москва) с наличием только одного склада,а у другой шесть таких мест;

б) у одной компании только клиентская доставка, а в другой есть еще и забор (Pickup and Delivery). В третьей компании, при наличии определенного товара, нужно автоматически добавлять новую точку в маршруте.

В обоих случаях это абсолютно разные бизнес-задачи и решаются они диаметрально противоположно. Каждая бизнес-логика принципиально влияет на набор датасетов и конфигурационный файл на платформе.

1.3. Классы бизнес-задач

Согласно API (данные) планировщик может получить для решения очень сложную комбинаторную задачу. Поскольку сложные задачи требуют сложных подходов, сложную модель вычислений, трепетное отношение к качеству результата и очень сложное пространство решений, во многих случаях расчет без дополнительных предположений будет требовать намного больше времени и вдобавок качество решения будет хуже, чем если бы эти предположения были введены. Поскольку кроме формальной модели, представленной в API, есть еще и человеческие ожидания, а вследствие этого и неявные предположения, в планировщике используется понятия класса задачи, решающее проблему формализации человеческих ожиданий.

Класс задачи есть система дополнительных ограничений, которые чисто формально следуют из постановки задачи (например, что исполнитель не может выполнить более 100 заказов за рабочую смену исходя из продолжительности заказов). Планировщик при старте расчета определяет класс задачи и в дальнейшем использует его в процессе расчета

для различных оптимизаций (качество, скорость, разные наборы решений для разных классов задач, сам процесс расчета может отличаться для разных классов задач). В то же время это усложняет тестирование планировщика, поскольку на разных классах задач его поведение может существенно отличаться.

Система дополнительных ограничений состоит из конкретных, достаточно простых свойств, которые должны быть понятны человеку и в большинстве случаев легко диагностируемы «на глаз».

При старте расчета планировщик выводит в лог подробную информацию обо всех свойствах, а также определенный на основании свойств тип задачи.

Имеются следующие классы задач:

— Delivery (развозка с одного склада)
— Multiple Depots (развозка с нескольких складов)
— Service Engineers (сервисные инженеры / обслуживание объектов)
— Pickup and Delivery

1.3.1. Pickup and Delivery

Для задачи типа «Pickup and Delivery» нет каких-либо специфичных свойств. Это самый общий класс задач, любая задача по умолчанию относится к этому классу.

1.3.2. Service Engineers

Этот тип задачи требует следующего набора свойств:

а) Простые приоритеты
   1) haveOnlySimplePrecedences

      Отсутствие сложных приоритетов (haveOnlySimplePrecedences) — у работы (Work) приоритет внутри рейса равен 0, у забора (Pickup) равен 1, у доставки (Drop) равен 2. Приоритеты внутри заказа для всех заявок равны 0. Соответствует тому, что в рамках любого маршрута все заборы будут раньше всех доставок (на работы ограничений нет).

б) Каждый заказ есть ровно одна заявка типа Work
   1) haveOnlyWorkDemands

Все заявки только типа Work

2) haveNoConsolidatedDemands
   Ровно одна заявка в любом заказе
3) haveNoMultiEvents
   Наличие ровно одного события в рамках одной любой заявки.

в) Все заявки достаточно продолжительны по времени
   1) allEventsAreLongLasting
      Продолжительность любого события составляет не менее 10% времени любой смены, в которой он может быть выполнен.

1.3.3. Multiple Depots

Этот тип задачи требует следующего набора свойств:

а) Простые приоритеты
   1) haveOnlySimplePrecedences
   2) mustAllPickupsBeBeforeDrops
      Любой забор раньше любой доставки. Следует из свойства haveOnlySimplePrecedences, но может выполняться и в других случаях.

б) Все заказы имеют только одну простую заявку на забор и развозку одного груза
   1) haveOnlyWorkDemands
   2) haveNoConsolidatedDemands
   3) haveNoMultiEvents
   4) doAllPickupsHaveExactlyOneCargo
      Во всех заборах фигурирует ровно один груз.

в) Нет сложных временных ограничений
   1) doAllPickupTimeWindowsCoincide
      У всех событий всех заборов мягкие (Soft) и жесткие (Hard) временные окна совпадают между собой и одни и те же у всех заборов (например, с 6 до 20).
   2) haveNoComplicatedLocationTimeWindows
      Для каждой заявки есть временное окно работы локации, в рамках которого заявка целиком выполнима.
   3) doAllTransportsHaveSameShifts

У всех транспортов одинаковые окна смен. Если взять два разных транспорта, то смен у них поровну и в соответствующих сменах временные окна смен у них совпадают.

4) doAllPerformersHaveSameWorkShifts
   У всех исполнителей одинаковые рабочие смены. Если взять два разных исполнителя, то смен у них поровну и в соответствующих сменах временные окна смен у них совпадают.

г) А также у всех транспортов ровно один отсек

1) doAllTransportsHaveExactlyOneBox
   У всех транспортов ровно один отсек.

1.3.4. Delivery

Это подкласс Multiple Depots, и все свойства этого класса задач сохраняются, а также нужны следующие дополнительные свойства:

а) Все заказы стартуют из одного места

1) doAllPickupsStartFromSamePlace
   Все события всех заборов расположены в одной и той же локации.

б) Нет каких-бы то ни было несовместимостей

1) haveNoPerformerTransportNonCompatibilities
   Все исполнители совместимы со всеми транспортами.
2) haveNoPerformerOrderNonCompatibilities
   Все исполнители совместимы со всеми заказами.
3) haveNoOrderOrderNonCompatibilities
   Все заказы совместимы друг с другом.
4) haveNoCargoBoxNonCompatibilities
   Все грузы совместимы со всеми отсеками.
5) haveNoTransportLocationNonCompatibilities
   Все транспорты совместимы со всеми локациями.

