Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Волгоградский государственный технический университет»

Факультет Электроники и Вычислительной Техники

Кафедра ЭВМ и Системы

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе (проекту)

по дисциплине Системы обработки больших данных

на тему Использование алгоритмов машинного обучения для анализа набора
больших данных «Cyclistic»

Студент Шебаршов Артем Александрович
(фамилия, имя, отчество)

Группа САПР 1.4.

Руководитель работы (проекта) _____ Кравченя П.Д.
(подпись и дата подписания) (инициалы и фамилия)

Члены комиссии:

_____ _____
(подпись и дата подписания) (инициалы и фамилия)

_____ _____
(подпись и дата подписания) (инициалы и фамилия)

_____ _____
(подпись и дата подписания) (инициалы и фамилия)

Нормоконтролер _____ _____
(подпись, дата подписания) (инициалы и фамилия)

Волгоград 2022 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Волгоградский государственный технический университет»

Факультет Электроники и Вычислительной Техники_____

Направление (специальность) Информатика и Вычислительная Техника_____

Кафедра ЭВМ и Системы_____

Дисциплина Системы обработки больших данных _____

Утверждаю
Зав. кафедрой_____

«_____» _____20 ___ г.

ЗАДАНИЕ
на курсовую работу (проект)

Студент Шебаршов Артем Александрович_____
(фамилия, имя, отчество)

Группа САПР 1.4._____

1. Тема: Использование алгоритмов машинного обучения для анализа набора _
больших данных «Cyclistic» _____
_____
_____

Утверждена приказом от «_____» _____ 20___ г. № _____

2. Срок представления работы (проекта) к защите «___»_____20__ г.

3. Содержание расчетно-пояснительной записки: разведочный анализ датасета_
очистка и подготовка датасета, корреляции, построение диаграмм, _____
решение задачи бинарной классификации, решение задачи регрессии_____
_____
_____

4. Перечень графического материала: _____
_____
_____

5. Дата выдачи задания «_____» _____20 ___ г.

Руководитель работы (проекта)_____ ___Кравченя П.Д.__
подпись, дата инициалы и фамилия

Задание принял к исполнению_____ __Шебаршов А.А.__
подпись, дата инициалы и фамилия

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Определение больших данных — это данные, содержащие большее разнообразие (variety), поступающие в возрастающих объемах (volume) и с большей скоростью (velocity). Это также известно как три V.

Проще говоря, большие данные — это более крупные и сложные наборы данных, особенно из новых источников данных. Эти наборы данных настолько объемны, что традиционное программное обеспечение для обработки данных просто не может ими управлять. Но эти огромные объемы данных можно использовать для решения бизнес-задач, с которыми раньше не могли справиться.

За последние несколько лет появились еще два V: ценность (value) и достоверность (veracity). Данные имеют внутреннюю ценность. Но она бесполезна, пока она не будет обнаружена. Не менее важно и насколько правдивы ваши данные и насколько вы можете на них полагаться?

Сегодня большие данные стали капиталом. Подумайте о некоторых крупнейших технологических компаниях мира. Большая часть ценности, которую они предлагают, исходит от их данных, которые они постоянно анализируют для повышения эффективности и разработки новых продуктов.

Недавние технологические прорывы в геометрической прогрессии снизили стоимость хранения данных и вычислений, упрощая и удешевляя хранение большего объема данных, чем когда-либо прежде. Благодаря увеличению объема больших данных, которые стали дешевле и доступнее, вы можете принимать более точные и точные бизнес-решения.

Поиск ценности в больших данных заключается не только в их анализе (что является еще одним преимуществом). Это целый процесс исследования, который требует проницательных аналитиков, бизнес-пользователей и руководителей, которые задают правильные вопросы, распознают закономерности, делают обоснованные предположения и предсказывают поведение.

Целью данной работы является исследование датасета «Cyclistic», содержащего данные о поездках на велосипедах в городе Чикаго за 2021 год.

Для достижения поставленной цели необходимо решение следующих задач:

– провести разведочный анализ датасета;

– определить корреляции между значениями в датасете;

– провести регрессионный анализ и определить зависимость расстояния от других параметров.

В первой главе рассматривается разведочный анализ датасета и его подготовка к применению алгоритмов машинного обучения.

Во второй главе рассматривается решение задач машинного обучения с использованием алгоритмов случайного леса и линейной регрессии.

1 РАЗВЕДОЧНЫЙ АНАЛИЗ ДАННЫХ

Разведочный анализ данных – анализ основных свойств данных, нахождение в них общих закономерностей, распределений и аномалий, построение начальных моделей, зачастую с использованием инструментов визуализации.

Проведём разведочный анализ датасета с использованием фреймворка PySpark. Apache Spark — это платформа параллельной обработки с открытым кодом, которая поддерживает обработку в памяти, чтобы повысить производительность приложений, анализирующих большие данные. Решения для работы с большими данными предназначены для обработки данных со слишком большим объемом или сложностью для традиционных баз данных. Spark обрабатывает большие объемы данных в памяти, что намного быстрее, чем альтернативная обработка с использованием диска.

PySpark – это API Apache Spark для Python, позволяющий использовать возможно по обработке данных Spark в удобной среде исполнения Python. При этом запускается сервер Spark, осуществляющий обработку данных, взаимодействие с которым происходит через SparkContext в Python-программе. При этом пользователю доступны все возможности Spark, включая чтение и запись данных на диск, SQL-подобные манипуляции над данными (Spark SQL) и использование различных алгоритмов анализа данных и машинного обучения (Spark ML).

Работа выполнялась в Jupyter Notebook, сервер которого запускался в виде Docker-контейнера на основе образа jupyter/pyspark-notebook.

