# 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 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.

## 2.2 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 1).

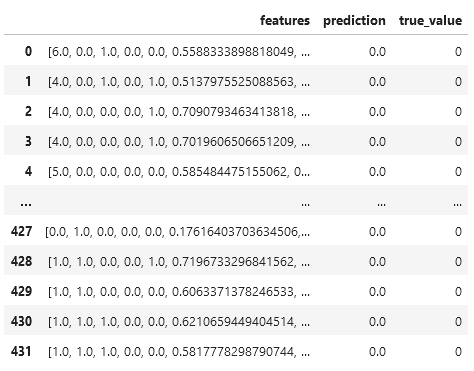


Рисунок 1 – Предсказания случайного леса

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 2).

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.

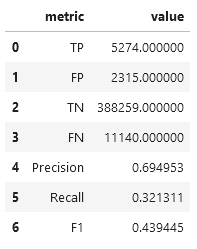


Рисунок 2 – Метрики классификации

Построим матрицу несоответствий, сопоставляющую предсказанные значения с реальными (рисунок 3). По матрице видно, что основную часть предсказаний составляют истинно негативные предсказания.

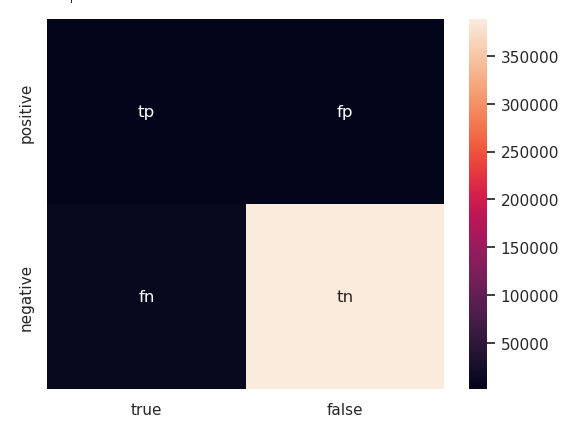


Рисунок 3 – Матрица несоответствий

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.

## 2.3 Решение задачи регрессии с использованием 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 4).

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)

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

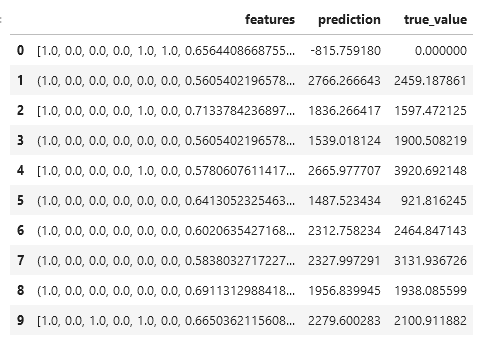


Рисунок 4 – Предсказания линейной регрессии

Коэффициент детерминации (R2) — 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.

## 2.4 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.