в) Все транспорты одинаковые

1) doAllTransportsHaveSameShifts
2) doAllTransportsHaveSameLocations
   У всех транспортов одинаковые локации, если взять два разных транспорта, то смен у них поровну и в соответствующих

сменах локации старта и финиша у них совпадают (не старт с финишом, а старт одной смены со стартом другой смены).

г) Все исполнители одинаковые
   1) doAllPerformersHaveSameWorkShifts
   2) doAllPerformersHaveSameLocations

      У всех исполнителей одинаковые локации, если взять два разных исполнителя, то смен у них поровну и в соответствующих сменах локации старта и финиша у них совпадают (не старт с финишом, а старт одной смены со стартом другой смены).

д) Все исполнители одинаковые
   1) doAllPerformersHaveSameWorkShifts
   2) doAllPerformersHaveSameLocations

      У всех исполнителей одинаковые локации, если взять два разных исполнителя, то смен у них поровну и в соответствующих сменах локации старта и финиша у них совпадают (не старт с финишом, а старт одной смены со стартом другой смены).

е) Нет каких-либо сложных дополнительных ограничений
   1) haveNoMaxWorkShiftsRestrictions

      Ограничения на максимальное число рабочих смен не мешают планированию. У всех исполнителей ограничение на максимальное количество смен не меньше, чем общее количество смен исполнителя.
   2) haveNoMaxTransportsRestrictions

      Ограничение на максимальное количество транспортов в локации не мешает планированию. Для каждой локации максимальное теоретически возможное количество транспортов, которые могли бы в ней одновременно находится, не превосходит ограничение на одновременное количество транспортов в локации.
   3) haveNoMaxTotalOrdersRestrictions

      Ограничение на максимальное число выполненных заказов не мешает планированию. У всех исполнителей число заказов, которые он может теоретически выполнить за одну сме-

ну с учетом совместимостей и продолжительности заказов, не больше, чем ограничение на количество заказов в эту смену.

4) haveNoMaxTotalLocationsRestrictions
   Ограничение на максимальное число посещенных локаций не мешает планированию. У всех исполнителей в каждой смене максимальное дозволенное число посещенных локаций не менбше, чем общее число локаций.
5) !haveBreaks
   Есть ли правила перерывов исполнителей.

ж) Можно игнорировать тарификацию и допущения

1) haveTrivialTariffs
   Простая тарификация. Ограничение на продолжительность работы исполнителя не меньше, чем продолжительность смены; все время работы исполнителя тарифицируется одинаково; весь пробег транспорта тарифицируется одинаково.
2) haveTrivialContraventions
   Простые допущения. Ровно одна группа; требуется выполнить все заказы, то есть SuccessOrdersQuota 100; SoftWindowsQuota 0.

1.4. Описание данных и работа с ними

1.4.1. Исходные данные

Что же представляет из себя набор данных в представленной задаче.

В целом, это документы (xlsx формата), в которых представлена следующая информация:

а) Заказы;
б) Грузы;
в) Локации;
г) Транспортные средства;
д) Водители;
е) Передвижения (любые перемещения водителей и транспорта, и вся информация, связанная с этим).

1.4.2. Конвертирование данных в программный формат

Абсолютно очевидно, что работа с данными формата xlsx весьма затруднительна. Для этого в компании специально была написана утилита, которая называется «Mission Control Center». Эта утилита запускает локальный сервер, где можно делать различные вещи с данными в рамках компании VeeRoute. В том числе и конвертировать из xlsx в удобный json формат. Помимо простого перегона одного в другое, конвертер добавляет некоторые настройки роутинга, маршрутизации и вообще способствует более грамотному представлению данных.

Выводы по главе 1

В этой главе была рассмотрена предметная область, описаны данные, с которыми происходит работа. Также поднимается описание бизнес-задачи, которую мы решаем,и ее актуальность для компании. Рассказывается про данные, с которыми происходит работа.

ГЛАВА 2. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ РЕШЕНИЯ

В этой главе поднимаются некоторые теоретические аспекты решения. Описываются, какие признаки использовались для построения классификатора. Также подробно говорится о типах машинного обучения, использующихся при решении задачи.

2.1. Выбор признаков

Для построения классификатора были выбраны более 50 признаков. Полная информация о них представлена в приложении А. Здесь будет лишь частично рассказано о признаках в общем.

Признаки, выбранные для построения классификатора можно разбить на следующие категории:

а) Количественные признаки (количество заказов, транспортных средств, водителей, локаций в одном наборе данных)
б) Матрицы совместимости (например, между исполнителем-транспортом, транспортом-локацией, исполнителем-заказом, грузом-отсеком траспнортного средства и т.д.)
в) Статистические признаки (например, среднее количество рабочих смен для исполнителей, среднее количество грузов в заказах, средняя длительность временного отрезка в минутах (для исполнителя) длительность пути в метрах для транспорта и т.д.)
г) Геокоординаты

Иногда в машинном обучении первоначально делают отбор признаков, чтобы сократить их число при построении классификатора. В данном случае, этого делать не нужно, потому что классов наборов данных не так много.

2.2. Типы многоклассовой классификации

При решении данной задачи будут использоваться различные типы классификация. Посколько решение основано на разбивание элементов на достаточно большое число классов, то многие методы классификации плохо подходят для машинного обучения. Об это также сказано в [1]. Например, тяжело решать эту задача наивным Байесом или k-ближайшими соседями.