1.1 Обзор датасета

В работе исследуется следующий датасет с Kaggle: https://www.kaggle.com/datasets/sarfarazmulla/google-data-analytics-capstone-cyclistic. Данный датасет содержит данные о поездках на велосипедах,

предоставленные компанией Cyclistics, занимающейся арендой велосипедов. Датасет содержит такие данные как точки начала и конца поездки, дата и время, длительность поездки. Пример содержимого датасета представлен на рисунках 1 и 2.

	ride_id	bike_type	start_date	end_date	start_station_name	start_station_id	end_station_name	end_station_id
0	7C660BB0C4949ABC	classic_bike	2021-08-23 18:47:03	2021-08-23 19:00:08	Clark St & Wrightwood Ave	TA1305000014	Southport Ave & Clybourn Ave	TA1309000030
1	FE73183E7F2B55D2	classic_bike	2021-08-17 18:04:40	2021-08-17 18:34:11	Clark St & Wrightwood Ave	TA1305000014	DuSable Lake Shore Dr & Monroe St	13300
2	121289DD4E7C56DE	docked_bike	2021-08-11 16:52:15	2021-08-11 17:21:23	Halsted St & Polk St	TA1307000121	Wallace St & 35th St	TA1308000045
3	EDD8BE43E39C14A2	classic_bike	2021-08-08 17:09:02	2021-08-08 17:33:43	900 W Harrison St	13028	DuSable Lake Shore Dr & Monroe St	13300
4	E47F8C8A242BA089	classic_bike	2021-08-26 18:37:23	2021-08-26 18:54:28	Morgan St & Polk St	TA1307000130	DuSable Lake Shore Dr & Monroe St	13300

Рисунок 1 – Начало датасета

start_lat	start_lng	end_lat	end_lng	member_casual	duration	day_of_week	month	distance
41.929546	-87.643118	41.920771	-87.663712	member	0:13:05	Monday	August	1963.381209
41.929546	-87.643118	41.880958	-87.616743	member	0:29:31	Tuesday	August	5826.986286
41.871840	-87.646640	41.830629	-87.641290	casual	0:29:08	Wednesday	August	4603.825441
41.874754	-87.649807	41.880958	-87.616743	casual	0:24:41	Sunday	August	2823.031438
41.871737	-87.651030	41.880958	-87.616743	casual	0:17:05	Thursday	August	3018.261016

Рисунок 2 – Конец датасета

Рассмотрим подробнее каждую колонку (таблица 1).

Таблица 1 – Описание признаков в датасете

Признак	Тип признака	Описание
ride_id	порядковый	уникальный идентификатор поездки
bike_type	категориальный	тип велосипеда
start_date, end_date	датавременной	временные метки начала и окончания поездки
start_station_name, end_station_name	категориальный	названия начальной и конечной станции
start_station_id, end_station_id	категориальный	идентификаторы начальной и конечной станции
start_lat, start_lng, end_lat, end_lng	количественный	координаты точек начала и окончания поездки
member_casual	бинарный	имеет ли пользователь членство компании Cyclistics
duration	количественный	длительность поездки
day_of_week, month	категориальный	день недели и месяц, когда была совершена поездка
distance	количественный	расстояние между начальной и конечной точками

1.2 Очистка и подготовка датасета

Очевидно, некоторые из колонок имеют неподходящий для них тип – даты записаны в виде строк и имеют два различных формата; длительность указана в виде строки в формате “h:mm:ss”.

Для преобразования дат к нужному формату воспользуемся функцией parse модуля dateutil, преобразовав её к udf. Длительность поездки приведём к секундам, реализовав собственную функцию.

parse_date_udf = udf(parse, returnType=TimestampType())

```
@udf(returnType=IntegerType())
def get_total_seconds(time):
    h, m, s = time.split(':')
    return int(h) * 3600 + int(m) * 60 + int(s)
```

Посмотрим на описательные характеристики датасета (рисунки 3 и 4).

	summary	ride_id	bike_type	start_date	end_date	start_station_name	start_station_id	end_station_name	end_station_id	
0	count	5901044	5901044	5901044	5901044	5040284	5040286	4981382	4981382	
1	mean	Infinity	None	None	None	351.0	12460.295613189059	None	12522.998857156143	41.9
2	stddev	NaN	None	None	None	0.0	7497.373639072129	None	7563.218453559967	0.047
3	min	00000123F60251E6	classic_bike	01/04/2022 00:01:48	01/04/2022 00:02:15	"Senka ""Edward Duke"" Park"	1011	"Senka ""Edward Duke"" Park"	1011	
4	max	FFFFFF1C07792282	electric_bike	30/04/2022 23:59:54	30/04/2022 23:59:57	Zapata Academy	chargingstx5	Zapata Academy	chargingstx5	

Рисунок 3 – Описательные характеристики (начало)

start_lat	start_lng	end_lat	end_lng	member_casual	duration	day_of_week	month	distance
5901044	5901044	5895454	5895454	5901044	5901044	5901044	5901044	5901044
41.900768479881606	-87.64742205707789	41.90102103858501	-87.64760841946536	None	None	None	None	11455.262955632648
0.04725636724911936	0.030938997541463977	0.04734829568439617	0.0305654627672688	None	None	None	None	301802.91674231284
41.64	-87.84	41.39	-88.97	casual	#REF!	Friday	April	0.0
45.63503432	-73.79647696	42.37	-87.5	member	9:59:59	Wednesday	September	9823369.411

Рисунок 4 – Описательные характеристики (продолжение)

Можно заметить нереалистичные значения некоторых характеристик – например, расстояние более чем 9 тыс. км. Как видно из рисунка 5, у таких поездок не указана точка назначения.

start_lat	start_lng	end_lat	end_lng	member_casual	duration	day_of_week	month	distance
41.933140	-87.647760	NaN	NaN	member	0:30:09	Sunday	August	9812982.642
41.929143	-87.649077	NaN	NaN	casual	3:16:43	Sunday	August	9813079.355
41.692263	-87.642612	NaN	NaN	casual	24:59:56	Wednesday	August	9811822.071

Рисунок 5 – Поездки без точки назначения

Удалим такие строки при помощи функции dropna(), а также применим ранее объявленные функции для приведения времени и даты к правильным форматам (рисунок 6).

	summary	start_lat	start_lng	end_lat	end_lng	duration	distance
0	count	4629053	4629053	4629053	4629053	4629053	4629053
1	mean	41.90257810060474	-87.64433041470112	41.90286330888799	-87.64457389527288	1115.824882756797	2110.26820530563
2	stddev	0.04150126366706359	0.024994098451104636	0.041635399506594256	0.024377122544771136	5026.930084473847	1950.7612951168765
3	min	41.64850076	-87.83325417	41.64850076	-87.83	-6779	0.0
4	max	45.63503432	-73.79647696	42.16811567	-87.52740467	2497750	1189521.648

Рисунок 6 – Новые храктеристики

Можно заметить, что минимальное значение длительности поездки – отрицательное. Отфильтруем поездки с нулевой или отрицательной длительностью

Максимальное значение уменьшилось, но оно всё ещё превышает 1 тыс. км (рисунок 7). Определим строки с таким значением.

start_station_id	end_station_name	end_station_id	start_lat	start_lng	end_lat	end_lng	member_casual	duration	day_of_week	month	distance
Pawel Bialowas - Test- PBSC charging station	Pawel Bialowas - Test- PBSC charging station	Pawel Bialowas - Test- PBSC charging station	45.635034	-73.796477	41.8646	-87.681	casual	155	Friday	January	1189521.648

Рисунок 7 – Обнаруженный выброс по дистанции

Судя по координатам, этот человек доехал от Монреаля до Чикаго, причём сделал это всего за 2.5 минуты. Вероятно, это тестовые данные, которые остались в базе данных компании (об этом также говорит Test в названии станции).

Для удаления выбросов воспользуемся межквартильным расстоянием. Определим первую (0.25) квартиль $q1$ и третью (0.75) квартиль $q3$, расстояние iqr между ними и зададим диапазон допустимых значений как $[q1 - 1.5iqr, q3 + 1.5iqr]$. Отфильтруем все строки, у которых значения в данной колонке не попадают в заданный диапазон. Проведём эту операцию для колонок *duration* и *distance* (рисунок 8).

	summary	duration	distance
0	count	4066918	4066918
1	mean	717.112921873517	1788.4576520259002
2	stddev	484.71316530615803	1215.9544353160509
3	min	1	0.0
4	max	2378	5392.1698

Рисунок 8 – Результат удаления выбросов

Некоторые поездки имеют нулевую дистанцию при адекватном времени поездки. Эти поездки, также, начинаются и заканчиваются на одной и той же станции. Пометим такие поездки как “круговые” в новой колонке.

```
df = df.withColumn('round_trip', when(df.distance == 0.0, 1).otherwise(0))
```

Колонки bike_type и member_casual можно преобразовать к бинарным признакам:

```
df = (
    df.withColumn('electric_bike', when(df.bike_type == 'electric_bike', 1).otherwise(0))
      .withColumn('docked_bike', when(df.bike_type == 'docked_bike', 1).otherwise(0))
      .withColumn('member', when(df.member_casual == 'member', 1).otherwise(0))
      .drop('bike_type', 'member_casual', 'ride_id', 'start_station_name', 'start_station_id', 'end_station_name', 'end_station_id')
)
```

После преобразований и удаления ненужных для дальнейшего анализа колонок получаем следующий датасет (рисунок 9).

	start_date	end_date	start_lat	start_lng	end_lat	end_lng	duration	day_of_week	month	distance	round_trip	electric_bike	docked_bike	member
0	2021-08-23 18:47:03	2021-08-23 19:00:08	41.929546	-87.643118	41.920771	-87.663712	785	Monday	August	1963.381209	0	0	0	1
1	2021-08-11 16:52:15	2021-08-11 17:21:23	41.871840	-87.646640	41.830629	-87.641290	1748	Wednesday	August	4603.825441	0	0	1	0
2	2021-08-08 17:09:02	2021-08-08 17:33:43	41.874754	-87.649807	41.880958	-87.616743	1481	Sunday	August	2823.031438	0	0	0	0
3	2021-08-26 18:37:23	2021-08-26 18:54:28	41.871737	-87.651030	41.880958	-87.616743	1025	Thursday	August	3018.261016	0	0	0	0
4	2021-08-21 10:20:20	2021-08-21 10:55:29	41.892278	-87.612043	41.880958	-87.616743	2109	Saturday	August	1317.484670	0	0	1	0

Рисунок 9 – Результирующий датасет

1.3 Корреляции

Для построения матрицы корреляции числовых признаков датасета воспользуемся классом Correlation из библиотеки pyspark. После этого отобразим полученную матрицу в виде тепловой карты используя функцию heatmap библиотеки seaborn (рисунок 10).

Рисунок 10 – Матрица корреляций в виде тепловой карты

1.4 Построение диаграмм

Построим круговые диаграммы для типа поездки, типа велосипеда и типа членства (рисунок 11). Для этого сгруппируем датасет по исследуемому признаку и определим количество строк в каждой группе. Полученный датафрейм преобразуем к датафрейму pandas и воспользуемся функцией pyplot.pie() из библиотеки matplotlib. Последующие графики строятся схожим образом.

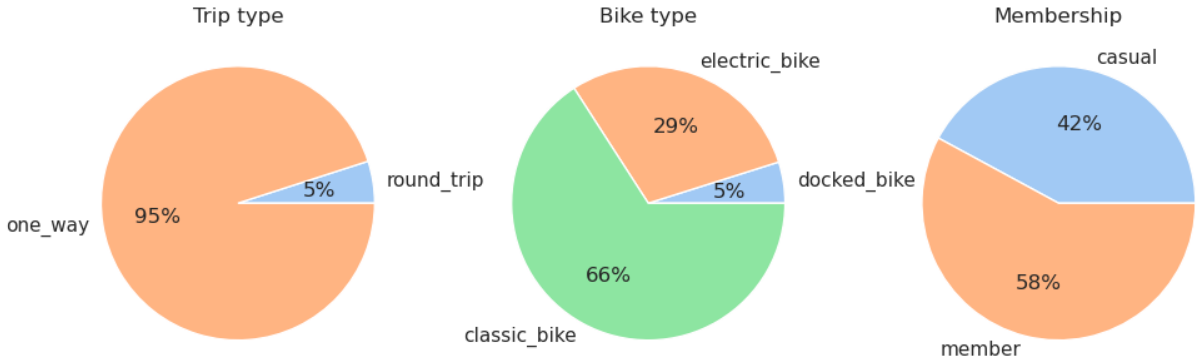

Рисунок 11 – Тип поездки, тип велосипеда и тип членства

По диаграммам видно, что большинство совершаемых поездок – в одну сторону; наибольшей популярностью пользуются классические велосипеды, наименьшей – docked велосипеды; членство компании имеет чуть больше половины пользователей велосипедов. Построим диаграммы (рисунок 12) распределения поездок по дням недели и месяцам.