Про поиск эффективных методов снижения размерности при решении задач многоклассовой классификации путем ее сведения к решению бинарных задач хорошо говорится в [2]

Ниже будут расписаны методы, которые использовались для построения классификатора. Все они считаются достаточно удачными для работы со многими классами.

2.2.1. Дерево Решений. Алгоритм C4.5

Про деревья решений хорошо рассказано в [3] и [4].

Для того, чтобы с помощью C4.5 построить решающее дерево и применять его для классификации, данные должны удовлетворять нескольким условиям, о которых речь пойдет в дальнейшем.

Информация об датасетах, которые надо классифицировать, должна быть представлена в виде конечного набора атрибутов, каждый из которых имеет либо дискретное значение, либо числовое значение. Такой набор атрибутов назовём тестовой частью. В данном исследовании используются только числовые. Для всех примеров количество примеров и их состав должны быть const.

Множество категорий, на которые будут разбиваться примеры, должно иметь конечное число объектов, а каждый атрибут должен однозначно относиться к конкретной категории.

В обучающей выборке количество атрибутов должно быть сильно больше количества категорий, к тому же каждый атрибут должен быть заранее соотнесен со своим классом. Без этого грамотного построения дерева не получится.

Пусть имеется D — обучающая выборка примеров, а S — множество классов, состоящее из t элементов. Для каждого атрибута из D известна его принадлежность к какому-либо из классов $S_1 \dots S_t$.

На первом шаге имеется корень и соотнесенное с ним множество D, которое необходимо разбить на подмножества. Для этого надо выбрать один из примеров в качестве сверки. Выбранный пример V имеет t значений, что разбивает множество на t подмножеств. Далее создаются t потомков корня, каждому из которых соотносятся своё подмножество, полученное при разбиении D. Процедура выбора примера и разбиения

по нему рекурсивно применяется ко всем t потомкам и останавливается только в следующих случаях:

— Вершина оказалась ассоциированной с пустым множеством (тогда она становится листом, а в качестве решения выбирается наиболее часто встречающийся класс у предка этой вершины).
— После очередного ветвления в вершине оказываются примеры из одного класса (тогда она, то есть вершина, становится листом, а класс, которому принадлежат её примеры, будет именоваться решением листа);

2.2.2. Многоклассовый метод опорных векторов

Про обычный SVM хорошо рассказано в [5].

Суть заключается в решении нескольких бинарных задач: последовательного отделения первого класса от остальных, второго класса от оставшихся и т.д., или выделения каждого класса из всего множества. После решения этих бинарных задач получается несколько обученных SVM, соответствующих каждому классу. Далее при определении класса нового объекта каждая SVM вернет коэффициент принадлежности, и класс объекта будет определен по максимальному значению этого коэффициента. Хорошо про это рассказано в [6].

Такой подход в машинном обучении называется «Один против всех». Более подробно можно ознакомиться в [7].

2.2.3. Многоклассовая логистическая регрессия

Про обычную логистическую регрессию можно почитать в [8].

По поводу многоклассовой можно сказать,что это она схоже с предыдущим пунктом. Только в данном случае после решения бинарных задач получается несколько обученных логистических регрессий, соответствующих каждому классу. По вероятностям и будет определен класс объекта.

2.2.4. Random Forest

RF (random forest) — это множество решающих деревьев.В задаче классификации принимается решение голосованием по большинству. Все деревья строятся независимо по следующей схеме:

— Выбирается подвыборка обучающей выборки определенного размера — по ней строится дерево (для каждого дерева — своя подвыборка);
— Для построения каждого расщепления в дереве просматривается максимальное число случайных признаков (для каждого нового расщепления — свои случайные признаки);
— Выбирается наилучшие признак и расщепление по нему (по заранее заданному критерию). Дерево строится, как правило, до исчерпания выборки (пока в листьях не останутся представители только одного класса).

Выводы по главе 2

В этой главе говорилось о теоретической области решения данной задачи. Рассказывается о признаках, которые были выбраны для построения классификатора. Выделены их основные типы. Также приводится описание методов классификации, которые используются при машинном обучении в данной задаче.

ГЛАВА 3. ПРАКТИЧЕСКОЕ ИССЛЕДОВАНИЕ

В данной главе речь пойдет об анализе данных. Рассматривается практическая сторона решения. Рассказывается про классы наборов данных, которые использовались. Приведены результаты классификации новых датасетов по имеющейся информации об известных.

3.1. Классы исследования

На данный момент в компании «VeeRoute» данных не так много. Они будут появляться со временем. Для исследования брались 4 разных класса наборов данных, практически никак не связанных друг с другом.

В исследовании участвовали следующие классы:

— Norway;
— Gulfstream;
— Austria;
— DelLine.

3.1.1. Класс Norway

а) Норвежские бенчмарки
б) Были сгенерированы автоматически
в) Класс связан с задачей Delivery

3.1.2. Класс Gulfstream

а) Данные по гольфстриму
б) Сервисное обслуживание / обслуживание объектов
в) Москва и Московская область

3.1.3. Класс Austria

а) Данные почты Австрии
б) Класс связан с задачей Pickup and Delivery
в) Широкие и узкие окна в одном файле на разные даты

3.1.4. Класс DelLine

а) Москва
б) Класс связан с задачей Pickup and Delivery
в) Деловые линии, под платформу «VeeRoute», 5489 заказов

3.2. Машинное обучение в данной задаче

В каждом классе выбиралось одинаковое числов наборов для рассмотрения (10). В качестве обучающей выборки бралось — 50 % от каждого класса наборов данных. В качестве тестовой выборки — 50 % оставшихся данных. При каждом запуске данные перемешивались.