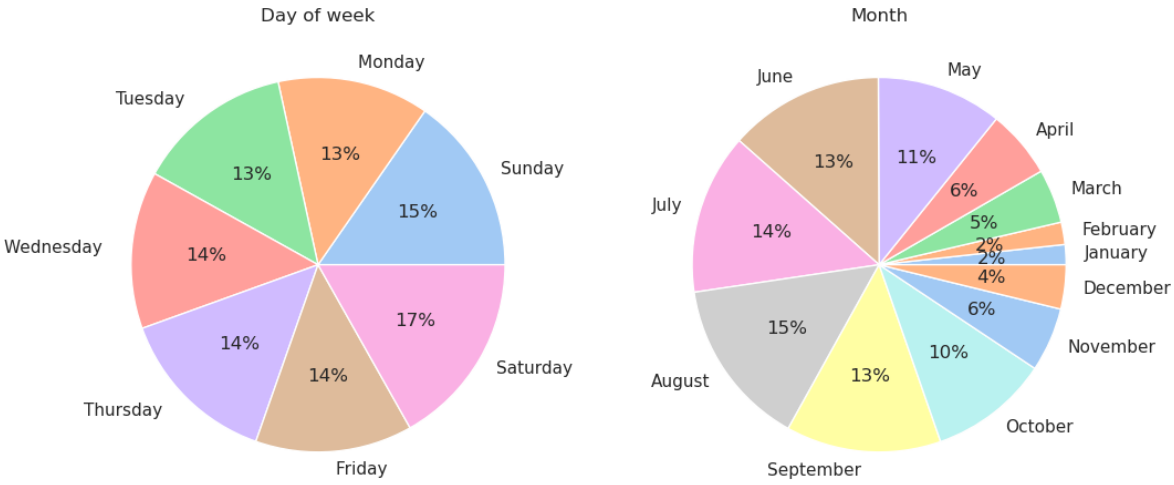

Рисунок 12 – Распределение по дням недели и месяцам

Можно заметить, что день недели лишь незначительно влияет на количество поездок. С другой стороны, месяц оказывает сильное влияние на количество поездок – более тёплые месяца лучше способствуют поездкам на велосипеде. Построим то же распределение в виде гистограммы (рисунок 13).

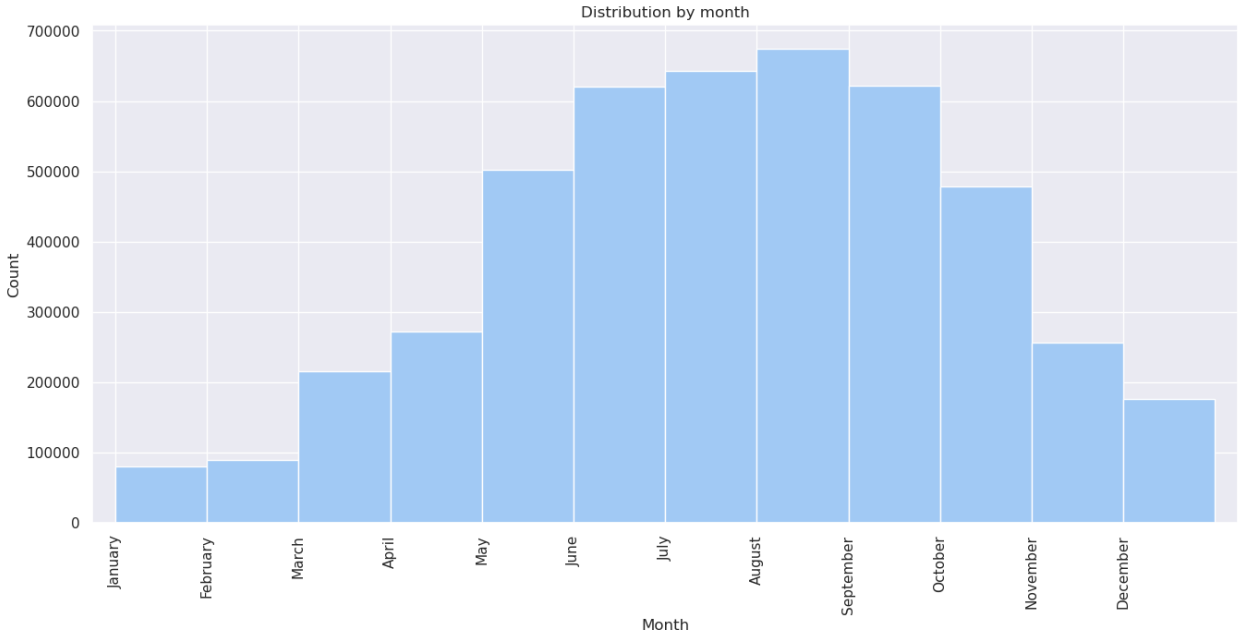

Рисунок 13 – Распределение поездок по месяцам

Из такого графика сразу видно, что количество поездок распределено по нормальному закону распределения.

Так как датасет содержит географические данные, логичным шагом является отметить их на карте. Было определено, что все поездки итогового датасета происходили в Чикаго. Воспользуемся модулем geopandas для изображения очертаний города на графике. Отметим точки начала поездок синими знаками “+”, а точки окончания – оранжевыми знаками “x” (рисунок 14).

Большая часть поездок сконцентрирована в северо-восточной – восточной части города, в то время как южный и западный концы города практически не имеют у себя отметок.

Таким образом, был проведён разведочный анализ датасета, а также датасет был очищен и подготовлен к дальнейшему исследованию.

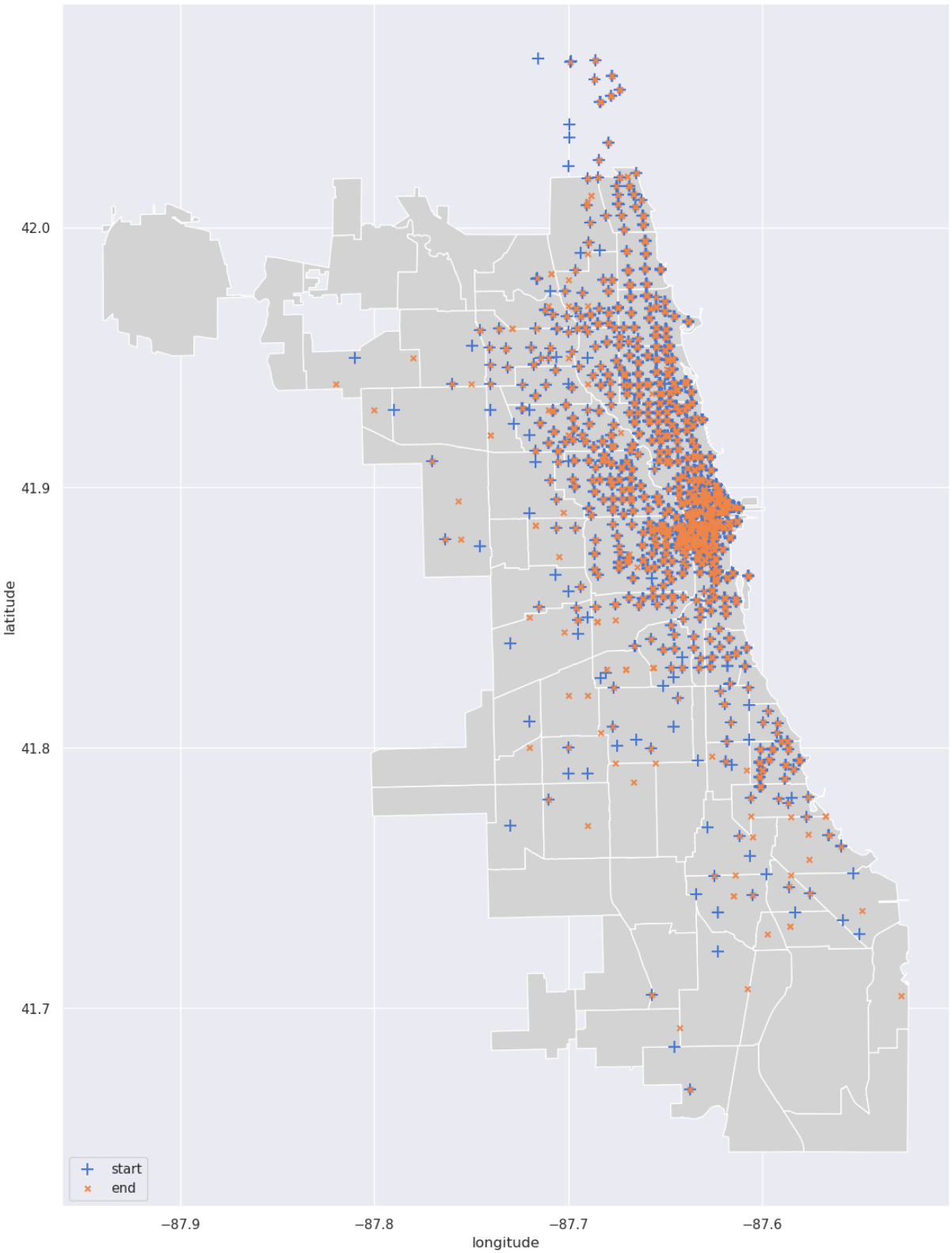

Рисунок 14 – Точки на карте города Чикаго

1.5 Выводы

В результате разведочного анализа были определены основные характеристики исследуемого датасета и выделены некоторые закономерности. Так, время года и день недели оказывают значимое влияние на число поездок на велосипедах; самым популярном видом велосипеда является классический; количество поездок примерно одинаково распределено между людьми, имеющими членство компании и теми, кто его не имеет; лишь 5% поездок заканчиваются на той же станции, что и начинаются.

В процессе анализа данные были подготовлены к дальнейшему исследованию датасета: исключены пропуски и выбросы, некоторые категориальные признаки преобразованы к бинарным, введены новые бинарные признаки на основе имеющихся данных, уделены ненужные поля.

2 АЛГОРИТМЫ МАШИННОГО ОБУЧЕНИЯ

Инструмент машинного обучения Apache Spark, библиотека MLlib стандартизирует API-интерфейсы для ML-алгоритмов, чтобы упростить объединение нескольких алгоритмов в один конвейер или рабочий процесс. Это реализовано с помощью следующих специальных структур данных и методов:

1) DataFrame – ML API, который использует DataFrame из Spark SQL в качестве датасета для машинного обучения, который может содержать различные типы данных, включая специфические для Machine Learning.

2) Преобразователь (Transformer) – алгоритм, который может преобразовывать один DataFrame в другой. Например, ML-модель – это трансформер, который преобразует DataFrame с предикторами в DataFrame с прогнозами. Трансформер можно представить в виде абстракции, которая включает преобразователи фичей и обученные модели. Технически Transformer реализует метод transform(), который преобразует один DataFrame в другой путем добавления одного или нескольких столбцов. К примеру, VectorAssembler является преобразователем, поскольку он принимает входный датафрейм и возвращает преобразованный с новым столбцом, который является векторным представлением всех функций.