Стоит также отметить,что наборы данных для всех классов, кроме Norway, были получены из одного большого датасета, путем семплирования. Каждый большой набор данных соответственно делился на много маленьких.

Определение. Истинно-положительные (true positives) — те объекты, которые должны были быть определены и определились на выходе.

Определение. Ложно-положительные (false positives) — те объекты, которые не должны были быть определены, но анализатор их вернул на выходе.

Определение. Ложно-отрицательные (false negatives) — те объекты, которые должны были быть определены, но их не вернули на выходе.

Определение. Истинно-отрицательные (true negatives) — объекты, которых быть на выходе не должно и анализатора их совершенно правильно не вернул.

Определение. Истинно-отрицательные (true negatives) — объекты, которых быть на выходе не должно и анализатора их совершенно правильно не вернул.

Определим меру точности как:

$$precision = \frac{tp}{tp + fp}$$

Определим меру полноты как:

$$recall = \frac{tp}{tp + fn}$$

Определение. F_1 мера — среднее гармоническое величин $precision$ и $recall$.

$$F_1 = 2 * \frac{precision * recall}{precison + recall}$$

3.3. Результаты работы программы

Для классификации использовалась утилита Weka. Тестовые и обучающие выборки представлены в формате ARFF. Файлы включает в себя заголовок с типами (в данном случае, с числовыми) и информационную секцию со значениями по каждому признаку. Также указано количество классов, на которые будет происходить классификация (входит в заголовок).

В таблицах приведены результаты пяти запусков, каждый раз выборки менялись. В левом столбце указаны типы классификаций, в верхней строчке указываются прогоны и среднее арифметическое по всем запускам в этой категории.

Благодаря этому тестированию, можно будет понимать, какой тип классификации наилучшим образом работает с данной задачей.

Очень важно также понимать, какой ключевой фактор определения принадлежности к тому или иному классу. То есть, какие признаки чаще всего «классифицируют» датасеты.

3.3.1. Все признаки используются

Таблица 1 – Все признаки используются

	1	2	3	4	5	Среднее
Дерево Решений	0.949	0.949	0.949	0.949	0.949	0.949
SVM	0.949	0.949	0.949	0.949	0.949	0.949
LogReg	0.949	0.949	0.949	0.949	0.949	0.949
RandomForest	0.903	0.949	0.903	0,852	0.949	0.911

В целом, опираясь на таблицу 1, можно сказать,что все типы, за исключением Random Forest, показали отличный результат. С помощью «Деревьев Решений» можно внимательно оценить, какие признаки влияли на результат.

Как видно из рисунка 1 , ключевыми признаками являются:

— $AverageCostPerShift$ / Средняя цена за использование смены исполнителем или транспортом
— $AverageCostPerUnit$ / Средняя цена за минуту (для исполнителя), метр (для транспорта)
— $AmountOfPerformers$ / Количество исполнителей заказов

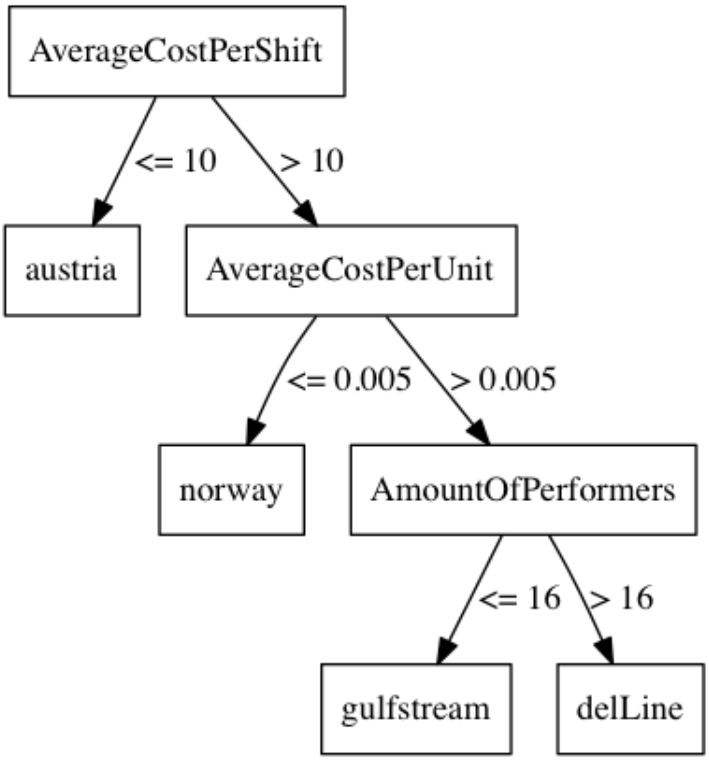

Рисунок 1 – «Дерево решений» в случае со всеми признаками

3.3.2. Убираются три признака при построении классификатора

Очень важно иногда убирать признаки, при тестировании, чтобы посмотреть, как поведет себя F_1 мера. Некоторые признаки могут отрицательно влиять на классификацию. На самом деле, отрицательно — не значит плохо. Просто если какой-то признак будет постоянно являться ключевым при определении принадлежности к тому или иному датасету, то это не совсем хорошо, поскольку важно собрать, как можно больше различных признаков, которые будут являться ключевыми при классификации. Например, в предыдущем пункте указаны признаки, по которым, если внедрить какие-нибудь данные из реальной жизни, классификатор не будет к этому готов. Имеет место переобучение.

Далее мы «зашумим» три признака, которые влияли на результат в прошлой подглаве.

В целом, опираясь на таблицу 2, можно сказать,что все типы показали отличный результат. Чуть хуже повела себя логистическая регрессия.

Как видно из рисунка 2 , ключевыми признаками являются:

— $AverageCargosPerDemand$ / Среднее количество грузов в заказах

Таблица 2 – Не используется признаки количества водителей, средняя цена за использование смены исполнителем или транспортом, средняя цена за минуту (для исполнителя), метр (для транспорта)

	1	2	3	4	5	Среднее
Дерево Решений	1.000	0.949	1.000	0.949	0.949	0.969
SVM	1.000	0.949	1.000	0.949	0.949	0.969
LogReg	0.949	0.949	1.000	0.949	0.899	0.949
RandomForest	1.000	0.949	1.000	0,949	0.949	0.969

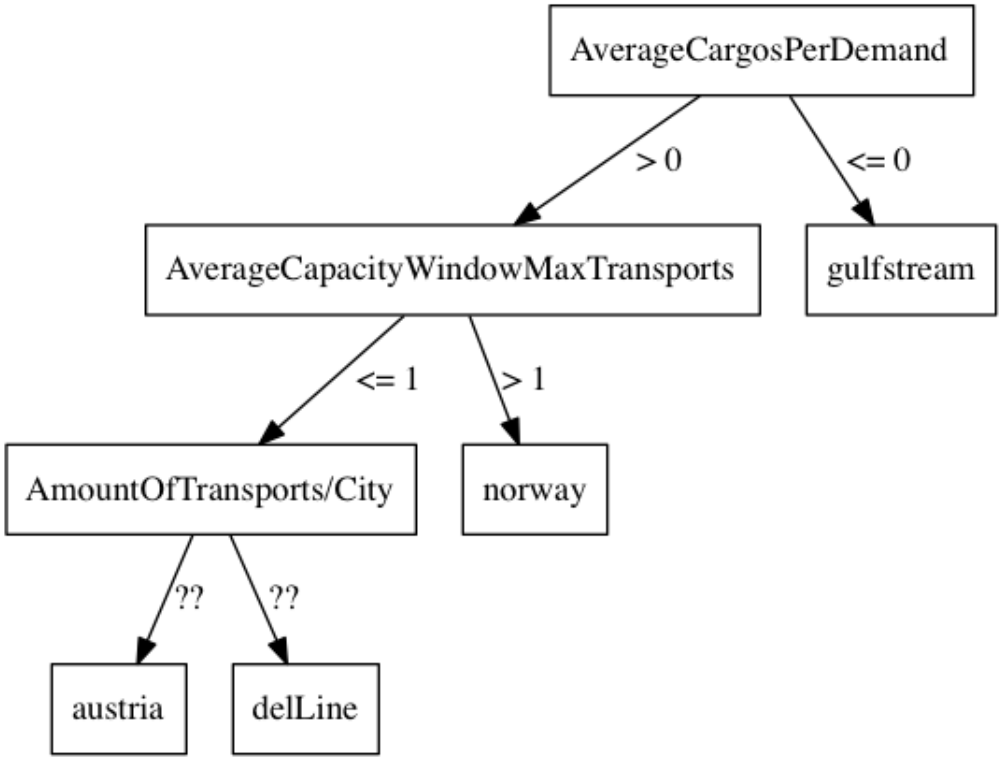

Рисунок 2 – «Дерево решений» в случае с 3 зашумленными признаками

— $AverageCapacityWindowMaxTransports$ / Количество транспорта, который может работать в локации одновременно

Собственно $AverageCargosPerDemand$ действительно очень правильно определяет класс *gulfstream*, так он очень существенно выделяется количеством грузов в заказах. $AverageCapacityWindowMaxTransports$ чаще всего пишется тестировщиками, так что по этому признаку очень тяжело судить, его можно отбросить.

3.3.3. «Зашумляется» еще один признак при построении классификатора

Далее исследование продолжается без четырех признаков, рассматриваемых выше.

Таблица 3 – Не используются 4 признака, указанных выше в подглавах 3.3.1 и 3.3.2

	1	2	3	4	5	Среднее
Дерево Решений	1.000	0.949	1.000	1.000	0.949	0.980
SVM	1.000	0.949	1.000	1.000	0.949	0.980
LogReg	0.949	0.949	1.000	1.000	0.949	0.969
RandomForest	0.949	0.949	0.949	0.949	0.899	0.940

Опираясь на таблицу 3, можно заметить,что результаты на всех типа показаны одинаково неплохие.

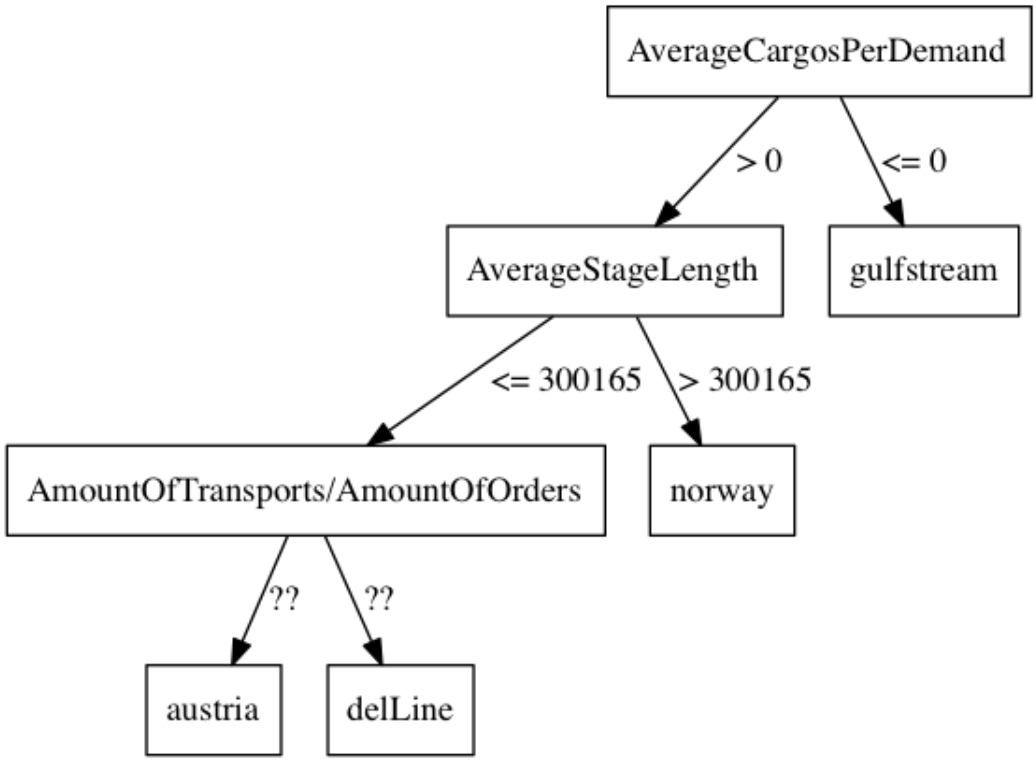

Рисунок 3 – Первый из встречающихся вариантов «Дерева решений» в случае с 4 «зашумленными» признаками

Как видно из рисунка 3 и рисунка 4 , ключевыми признаками в данном случае чаще всего являлись:

— $AverageStageLength$ / Средняя длительность временного отрезка в минутах (для исполнителя), длительность пути в метрах (для транспорта)

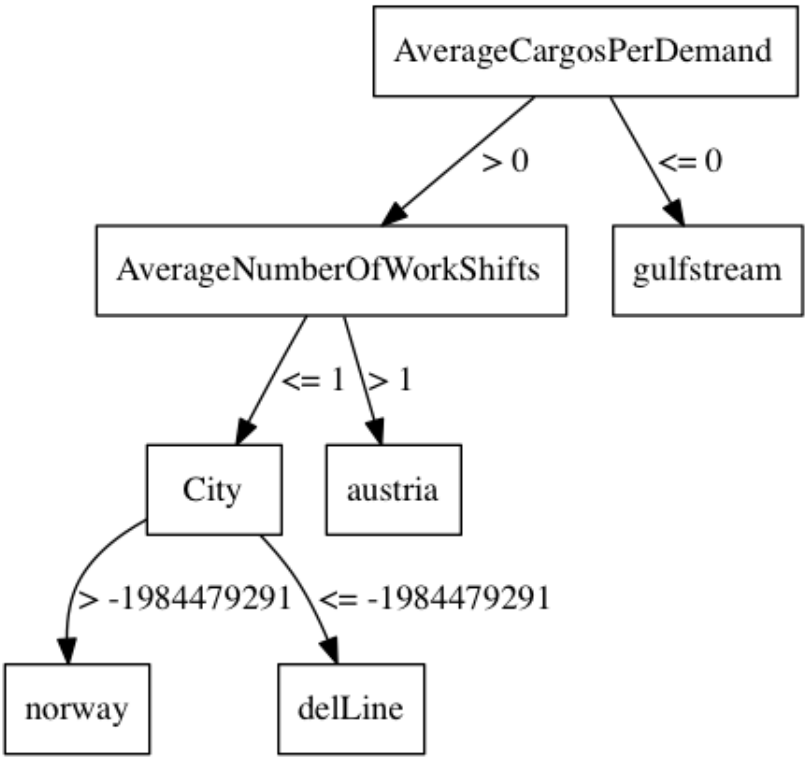

Рисунок 4 – Второй из встречающихся вариантов «Дерева решений» в случае с 4 «зашумленными» признаками

— $AverageNumberOfWorkShifts$ / Среднее количество рабочих смен для исполнителей
— $AmountOfOrders$ / Количество заказов
— $AmountOfTransports$ / Количество транспортых средств, выполняющих заказы
— $City$ / Город, где проводится доставка

Первый признак действительно грамотный для определения класса *norway*, второй хорошо идентифицирует класс *austria*.

Про последние 3 признака и их не слишком большую информативность, речь пойдет в следующем разделе.

3.3.4. Дополнительные прогоны

Признаки $AmountOfOrders$, $AmountOfTransports$, $City$ на тех данных, на которых происходит исследование, могут давать не всегда информативные результаты.Во многом, это связано со спецификой семплирования. К тому же город тоже очень сильно может перетягивать одеяло на себя по той причине, что данных очень мало, и регионы в датасетах практически все различные.

Попробуем посмотреть,что получится, если «зашумить» все неинформативные признаки, и те, что указаны в этом разделе.

Таблица 4 – Дополнительные прогоны с «зашумлением» неинформативных признаков

	1	2	3	4	5	Среднее
Дерево Решений	1.000	0.949	0.949	1.000	1.000	0.980
SVM	0,949	0,949	0.949	1.000	0.949	0.960
LogReg	0.675	0.764	0.804	0.709	0.719	0.734
RandomForest	0,899	0,896	0.899	1.000	0.949	0.929

Можно обратить внимание, на то, что на этих прогонах сильно «просела» LogReg, как и Random Forest. Лучший результат показало «Дерево Решений».

Признаки чаще всего определялись из того множества, что было указано в предыдущих разделах. Без многих неинформативных признаков ключевыми также стали features, посчитанные по матрицам совместимости. Пример «точного» решения представлен на рисунке 5. Здесь F_1 мера постоянно равнялась 1, на нижнем уровне менялись признаки, но они всегда относились к категории «матрицы совместимости».

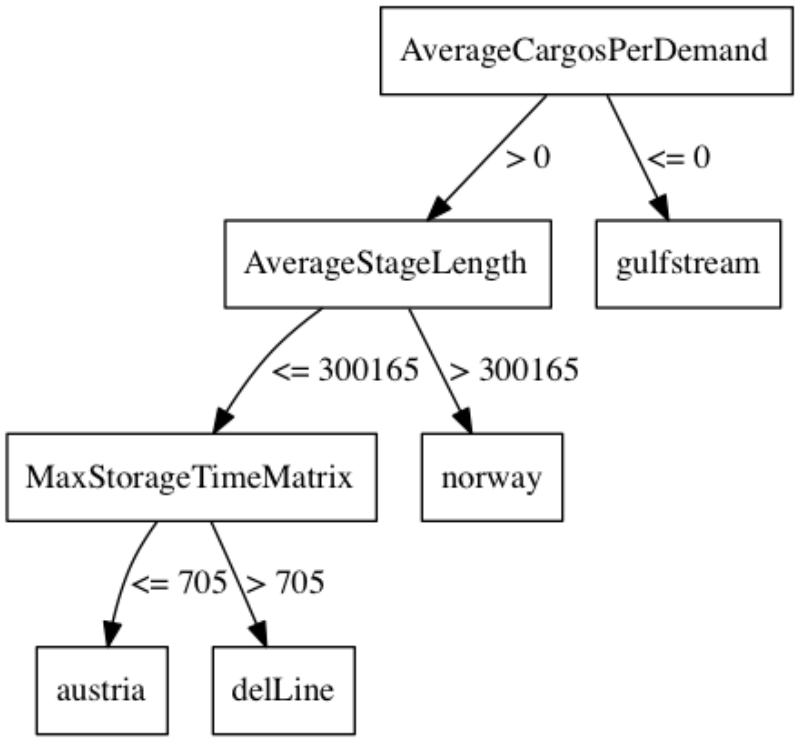

Рисунок 5 – Пример «Дерева Решений», в котором присутствуют признаки «матрицы совместимости»

Выводы по главе 3

В данной главе речь идет об анализе данных, рассматривается практическая сторона решения. Рассказывается про классы наборов данных, которые были рассмотрены. Высчитывается метрика на нескольких моделях, выделяются ключевые признаки, делаются выводы. Как можно заметить, лучший результат в среднем показывается при «Дереве Решений». Усредненная F_1 мера составляет 0.967.

ЗАКЛЮЧЕНИЕ

В данной работе была рассмотрена важная и интересная бизнес-задача, необходимая компании VeeRoute. С ее помощью компания сможет классифицировать новые наборы данных,на основании информации об известных. На данные, которые уже есть у компании, имеется готовый набор конфигурационных файлов, который учитывает специфику бизнес-процессов и формат входного файла. Указав машинным обучением в данной работой на то, к какой категории относится датасет, и по каким признакам это определяется, компания VeeRoute сможет уйти от ручной проверки, которая занимает очень много времени при большом количестве данных. В данный момент, датасетов, к сожалению, не так много, поэтому достоверно оценить качество проведенного исследования достаточно проблематично. Но на данном этапе результаты, полученные в данной работе вполне удовлетворяют запросам компании. Достигнута основная задача — можно давать программе тестовую выборку датасетов, и она определит к катой категории из обучающей выборки принадлежит искомые объекты.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Sobolev A.*, *Barinova O.* Многоклассовая классификация в задаче семантической сегментации //. — Факультет Вычислительной Математики и Кибернетики Московский Государственный Университет им. Ломоносова, Москва, Россия.

2 *Карасиков М. Е.*, *Максимов Ю. В.* Поиск эффективных методов снижения размерности при решении задач многоклассовой классификации путем её сведения к решению бинарных задач // Поиск эффективных методов решения задач многоклассовой классификации. — 2014.

3 *Сидоров А.* Алгоритмы создания дерева принятия решений.

4 Classification and Regression Trees. — 2014.

5 *Воронцов К. В.* Лекции по методам опорных векторов // Курс по машинному обучению. — 2007.

6 Классификация веб-страниц на основе алгоритмов машинного обучения / П. В. Борисова [и др.] //. — 2005.

7 *Соколов Е. А.* Многоклассовая классификация и категориальные признаки // Поиск эффективных методов решения задач многоклассовой классификации. — 2016.

8 *Воронцов К. В.* Лекции по алгоритмам восстановлении регрессии // Курс по машинному обучению. — 2007.

ПРИЛОЖЕНИЕ А. ПРИЗНАКИ ДЛЯ ПОСТРОЕНИЯ КЛАССИФИКАТОРА

Ниже представлено название/описание, использованных для построения классификатора признаков.

- AmountOfOrders / Количество заказов
- AmountOfPerformers / Количество исполнителей заказов
- AmountOfTransports / Количество транспортых средств, выполняющих заказы
- City / Город, где проводится доставка
- MaxStorageTimeMatrix / Матрица совместимости (состоит из максимального времени возможного нахождения груза заявки в данном отсеке)
- OnBoardMatrix / Матрица загрузки груза заявок, которые уже загружены в отсек транспорта
- OrderOrderMatrix / Матрица совместимости. Данная связь (OrderOrderRelations) используется для определения возможности перевозки грузов разных заказов в одном отсеке одного транспорта и одним исполнителем, данная совместимость интегральная, т.е. учитываются ограничения на все заявки (грузы) входящие в заказ
- PerformerOrderMatrix / Матрица совместимости. Данная связь (PerformerOrderRelations) используется для определения возможности исполнителя работать с определенным заказом (и всеми заявками, которые в него входят).
- PerformerTransportMatrix / Матрица совместимости. Данная связь (PerformerTransportRelations) используется для определения возможности исполнителя управлять определенным транспортным средством
- TransportLocationMatrix / Матрица совместимости. Данная связь (TransportLocationRelations) используется для определения возможности транспорта производить работу на конкретных локациях
- TransportType / Типы транспортов и количество, которые участвуют в маршрутизации

— AverageDistanceInLengthMatrix / Базовая матрица расстояний между точками, в метрах. Среднее расстояние между соседними точками
— RangTimeMatrix / Базовая матрица времен, необходимых на перемещения из каждой точки в каждую, в минутах. Ранг
— AverageTimeInTimeMatrix / Базовая матрица времен, необходимых на перемещения из каждой точки в каждую, в минутах. Среднее время перемещения между соседними точками.
— RangLengthMatrix / Базовая матрица расстояний между точками, в метрах. Среднее расстояние между соседними точками.
— AverageCostPerUnit / Средняя цена за минуту (для исполнителя), метр (для транспорта)
— AverageStageLength / Средняя длительность временного отрезка в минутах (для исполнителя), длительность пути в метрах (для транспорта)
— AverageCostPerShift / Средняя цена за использование смены исполнителем или транспортом
— AverageArrivalDuration / Среднее время на подъезд к локации
— AverageDepartureDuration / Среднее время на отъезд с локации
— AmountOfLocations / Количество локаций
— AverageCapacityTimeWindowDuration / Окна загрузки или разгрузки (CapacityWindow) для локаций определяют количество груза, которое локация способна отдать принять. Продолжительность среднего временное окно по всем локациям
— AverageCapacityWindowMaxTransports / Количество транспорта, который может работать в локации одновременно. Среднее количество такого транспорта по всем локациям,где есть окна загрузки или разгрузки
— CapacityWindowTypePICKUP / Количество окон на загрузку (перемещение груза из локации в транспорт)
— CapacityWindowTypeDROP / Количество окон на разгрузку (перемещение груза из транспорта в локацию)
— CapacityWindowTypeWORK / Количество окон на доступное время, когда возможна работа на локации

— AverageCargosCapacity / Средняя вместимость грузов по заказам
— AverageCargosPerDemand / Среднее количество грузов в заказах
— AverageEventDuration / Среднее время выполнения события, в минутах
— AverageHardTimeWindowDuration / Среднее жесткое временное окно для события. Продолжительность
— AverageEventSoftTimeWindowDuration / Среднее мягкое(желаемое) временное окно для события. Продолжительность
— AverageNumberOfPossibleEvents / Среднее количество событий в каждой заявке, выполнение любого из которых (из событий) дает выполнение заказа
— AverageCapacityTimeWindowFrom / Окна загрузки или разгрузки (CapacityWindow) для локаций определяют количество груза, которое локация способна отдать(принять). Среднее начало временного окна по всем локациям
— AverageCapacityTimeWindowTo,1351.0,Окна загрузки или разгрузки (CapacityWindow) для локаций определяют количество груза, которое локация способна отдать (принять). Среднее окончание временного окна по всем локациям
— AverageEventHardTimeWindowFrom / Среднее жесткое временное окно для события. Конец окна
— AverageEventHardTimeWindowTo / Среднее жесткое временное окно для события. Начало окна
— AverageEventSoftTimeWindowFrom / Среднее мягкое (желаемое) временное окно для события. Начало окна
— AverageEventSoftTimeWindowTo / Среднее мягкое (желаемое) временное окно для события.Конец окна
— AverageStartBreakTime / Среднее время начала перерыва (использование данного параметра зависит от типа перерыва), в минутах
— AverageNumberOfBreaks / Среднее количество перерывов, в которое исполнители не доступны для выполнения заказов
— AverageDurationBreakTime / Средняя длительность перерыва, в минутах

— NumberOfBreakTypeBTFIXED / Количество перерывов, которые начинаются в указанное время (break time)
— NumberOfBreakTypeBTITERATIVE / Количество перерывов, которые необходимо выполнять раз в указанное количество минут (break time), при этом считается время работы driving time + waiting time.
— AverageNumberOfWorkShifts / Среднее количество рабочих смен для исполнителей
— AveragePerformerTimeWindowDuration / Среднее временное окно действия смены, в минутах. Продолжительность окна
— AveragePerformerTimeWindowFrom / Среднее временное окно действия смены, в минутах. Начало окна
— AveragePerformerTimeWindowTo / Среднее временное окно действия смены, в минутах. Конец окна
— AverageNumberOfBoxes / Среднее число отсеков — часть транспорта, способная вмещать груз
— AverageNumberOfCapacitiesInBoxes / Среднее число вариантов вместимости в отсеках
— AverageBoxesCapacity / Средняя вместимость отсеков
— AverageTransportTimeWindowDuration / Среднее временное окно действия смены, в минутах. Продолжительность окна
— AverageTransportTimeWindowFrom / Среднее временное окно действия смены, в минутах. Начало окна
— AverageTransportTimeWindowTo / Среднее временное окно действия смены, в минутах. Начало окна