3) Оценщик (Estimator) – алгоритм, который можно разместить в DataFrame для создания преобразователя. Например, алгоритм обучения – это оценщик, который обучается на DataFrame и создает ML-модель. Можно сказать, что оценщик — это высокоуровневая абстракция алгоритма обучения, который возвращает модель (преобразователь). Она, в свою очередь, преобразует датафрейм в соответствии с параметрами, которые исследуются на этапе подгонки (fitting) или обучения. Технически каждый оценщик реализует метод fit(), принимающий DataFrame и создающий ML-модель, которая имеет метод transform(). Например, алгоритм обучения LogisticRegression, является оценщиком, который возвращает преобразователь LogisticRegresionModel после исследования параметров данных.

4) Конвейер (Pipeline), который связывает несколько преобразователей и оценщиков в единый рабочий процесс машинного обучения. Apache Spark предоставляет класс, который формируется путем объединения различных этапов конвейера, т.е. Estimator'ов и Transformer'ов, выполняемых последовательно. В классе конвейера есть метод fit(), который запускает весь рабочий процесс. Он возвращает модель PipelineModel, которая имеет точно такое же количество этапов, что и конвейер, за исключением того, что все этапы оценщика заменяются соответствующим преобразователем, полученным во время выполнения. Эта модель конвейера может быть сериализована для повторного использования без затрат на настройку или обучение. Во время выполнения каждый этап вызывается последовательно, в зависимости от его типа (преобразователь или оценщик) вызываются соответствующие методы fit() или transform().

5) Параметр (Parameter) для задания настроечных параметров у преобразователей и оценщиков через общий API. Каждый экземпляр преобразователя или оценщика имеет уникальный идентификатор, который полезен при указании параметров.

2.1 Постановка задачи

Задачей данной главы работы является решение задач бинарной классификации и регрессии с использованием алгоритмов машинного обучения.

Будет производиться бинарная классификация по признаку round_trip (круговая поездка) с использованием алгоритма случайного леса. В качестве входных данных будут использованы координаты станций, длительность поездки, день и месяц поездки, а также тип велосипеда и наличие членства. Расстояние поездки не включается во входные данные так как было использовано для расчёта целевого признака.

Задача регрессии будет решаться при помощи алгоритма линейной регрессии. При этом целевым значением будет являться расстояние поездки, а в

качестве входных данных будут использоваться все признаки, что и в задаче бинарной классификации, а также признак того, что поездка является круговой.

2.2 Подготовка

Загрузим датасет, получившийся в результате разведочного анализа и разделим его на тренировочную и контрольную части.

```
df = spark.read.csv(data_path, inferSchema=True, header=True)
train, test = df.randomSplit([0.9, 0.1])
print(f'train: {train.count()}, test: {test.count()}')
```

Получившаяся тренировочная часть содержит 3,659,930 записей, контрольная – 406,988 записей.

2.3 Решение задачи бинарной классификации с использованием алгоритма случайного леса

Алгоритм случайного леса (Random Forest) — универсальный алгоритм машинного обучения, суть которого состоит в использовании ансамбля решающих деревьев. Само по себе решающее дерево предоставляет крайне невысокое качество классификации, но из-за большого их количества результат значительно улучшается. Также это один из немногих алгоритмов, который можно использовать в абсолютном большинстве задач.

В Spark алгоритм случайного леса представлен классом RandomForestClassifier. Основными параметрами алгоритма являются число деревьев и максимальная глубина. Первый определяет число деревьев в ансамбле, второй – максимальную высоту или глубину дерева – максимальный путь от корня дерева до его листа – узла, не имеющего дочерних узлов.

Так как алгоритм принимает в качестве входной колонки вектор значений, исходные данные необходимо преобразовать для передачи алгоритму. Для этого,

определим конвейер (pipeline), который подготовит входные данные для работы алгоритма и применит к ним сам алгоритм.

```
cl_pipeline = Pipeline(stages=[
    StringIndexer(inputCol='day_of_week', outputCol='day_idx'),
    StringIndexer(inputCol='month', outputCol='month_idx'),
    VectorAssembler(inputCols=['day_idx', 'month_idx', 'electric_bike', 'docked_bike', 'member'], outputCol='cat_features'),
    VectorIndexer(inputCol='cat_features', outputCol='cat_idx'),
    VectorAssembler(inputCols=['start_lat', 'start_lng', 'end_lat', 'end_lng', 'duration'], outputCol='num_features'),
    MinMaxScaler(inputCol='num_features', outputCol='num_scaled'),
    VectorAssembler(inputCols=['cat_idx', 'num_scaled'], outputCol='features'),
    RandomForestClassifier(labelCol='round_trip', featuresCol='features', numTrees=10)])
```

После этого, произведём обучение на тренировочной части датасета и получим предсказания для контрольной части.

```
cl_model = cl_pipeline.fit(train)
cl_predictions = cl_model.transform(test)
```

Посмотрим на получившиеся предсказания (рисунок 15).

Для оценки качества предсказаний определим количество истинно положительных, истинно отрицательных, ложно положительных и ложно отрицательных предсказаний, после чего вычислим метрики точности (precision) и отзыва (recall). Соберём результаты в датасет (рисунок 16).

По полученным результатам видно, что модель достаточно хорошо справляется с предсказанием отрицательных значений, однако лишь в 1 из 3 случаев правильно предсказывает положительные значениями. Об этом свидетельствует метрика отзыва (recall) равная 0.32 и показывающая, какую долю от общего числа положительных значений составляют верные положительные предсказания.

	features	prediction	true_value
0	[6.0, 0.0, 1.0, 0.0, 0.0, 0.5588333898818049, ...	0.0	0
1	[4.0, 0.0, 1.0, 0.0, 1.0, 0.5137975525088563, ...	0.0	0
2	[4.0, 0.0, 0.0, 0.0, 1.0, 0.7090793463413818, ...	0.0	0
3	[4.0, 0.0, 0.0, 0.0, 1.0, 0.7019606506651209, ...	0.0	0
4	[5.0, 0.0, 0.0, 0.0, 0.0, 0.585484475155062, 0...	0.0	0
...	...	...	...
427	[0.0, 1.0, 0.0, 0.0, 0.0, 0.17616403703634506,...	0.0	0
428	[1.0, 1.0, 0.0, 0.0, 1.0, 0.7196733296841562, ...	0.0	0
429	[1.0, 1.0, 0.0, 0.0, 0.0, 0.6063371378246533, ...	0.0	0
430	[1.0, 1.0, 1.0, 0.0, 0.0, 0.6210659449404514, ...	0.0	0
431	[1.0, 1.0, 1.0, 0.0, 0.0, 0.5817778298790744, ...	0.0	0

Рисунок 15 – Предсказания случайного леса

	metric	value
0	TP	5274.000000
1	FP	2315.000000
2	TN	388259.000000
3	FN	11140.000000
4	Precision	0.694953
5	Recall	0.321311
6	F1	0.439445

Рисунок 16 – Метрики классификации

Построим матрицу несоответствий, сопоставляющую предсказанные значения с реальными (рисунок 17). По матрице видно, что основную часть предсказаний составляют истинно негативные предсказания.

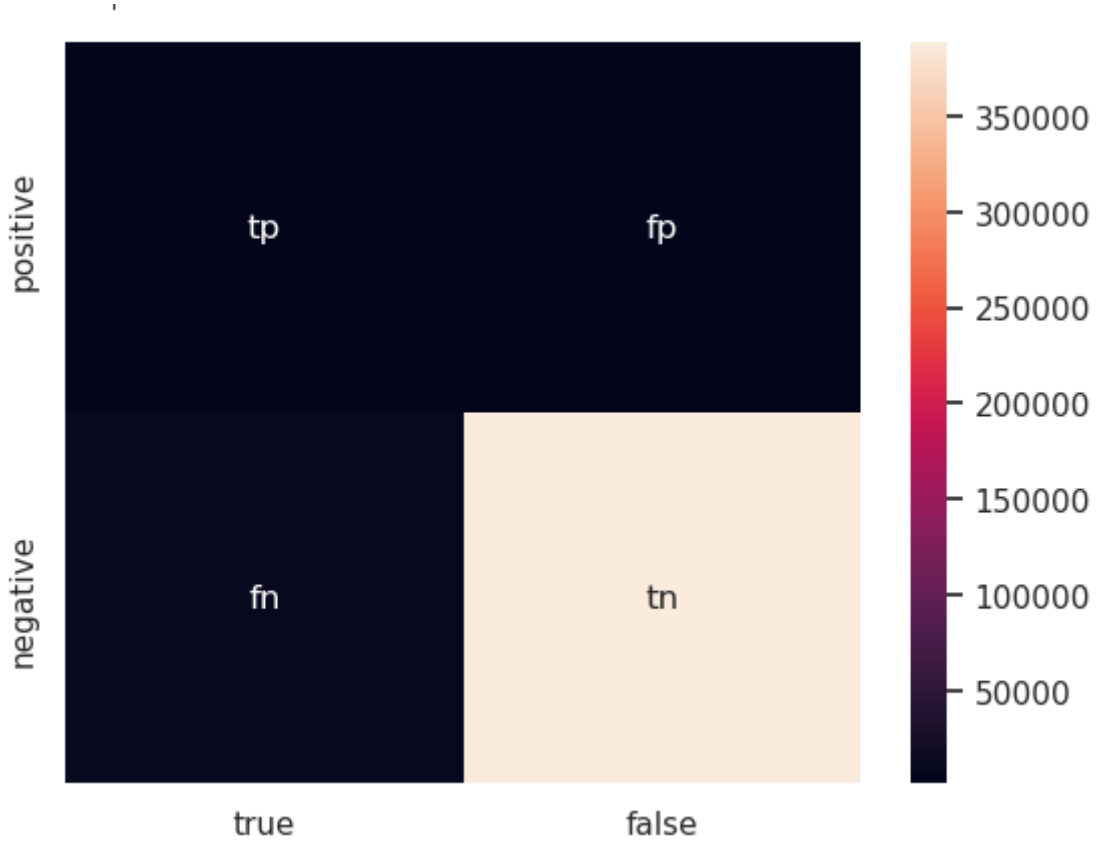

Рисунок 17 – Матрица несоответствий

ROC-кривая – график, позволяющий оценить качество бинарной классификации, отображает соотношение между долей объектов от общего количества носителей признака, верно классифицированных как несущие признак, и долей объектов от общего количества объектов, не несущих признака, ошибочно классифицированных как несущие признак при варьировании порога решающего правила.

Количественная интерпретация ROC даёт показатель AUC (англ. Area Under Curve, площадь под кривой) — площадь, ограниченная ROC-кривой и осью доли ложных положительных классификаций. Чем выше показатель AUC, тем качественнее классификатор, при этом значение 0,5 демонстрирует непригодность выбранного метода классификации (соответствует случайному гаданию). Значение менее 0,5 говорит, что классификатор действует с точностью до наоборот: если положительные назвать отрицательными и наоборот, классификатор будет работать лучше.

Используем BinaryClassificationEvaluator чтобы определить метрику ROC-AUC полученной модели.

```
cl_evaluator = BinaryClassificationEvaluator(labelCol='round_trip', rawPredictionCol="rawPrediction", metricName="areaUnderROC")
auc = cl_evaluator.evaluate(cl_predictions)
print(f'{auc=}')
```

[]: auc=0.8453152352819387

Значение ROC-AUC равное 0.85 свидетельствует о достаточно высокой точности полученной модели.

2.4 Решение задачи регрессии с использованием алгоритма линейной регрессии

Линейная регрессия — используемая в статистике регрессионная модель зависимости одной (объясняемой, зависимой) переменной y от другой или нескольких других переменных (факторов, регрессоров, независимых переменных) x с линейной функцией зависимости.

Модель линейной регрессии является часто используемой и наиболее изученной в эконометрике. А именно изучены свойства оценок параметров, получаемых различными методами при предположениях о вероятностных характеристиках факторов, и случайных ошибок модели. Предельные (асимптотические) свойства оценок нелинейных моделей также выводятся исходя из аппроксимации последних линейными моделями. С эконометрической точки зрения более важное значение имеет линейность по параметрам, чем линейность по факторам модели.

В Spark линейная регрессия представлена классом LinearRegression. Основными параметрами алгоритма являются максимальное число итераций, и параметр регуляризации, представляющий из себя величину штрафа за неверные предсказания.

Так же, как и для предыдущей задачи, определим конвейер и произведём обучение модели и получение предсказаний (рисунок 18).

```
reg_pipeline = Pipeline(stages=[
    StringIndexer(inputCol='day_of_week', outputCol='day_idx'),
    StringIndexer(inputCol='month', outputCol='month_idx'),
    VectorAssembler(inputCols=['day_idx', 'month_idx', 'electric_bike', 'docked_bike', 'member', 'round_trip'], outputCol='cat_features'),
    VectorIndexer(inputCol='cat_features', outputCol='cat_idx'),
    VectorAssembler(inputCols=['start_lat', 'start_lng', 'end_lat', 'end_lng', 'duration'], outputCol='num_features'),
    MinMaxScaler(inputCol='num_features', outputCol='num_scaled'),
    VectorAssembler(inputCols=['cat_idx', 'num_scaled'], outputCol='features'),
    LinearRegression(labelCol='distance', featuresCol='features', maxIter=10, regParam=0.3, elasticNetParam=0.8)])
reg_model = reg_pipeline.fit(train)
reg_predictions = reg_model.transform(test)
```

	features	prediction	true_value
0	[1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.6564408668755...	-815.759180	0.000000
1	(1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5605402196578...	2766.266643	2459.187861
2	[1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.7133784236897...	1836.266417	1597.472125
3	(1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5605402196578...	1539.018124	1900.508219
4	[1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.5780607611417...	2665.977707	3920.692148
5	(1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.6413052325463...	1487.523434	921.816245
6	(1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.6020635427168...	2312.758234	2464.847143
7	(1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5838032717227...	2327.997291	3131.936726
8	(1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.6911312988418...	1956.839945	1938.085599
9	[1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.6650362115608...	2279.600283	2100.911882

Рисунок 18 – Предсказания линейной регрессии

Можно заметить несколько достаточно близких значений, однако в целом заметны некоторые расхождения. Определим метрики среднеквадратичной ошибки и коэффициента детерминации.

Коэффициент детерминации (R^2) — это доля дисперсии зависимой переменной, объясняемая рассматриваемой моделью зависимости, то есть объясняющими переменными. Более точно — это единица минус доля необъяснённой дисперсии (дисперсии случайной ошибки модели, или условной по факторам дисперсии зависимой переменной) в дисперсии зависимой переменной. Его рассматривают как универсальную меру зависимости одной случайной величины от множества других.

Среднеквадратическая ошибка (RMSE) — часто используемая мера различий между значениями, предсказанными моделью, и наблюдаемыми значениями. RMSE представляет собой квадратный корень из второго момента выборки различий между предсказанными значениями и наблюдаемыми значениями или среднеквадратичное значение этих различий. Эти отклонения называются остатками (residuals).

RMSE — это квадратный корень из среднего квадрата ошибок. Влияние каждой ошибки на RMSE пропорционально величине квадрата ошибки; таким образом, большие ошибки оказывают непропорционально большое влияние на RMSE. Как следствие, RMSE чувствителен к выбросам.

```
lr = reg_model.stages[-1]
summary = lr.summary
print(f'RMSE: {summary.rootMeanSquaredError}, r2: {summary.r2}')
```

```
[]: RMSE: 913.0782371452432, r2: 0.4360709044311927
```

Коэффициент детерминации для данной модели равен 0.44, что означает, что модель объясняет 44% всей дисперсии данных. Это может свидетельствовать о плохом обучении модели, неправильно подобранных входных данных или о том, что линейная модель плохо подходит для предсказания данной величины.

2.5 Выводы

В результате решения задач классификации и регрессии были получены две модели машинного обучения. Модель случайного леса, используемая для определения, является поездка круговой или нет, достаточно хорошо справляется с предсказаниями. Модель линейной регрессии показывает посредственные результаты, и, вероятно, не подходит для описания зависимостей в данном датасете.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы был проведён анализ данных из набора данных Cyclistic. Был проведён разведочный анализ данных, позволивший определить основные закономерности в данных, очистить датасет от пропущенных значений и выбросов и подготовить его к дальнейшему анализу с использованием алгоритмов машинного обучения.

Были построены две модели машинного обучения. Модель бинарной классификации на основе алгоритма случайного леса позволяет определять, является ли поездка круговой. Модель регрессии на основе алгоритма линейной регрессии может предсказать расстояние поездки на основе остальных данных о ней.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Spark Documentation // Apache Spark. – Режим доступа : https://spark.apache.org/docs/latest/ (дата обращения 06.11.2022).

2. Python Documentation. – Режим доступа : https://docs.python.org/3.8/.

3. Matplotlib Documentation. – Режим доступа : https://matplotlib.org/.

4. Григорьев Е.А., Климов Н.С. Разведочный анализ данных с помощью Python // E-Scio. – 2020. – №2 (41). – Режим доступа : https://cyberleninka.ru/article/n/razvedochnyy-analiz-dannyh-s-pomoschyu-python.

5. Мамуров Б.Ж., Абдуллаев Ж.Ж. Регрессионный анализ как средство изучения зависимости между переменными // European science. – 2021. – №2 (58). – Режим доступа : https://cyberleninka.ru/article/n/regressionnyy-analiz-kak-sredstvo-izucheniya-zavisimosti-mezhdu-peremennymi.

6. What is linear regression? // IBM. – Режим доступа : https://www.ibm.com/topics/linear-regression.

7. Yiu T. Understanding Random Forest // Towards Data Science. – Режим доступа : https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

8. Classification: ROC Curve and AUC // Google Machine Learning Crash Course. – Режим доступа : https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc.