СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной работе необходимо выполнить исследование показаний энергопотребления со счетчиков. Выборка состоит 5 567 домов Лондона. Исследование необходимо провести с использованием технологии больших данных.

Цель данной работы – проведение разведочного анализа данных на датасете при помощи PySpark, а также анализ с помощью нижеперечисленных алгоритмов машинного обучения:

– задача регрессии;

– задача бинарной классификации.

Чтобы достичь цели необходимо решить следующие задачи:

– определить типы признаков;

– определить недопустимые значения (пропущенные значения и выбросы) и устранить их.

– рассчитать статистические показатели признаков;

– визуализировать распределения признаков;

– определить корреляцию между признаками;

– разбить выборку на обучающую и тестовую;

– построить модель для задачи регрессии (LinearRegression);

– построить модель для бинарной классификации (RandomForest).

– провести обучение и валидацию моделей;

В первом разделе работы:

– описывается датасет;

– определяются типы признаков;

– определяются и устраняются недопустимые значения;

– рассчитываются статистические показатели признаков;

– определяется корреляция между признаками;

– визуализируется распределение признаков.

Во втором разделе работы:

– осуществляется разбиение датасета на тестовый и тренировочный;

– описывается анализ с помощью задачи регрессии;

– описывается анализ с помощью задачи бинарной классификации.

– производится обучение и валидация моделей.

1 РАЗВЕДОЧНЫЙ АНАЛИЗ С ПОМОЩЬЮ PYSPARK

1.1 Постановка задачи

В данной работе необходимо выбрать датасет для анализа, в датасете должны быть представлены табличные данные, объемом от нескольких сотен мегабайт.

Требуется выполнить разведочный анализ выбранного датасета, для чего требуется определить:

– типы признаков в датасете;

– пропущенные значения и устранить их;

– выбросы и устранить их;

– статистические показатели признаков;

– корреляцию между признаками;

Также необходимо визуализировать распределение признаков.

1.2 Описание датасета

В данной работе использован Daily dataset из набора данных Smart meters in London (Умные счетчики в Лондоне) [1].

Датасета был составлен, чтобы лучше следить за потреблением энергии, поскольку правительство хочет, чтобы поставщики энергии установили умные счетчики в каждом доме в Англии, Уэльсе и Шотландии.

Развертывание счетчиков возглавляет Европейский союз, который попросил все правительства-члены рассмотреть интеллектуальные счетчики в рамках мер по модернизации энергоснабжения и борьбе с изменением климата.

После первоначального исследования британское правительство решило внедрить интеллектуальные счетчики в рамках своего плана по обновлению стареющей энергетической системы.

Daily dataset представляет собой версию данных из лондонского хранилища данных, которая содержит показания энергопотребления для выборки из 5 567 лондонских домохозяйств, принявших участие в проекте Low Carbon London под руководством UK Power Networks. (кВтч).

Колонки, содержащие в названии energy, представляют энергию, потребленная за последние 30 минут в кВт*ч.

Датасет содержит колонки со следующей информацией:

– идентификатор домашнего хозяйства (LCLid);

– дата измерения (day);

– медианное значение энергии (energy_median);

– среднее арифметическое значение энергии (energy_mean);

– максимальное значение энергии (energy_max);

– количество замеров энергии (energy_count);

– стандартное значение (energy_std);

– сумма энергии (energy_sum);

– минимальное значение энергии (energy_min).

1.3 Разведочный анализ

1.3.1 Определение типов признаков

Датасет в виде таблицы представлен на рисунке 1.

LCLid	day	energy_median	energy_mean	energy_max	energy_count	energy_std	energy_sum	energy_min
MAC000002	2012-10...	0.1385	0.15430...	0.88599...	46	0.19603...	7.09800...	0.0
MAC000002	2012-10...	0.18	0.23097...	0.933	48	0.19232...	11.0870...	0.076
MAC000002	2012-10...	0.158	0.27547...	1.085	48	0.27464...	13.223	0.07
MAC000002	2012-10...	0.131	0.2136875	1.164	48	0.22448...	10.257	0.07
MAC000002	2012-10...	0.145	0.20352...	0.991	48	0.18411...	9.769	0.087
MAC000002	2012-10...	0.148	0.22677...	0.784	48	0.18440...	10.8850...	0.067
MAC000002	2012-10...	0.166	0.22397...	0.897	48	0.17424...	10.751	0.102
MAC000002	2012-10...	0.134	0.17564...	0.82200...	48	0.12222...	8.43100...	0.072
MAC000002	2012-10...	0.23099...	0.36204...	1.4529999	48	0.29492...	17.3779...	0.073
MAC000002	2012-10...	0.385	0.51020...	2.1600001	48	0.33551...	24.4900001	0.23199...
MAC000002	2012-10...	0.39349...	0.39343...	1.126	48	0.19988...	18.885	0.094
MAC000002	2012-10...	0.1615	0.21843...	0.67099...	48	0.15072...	10.4849...	0.07
MAC000002	2012-10...	0.186	0.32368...	1.8890001	48	0.32960...	15.5370...	0.076
MAC000002	2012-10...	0.176	0.27349...	0.917	48	0.20554...	13.1279...	0.073
MAC000002	2012-10...	0.1885	0.31385...	0.962	48	0.25519...	15.0649...	0.072
MAC000002	2012-10...	0.319	0.35179...	1.228	48	0.25632...	16.886	0.075
MAC000002	2012-10...	0.313	0.40893...	1.173	48	0.32067...	19.6289999	0.072
MAC000002	2012-10...	0.1755	0.26622...	1.058	48	0.24881...	12.7790...	0.073

Рисунок 1 – Колонки датасета и их значения

Типы данных в датасете:

– string;

– timestamp;

– integer;

– double.

В данном датасете больше всего double типов данных, как видно из рисунка 2.

В машинном обучении существует несколько типов признаков:

– количественный;

– порядковый (конечное упорядоченное множество);

– бинарный (множество ограниченное 0 и 1);

– категориальный (конченое множество).

У всех признаков один тип – количественный. Количественные тип признаков представляет собой множество действительных чисел.

```
root
 |-- LCLid: string (nullable = true)
 |-- day: timestamp (nullable = true)
 |-- energy_median: double (nullable = true)
 |-- energy_mean: double (nullable = true)
 |-- energy_max: double (nullable = true)
 |-- energy_count: integer (nullable = true)
 |-- energy_std: double (nullable = true)
 |-- energy_sum: double (nullable = true)
 |-- energy_min: double (nullable = true)
```

Рисунок 2 – Типы данных датасета

1.3.2 Поиск и удаление лишних признаков

Избавимся от столбцов LCLid и day, поскольку они содержат идентификатор и дату замера (рисунок 3).

energy_median	energy_mean	energy_max	energy_count	energy_std	energy_sum	energy_min
0.1385	0.15430...	0.88599...	46	0.19603...	7.09800...	0.0
0.18	0.23097...	0.933	48	0.19232...	11.0870...	0.076
0.158	0.27547...	1.085	48	0.27464...	13.223	0.07
0.131	0.2136875	1.164	48	0.22448...	10.257	0.07
0.145	0.20352...	0.991	48	0.18411...	9.769	0.087
0.148	0.22677...	0.784	48	0.18440...	10.8850...	0.067
0.166	0.22397...	0.897	48	0.17424...	10.751	0.102
0.134	0.17564...	0.82200...	48	0.12222...	8.43100...	0.072
0.23099...	0.36204...	1.4529999	48	0.29492...	17.3779...	0.073
0.385	0.51020...	2.1600001	48	0.33551...	24.4900001	0.23199...
0.39349...	0.39343...	1.126	48	0.19988...	18.885	0.094
0.1615	0.21843...	0.67099...	48	0.15072...	10.4849...	0.07
0.186	0.32368...	1.8890001	48	0.32960...	15.5370...	0.076
0.176	0.27349...	0.917	48	0.20554...	13.1279...	0.073
0.1885	0.31385...	0.962	48	0.25519...	15.0649...	0.072
0.319	0.35179...	1.228	48	0.25632...	16.886	0.075
0.313	0.40893...	1.173	48	0.32067...	19.6289999	0.072
0.1755	0.26622...	1.058	48	0.24881...	12.7790...	0.073
0.22749...	0.29085...	0.893	48	0.22577...	13.9610...	0.07
0.2685	0.37129...	1.4450001	48	0.31837...	17.8220001	0.085

Рисунок 3 – Датасет без столбцов LCLid и day

1.3.3 Поиск и устранение пропущенных начений

Обычно, пропущенные значения заносятся в таблицу, как None, NaN или NULL.

При поиске было обнаружено более 11 тыс. пропусков, как можно заметить на рисунке 4. 3510434

Способы устранить пропущенные значения:

– заполнение медианными значением

– заполнение средними значением;

– заполнение при помощи линейной регрессии;

– заполнение на основе соседних клеток;

– удаление строк с пропущенными значениями.

В датасете представлено около 3510434 строк, удаление 11 тыс., то есть менее 1%, не повлияет на достоверность будущих предсказаний и не увеличит погрешность. На рисунках 4 и 5 продемонстрировано устранение пропущенных значений.

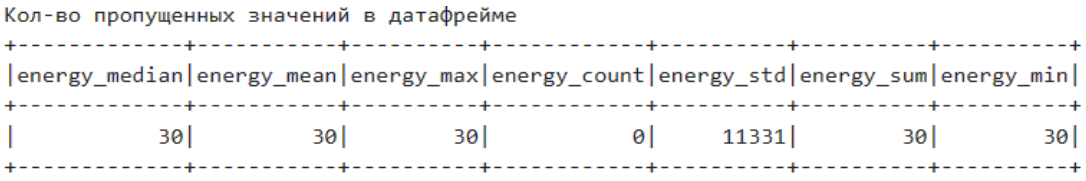

Рисунок 4 – Количство пропущенных значений по столбцам

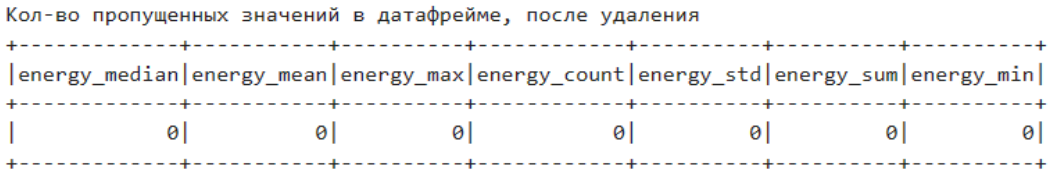

Рисунок 5 – Демонстрация устраненных пропущенных значений

1.3.4 Поиск выбросов и их устранение

Выброс – это наблюдение, удаленное от других в выборке. Другими словами, это Наблюдение, которое расходится с общей закономерностью Выборки [2].

Выбросы могут появляться из-за некорректно собранных данных, новых процессов или различных методов сборки данных.

Определим 75% и 25% квартиль. Это можно сделать с помощью approxQuantile(). Далее определим границы для каждого признака, за которыми будут находится выбросы.

Количественное отображение выбросов можно увидеть на рисунке 6. Также, выбросы продемонстрированы на ящиках с усами (BoxPlot) на рисунке 7.

Все что выше максимума или ниже минимума является выбросами Проверку того, что первоначальные выбросы успешно удалены, можно заметить на рисунке 8.

А на рисунке 9 продемонстрированы графики с данными после устранения. Для построения ящиков с усами была использована библиотека Pandas с использованием метода Sample, который позволяет взять часть из датасета.

Кол-во выбросов по столбцам:

energy_median_out	energy_mean_out	energy_max_out	energy_count_out	energy_std_out	energy_sum_out	energy_min_out
224973	201317	121012	29750	158187	201339	221460

Рисунок 6 – Количество выбросов по столбцам

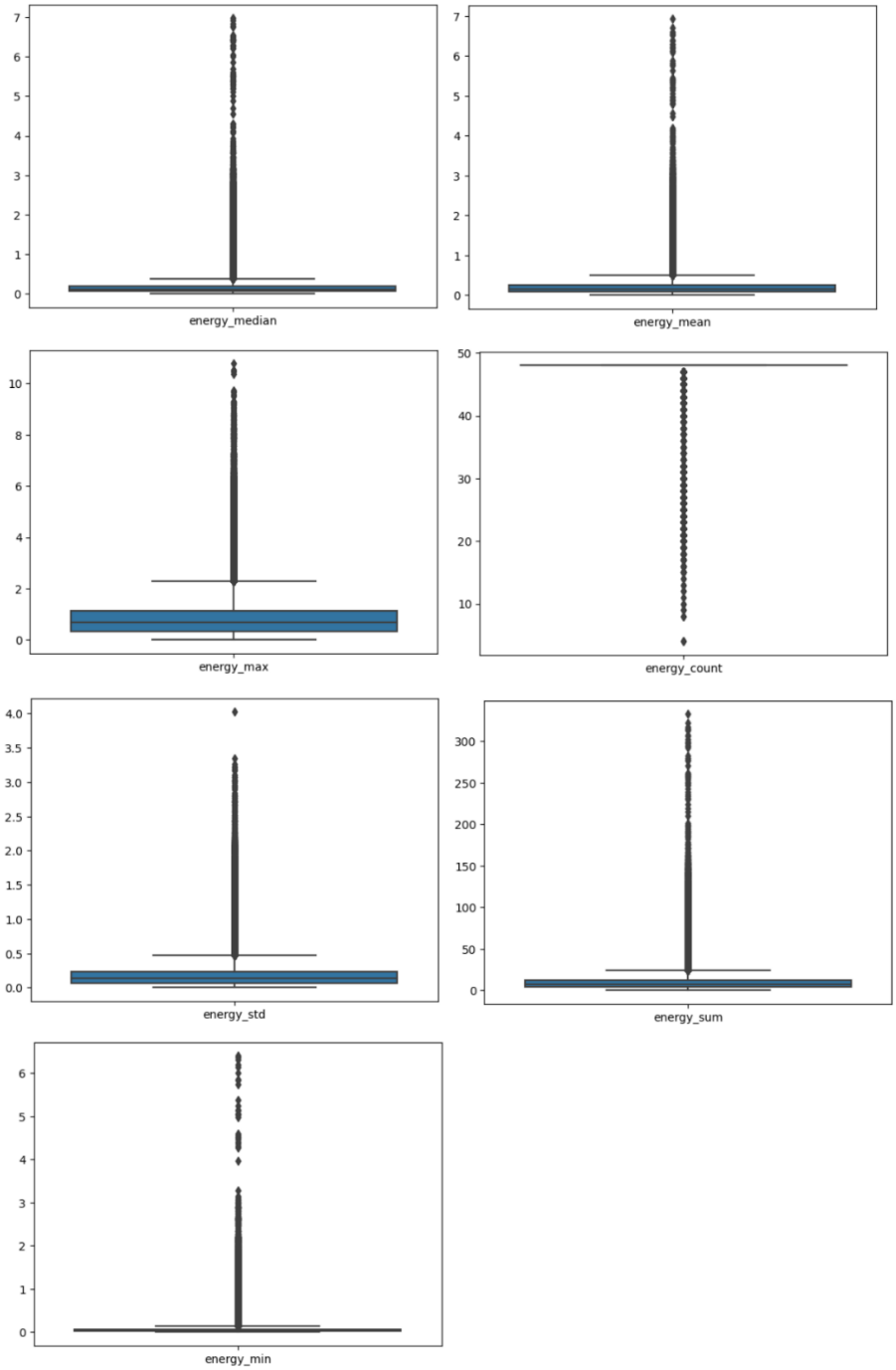

Рисунок 7 – Ящики с усами для наблюдения выбросов

```
Кол-во выбросов по столбцам, после удаления выбросов:
+-----------------+---------------+--------------+----------------+--------------+--------------+--------------+
|energy_median_out|energy_mean_out|energy_max_out|energy_count_out|energy_std_out|energy_sum_out|energy_min_out|
+-----------------+---------------+--------------+----------------+--------------+--------------+--------------+
|                0|              0|             0|               0|             0|             0|             0|
+-----------------+---------------+--------------+----------------+--------------+--------------+--------------+
```

Рисунок 8 – Подтверждение удаления выбросов

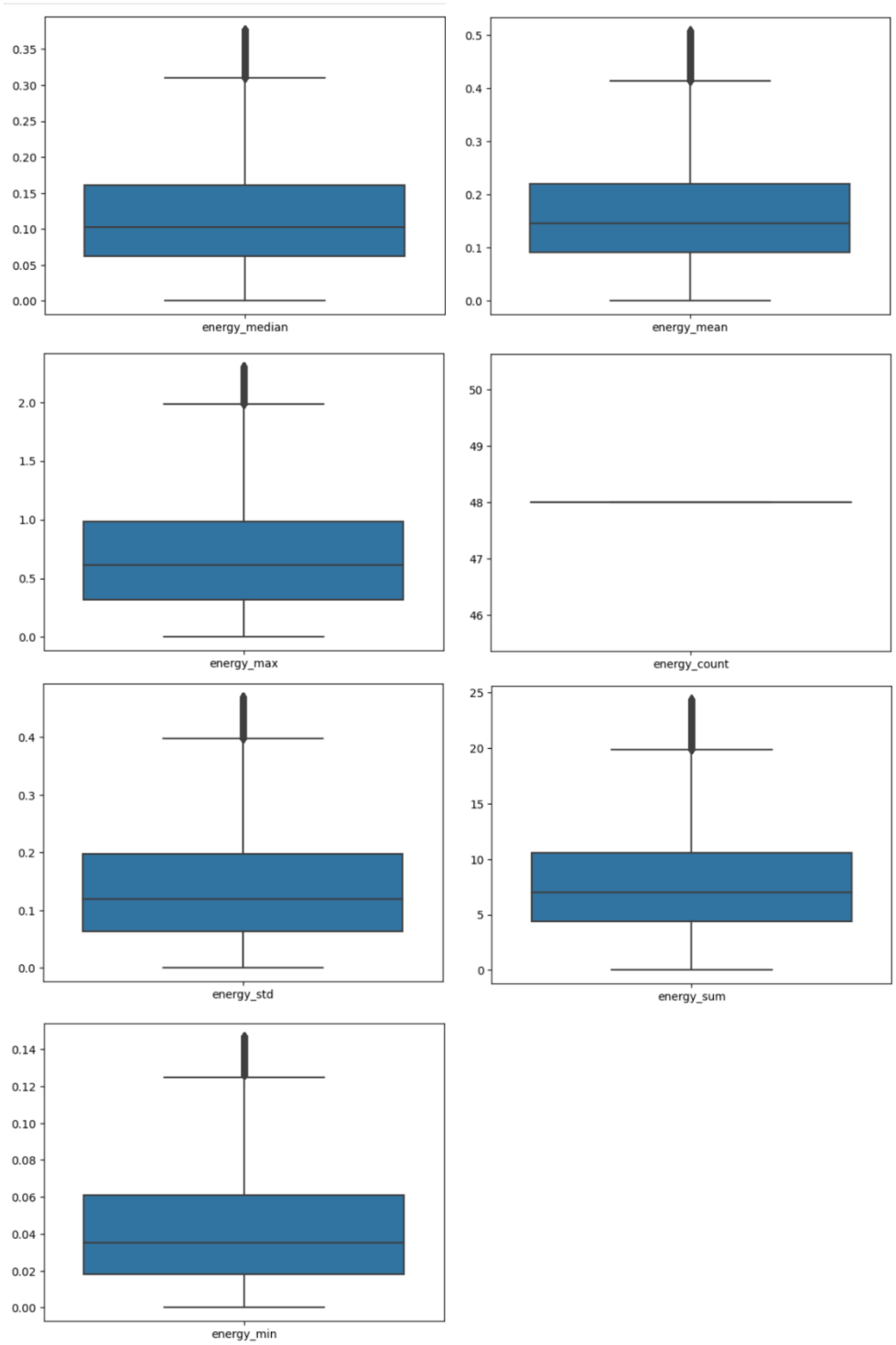

Рисунок 9 – Графики для визуального наблюдения отсутствия выбросов

1.3.5 Построение матрицы корреляции

Матрица корреляций на основе датасета показана на рисунке 10. В ней можно заметить, признаки столбца energy_count не коррелируют ни с одним другим признаком других столбцов, столбец содержит в себе одинаковые признаки и никак не меняется, в связи с чем является бесполезным. Данный столбец может быть удален.

energy_median	energy_mean	energy_max	energy_count	energy_std	energy_sum	energy_min
1.0	0.8862012883616622	0.49418765298674017	NaN	0.5149833234915282	0.8862012883616593	0.6829627699214574
0.8862012883616622	1.0	0.7476251232286238	NaN	0.8145762056794084	1.0000000000000926	0.621496713660836
0.49418765298674017	0.7476251232286238	1.0	NaN	0.9454504347866349	0.7476251232285466	0.2853915633221015
NaN	NaN	NaN	1.0	NaN	NaN	NaN
0.5149833234915282	0.8145762056794084	0.9454504347866349	NaN	1.0	0.814576205679333	0.24380361093410893
0.8862012883616593	1.0000000000000926	0.7476251232285466	NaN	0.814576205679333	1.0	0.6214967136607891
0.6829627699214574	0.621496713660836	0.2853915633221015	NaN	0.24380361093410893	0.6214967136607891	1.0

Рисунок 10 – Матрица корреляций

1.3.6 Показатели признаков

В статистичекие данные, представленные на рисунке 11 входят:

– средне;

– минимальное;

– максимальное;

– стандартное отклонение;

– квантили.

По столбцу count видно, что после обработки данных, осталось еще больше 3 млн. строк, из чего следует, что было удалено около 14% процента данных, в основном эти проценты составляют выбросы.

	count	mean	std	min	25%	50%	75%	max
energy_median	3053252.0	0.118768	0.074528	0.0	0.062000	0.103000	0.161000	0.377000
energy_mean	3053252.0	0.163161	0.094318	0.0	0.090979	0.146500	0.220083	0.508542
energy_max	3053252.0	0.695165	0.465291	0.0	0.316000	0.612000	0.983000	2.303000
energy_std	3053252.0	0.140305	0.097866	0.0	0.063570	0.118744	0.197008	0.469602
energy_sum	3053252.0	7.831735	4.527253	0.0	4.367000	7.032000	10.564000	24.410000
energy_min	3053252.0	0.042886	0.032184	0.0	0.018000	0.035000	0.061000	0.147000

Рисунок 11 –Статистические данные

1.3.7 Визуализация графиков распределения

Данные, визуализированные с помощью гистограмм, представлены на рисунке 12. Можно заметить, что все графики имеют вид логнормального распределения.

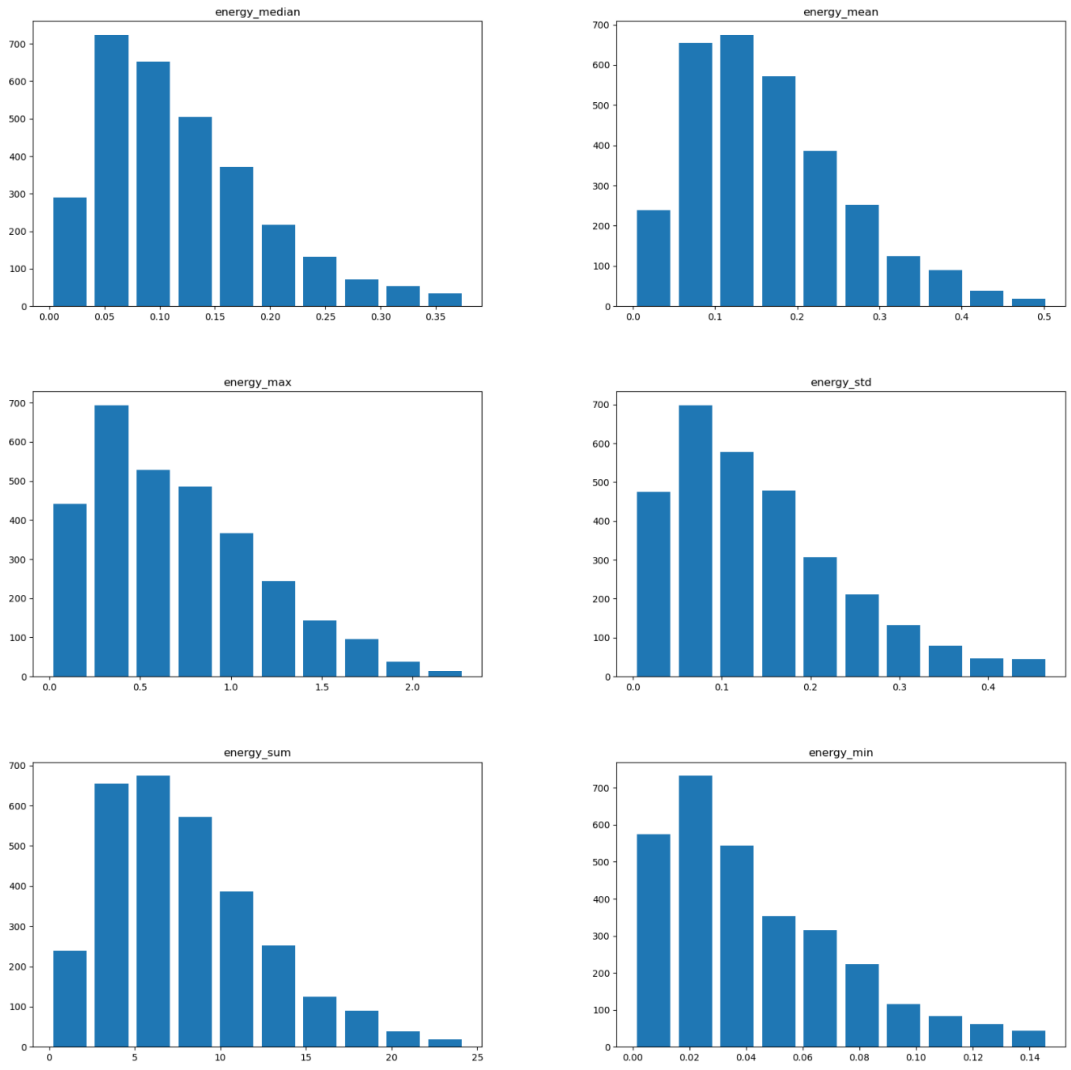

Рисунок 12 – Графики распределения

1.4 Выводы

В данном разделе был выбран Daily dataset, для которого был произведен разведочный анализ данных. В процессе работы было произведено знакомство с инструментом Apache Spark и его возможностями по обработке больших данных. Были найдены типы признаков, обнаружены и устранены пропущенные значения и выбросы, рассчитаны статистические показатели признаков, визуализированы распределения признаков и найдены корреляции между признаками. В результате данной главы были подготовлены данные для дальнейшего машинного обучения.

2 МАШИННОЕ ОБУЧЕНИЕ

2.1 Постановка задачи

В данной работе необходимо выполнить анализ датасета полученного после обработки в 1 этапе с помощью алгоритмов машинного обучения, таких как:

– задача регрессии (LinearRegression);

– задача бинарной классификации (RandomForest).

На вход подается датасет, где все признаки описывают параметры потребления электроэнергии.

Также необходимо:

– выполнить обучение и валидацию модели;

– рассчитать значения метрик классификации и регрессии;

– выполнить подбор гиперпараметров моделей по сетке.

После чего, модель регрессии сможет предсказывать величину стандартной погрешности в показателях электросчетчиков, а модель классификации – бинаризованное значение этого же показателя.

2.2 Подготовка данных для машинного обучения

2.2.1 Выделение бинарного признака

Поскольку для анализа датасета одним из методов является задача классификации, то необходим бинарный признак в данном датасете. Бинарный признак — это признак, в котором множество допустимых значений ограничено 0 и 1.

Однако в датасете нет бинарных признаков, для его создания возьмем среднее значения признака и значения больше среднего приравнять к 1, а ниже к 0.

За основу бинарного признака был взят количественный признак energy_max, поскольку у energy_max и energy_std корреляция самая большая – 0.945 (рисунок 10). На рисунке 13 показан датасет с бинарным признаком.

energy_median	energy_mean	energy_std	energy_max	energy_sum	energy_min	binary_std
0.1415	0.29616666875000003	0.2814713178628203	1.1160001	14.216000100000002	0.031	1
0.1015	0.1898125	0.1884046862418033	0.685	9.111	0.064	1
0.114	0.2189791666666666	0.20291927853038208	0.6759999999999999	10.510999999999996	0.065	1
0.191	0.32597916666666665	0.2592049619947409	0.7879999999999999	15.646999999999998	0.066	1
0.21800000000000005	0.3575	0.28759657027517305	1.077	17.16	0.066	1
0.1305	0.2350833333333333	0.2220696491599295	0.705	11.284	0.066	1
0.08900000000000001	0.22135416666666666	0.26723887549908265	1.094	10.625	0.062	1
0.16049999999999998	0.291125	0.24907604794434665	0.7490000000000001	13.973999999999998	0.065	1
0.2175	0.33918750000000003	0.26310119857478675	0.866	16.281000000000002	0.069	1
0.14950000000000002	0.2617083333333333	0.2447927441503373	0.838	12.562000000000001	0.066	1

Рисунок 13 – Датасет с бинарным признаком

2.2.2 Разделение данных

Данные были разделены, в соотношении 7:3, 70% – обучающая выборка, 30% – тестовая, что показано на рисунке 14.

```
splits = data.randomSplit([0.7, 0.3])
train = splits[0]
test = splits[1]
train_raws = train.count()
test_raws = test.count()
print("Обучающая:", train_raws, " Тестирующая:", test_raws)
```

```
Обучающая: 2137043  Тестирующая: 916209
```

Рисунок 14 – Разделение данных на разные выборки

2.3 Обучение моделей

Алгоритмы машинного обучения, как правило, работают лучше или сходятся быстрее, когда различные функции (переменные) имеют меньший масштаб.

Поэтому перед обучением на них моделей машинного обучения данные обычно нормализуются. Для этого данные сначала были преобразованы в единый вектор при помощи:

numVect = VectorAssembler(inputCols = numIn, outputCol="num_features")

где inputCols

numIn = ["energy_mean", "energy_max", "energy_median", "energy_sum", "energy_min"]

После чего данный вектор был нормализован:

minMax = MinMaxScaler(inputCol = numVect.getOutputCol(), outputCol="features")

Для задачи регрессии был применен алгоритм линейной регрессии, где в качестве предсказываемого признака был установлен energy_std, а в качестве признаков, на основе которых будет вестись предсказание-вектор, полученный ранее features:

Regression = LinearRegression (labelCol = "energy_std", featuresCol = "features", maxIter=2, regParam=0.8, elasticNetParam=0.0)

Далее мы указываем порядок обработки объектов конвейеру:

regressionPipeline = Pipeline(stages=[numVect, minMax, regresion])

После чего производиться само обучение:

regressionPiplineModel = regressionPipeline.fit(train)

Для задачи бинарной классификации был применен метод прогнозирования – случайный лес, порядок действий такой же, как и для задачи регрессии, рассмотренной выше:

classifier = RandomForestClassifier (labelCol = "binary_std", featuresCol = "features", maxDepth=2, maxBins=5, numTrees=5)

classifierPipeline = Pipeline(stages=[numVect, minMax, classifier])

classifierPiplineModel = classifierPipeline.fit(train)

В задании для модели регрессии была выбрана задача Линейной регрессии, а для модели классификации задача Случайного леса.

Параметры при обучении модели регрессии:

1 MaxIter – 2;

2 regParam – 0.8;

3 elasticNetParam – 0.0.

Параметры при обучении модели классификации:

1 MaxDepth – 2;

2 maxBins – 5;

3 numTrees – 5.

2.3.1 Задача линейной регрессии

Для определения точности предсказаний модели регрессии использованы метрики R2 и RMSE.

Коэффициент детерминации или R2 требуется для измерения количества отклонений в прогнозах, объясненных набором данных. По сути, это разница между выборками в наборе данных и прогнозами, сделанными моделью.

Если значение R2 равно 1, это означает, что модель хорошо себя показала, а если ее значение равно 0, это означает, что модель будет плохо работать с неизвестным датасетом.

Среднеквадратичная ошибка или RMSE это мера различий между значениями (выборочными или популяционными), предсказанными моделью или оценщиком, и наблюдаемыми значениями. RMSE всегда неотрицательно, и значение 0 указывает на идеальное соответствие данным. Как правило, более низкое RMSE лучше, чем более высокое [5].

Модель регрессии продемонстрировала следующие значения метрик:

– RMSE – 0.0773192;

– R2 – 0.376467.

Так как предсказываемый признак имеет среднее значение в виде 7.2, то среднеквадратичная ошибка близкая к 0, это наталкивает на мысль, что данные хорошо соответствуют, но в то же время мы имеем R2 равный 0.376, что означает, что данная модель будет предсказывать посредственно при наборе неизвестных данных. На рисунке 15 можно увидеть векторные

предсказания модели линейной регрессии. В данном случае кросс-валидация будет полезна.

features	prediction	energy_std
[0.0,2.048340835525027E-4,0.0021710811984368217,2.048340835525027E-4,0.0]	0.09648022874121527	7.216878364870322E-4
[0.0,4.096681671050053E-4,0.00303951367781155,4.0966816710500537E-4,0.0]	0.09653284659146416	0.0010907411755575583
[0.0,4.096681671050053E-4,0.00303951367781155,4.0966816710500537E-4,0.0]	0.09653284659146416	0.0010907411755575583
[0.0,4.096681671050053E-4,0.00303951367781155,4.0966816710500537E-4,0.0]	0.09653284659146416	0.0010907411755575583
[0.0,4.0966816710500537E-4,0.0034737299174989146,4.096681671050054E-4,0.0]	0.09655163862452776	0.001184264442387572
[0.0,4.0966816710500537E-4,0.004342162396873643,4.096681671050054E-4,0.0]	0.09658922269065495	0.0014433756729740645
[0.0,4.5063498381550595E-4,0.0034737299174989146,4.50634983815506E-4,0.0]	0.0965546453813521	0.0012245638968990768
[0.0,4.5063498381550595E-4,0.0034737299174989146,4.50634983815506E-4,0.0]	0.0965546453813521	0.0012245638968990768
[0.0,4.5063498381550595E-4,0.0034737299174989146,4.50634983815506E-4,0.0]	0.0965546453813521	0.0012245638968990768
[0.0,4.5063498381550595E-4,0.0034737299174989146,4.50634983815506E-4,0.0]	0.0965546453813521	0.0012245638968990768
[0.0,4.5063498381550595E-4,0.0034737299174989146,4.50634983815506E-4,0.0]	0.0965546453813521	0.0012245638968990768
[0.0,4.5063498381550595E-4,0.0034737299174989146,4.50634983815506E-4,0.0]	0.0965546453813521	0.0012245638968990768
[0.0,4.5063498381550595E-4,0.0034737299174989146,4.50634983815506E-4,0.0]	0.0965546453813521	0.0012245638968990768

Рисунок 15 – Результат работы модели регрессии

2.3.2 Задача рандомного леса

Для определения точности предсказаний модели классификации была использована метрика AUR и матрица ошибок.

AUR обеспечивает совокупный показатель производительности по всем возможным пороговым значениям классификации. Один из способов интерпретации AUR — это вероятность того, что модель ранжирует случайный положительный пример выше, чем случайный отрицательный пример. Значение AUR варьируется от 0 до 1.

Матрица ошибок — это таблица, которая позволяет визуализировать эффективность алгоритма классификации путем сравнения прогнозируемого значения целевой переменной с ее фактическим значением. Столбцы матрицы представляют наблюдения в прогнозируемом классе, а строки — наблюдения в фактическом классе.

Модель классификации показала следующие результаты, по которым можно судить точность ее предсказаний:

– AUR – 0.956175;

– точность (Precision) – 0.9159;

– отзыв (Recall) – 0.8988;

– F1 – 0.9073.

Судя по показанию метрик, модель достаточно хорошо обучена и способна предсказывать данные с высокой точностью.

На рисунке 16 представлены векторные предсказания модели классификации.

features	prediction	binary_std
[0.0,0.16341663185818664,0.11506730351715155,0.16341663185818667,0.0]	0.0	0
[0.0,0.17722244908962534,0.29179331306990886,0.17722244908962537,0.0]	0.0	1
[0.0,0.17959852445883437,0.24099001302648723,0.17959852445883437,0.0]	0.0	1
[0.0,0.19455141255816708,0.19105514546244032,0.19455141255816708,0.0]	0.0	0
[0.0,0.20618598850394917,0.331306990881459,0.20618598850394926,0.0]	1.0	1
[0.0,0.25714870849181193,0.22579244463742945,0.25714870849181193,0.0]	0.0	1
[0.0,0.3700532553459514,0.3538862353452019,0.3700532553459514,0.0]	1.0	1
[0.0,0.40319541416142796,0.723404255319149,0.40319541416142796,0.0]	1.0	1
[0.0,0.47189676168825573,0.33651758575770735,0.4718967616882557,0.0]	1.0	1
[0.0,0.5685374823083266,0.3933999131567521,0.5685374823083266,0.0]	1.0	1
[0.0,0.5747234716316121,0.5236647850629613,0.5747234716316121,0.0]	1.0	1
[0.0,0.5767718083704556,0.881893139383413,0.5767718083704557,0.0]	1.0	1
[0.0,0.5791888546530567,0.4515848892748589,0.5791888546530568,0.0]	1.0	1
[0.0,0.5874641434352144,0.8931827616152845,0.5874641434352146,0.0]	1.0	1

Рисунок 16 – Результат работы модели бинарной классификации

2.4 Кросс-валидация моделей

K-кратная перекрестная проверка выполняет выбор модели путем разделения набора данных на набор непересекающихся случайным образом разделенных сгибов, которые используются в качестве отдельных обучающих и тестовых наборов данных, например, с k = 3 раза перекрестная проверка с K-кратностью будет генерировать 3 пары наборов данных, каждая из которых использует 2/3 данных для обучения и 1/3 для тестирования. [3] Каждый фолд используется в качестве тестового набора ровно один раз [4].

При использовании кросс-валидации для модели задается некоторое количество значений для гиперпараметров.

Гиперпараметры – это параметры, значения которых контролируют процесс обучения и определяют значения параметров модели, которые в конечном итоге изучает алгоритм обучения. Префикс «гипер» предполагает, что это параметры «верхнего уровня», которые контролируют процесс обучения и параметры модели, которые являются его результатом.

При валидации модели регрессии было задано по 3 значения для трех гиперпараметров с использованием двух фолдов, что показано на рисунке 17. С такими входными данными, обучение модели будет проходить 54 раза, после чего будет выявлена лучше обученная модель.

```
regressionGrid = ParamGridBuilder()

regressionGrid = regressionGrid.addGrid(regresion.maxIter, [10, 20, 30])
regressionGrid = regressionGrid.addGrid(regresion.regParam, [0.1, 0.2, 0.3])
regressionGrid = regressionGrid.addGrid(regresion.elasticNetParam, [0.2, 0.5, 0.8])

regressionGrid = regressionGrid.build()

regressionValidation = CrossValidator(estimator = regressionPipeline,
                      evaluator = regressionEvaluatorR2,
                      estimatorParamMaps = regressionGrid,
                      numFolds = 2)
```

Рисунок 17 – Кросс–валидация для модели регрессии

После процесса кросс-валидации, была обучена модель с лучшим набором гиперпараметров. Метрики представлены на рисунке 18.

Сравним метрики:

– RMSE теперь равна 0.0515972 (было 0.0773192);

– R2 теперь равна 0.722325 (было 0.376467).

RMSE слегка улучшила свой показатель, но при этом метрика R2 теперь имеет значение превышающее предыдущее почти в 2 раза. В связи с чем модель теперь лучше будет предсказывать при наборе неизвестных данных.

```
Модель линейной регрессии с параметрами MaxIter = 10, RegParam = 0.1, ElasticNet = 0.2
RMSE(Среднеквадратичная ошибка) = 0.0515972
R2(Коэффициент детерминации) = 0.722325
```

Рисунок 18 – Результаты кросс–валидации линейной регрессии

При валидации модели классификации было задано по 3 значения для трех гиперпараметров с использованием двух фолдов, что показано на рисунке 19.

С такими входными данными, обучение модели будет проходить 54 раза, после чего будет выявлена лучше обученная модель.

```
classifierGrid = classifierGrid.addGrid(classifier.maxDepth, [2, 10, 5])
classifierGrid = classifierGrid.addGrid(classifier.maxBins, [5, 10, 20])
classifierGrid = classifierGrid.addGrid(classifier.numTrees, [5, 20, 50])

classifierGrid = classifierGrid.build()

classifierValidation = CrossValidator(estimator = classifierPipeline,
                                      evaluator = classifierEvaluatorAUR,
                                      estimatorParamMaps = classifierGrid,
                                      numFolds = 2)
```

Рисунок 19 – Кросс–валидация для модели классификации

После процесса кросс-валидации, была обучена модель с лучшим набором гиперпараметров. Для проверки воспользуемся матрицей ошибок и метрикой AUR:

– AUR при новых данных 0.994462 (было 0.956175);

– precision при новых данных 0.9519 (было 0.9159);

– recall при новых данных 0.9540 (было 0.8988);

– F1 при новых данных 0.9529 (было 0.9073).

Рисунок 20 показывает, что данные улучшились не так сильно, как при валидации модели регрессии, но при этом изменения наблюдается только в положительную сторону и приближены к идеальным, что означает, что модель будет достаточно точно предсказывать некоторые данные.

```
Модель случайного леса с параметрами MaxDepth = 10, MaxBins = 20, NumTrees = 50
+---------+------------------+
|   metric|             value|
+---------+------------------+
|       TP|          367206.0|
|       FP|           18547.0|
|       TN|          512782.0|
|       FN|           17674.0|
|Precision|0.9519200109914894|
|   Recall|0.9540791935148618|
|       F1|0.9529983792544571|
+---------+------------------+

AUR(Площадь под рабочей характеристикой приемника)= 0.994462
```

Рисунок 20 –Результаты кросс–валидации рандомного леса

2.5 Выводы

В данном разделе подготовленные для машинного обучение данные разделялись на обучающие и тестовые. Был осуществлен процесс обучения моделей регрессии и классификации, а также проведена кросс-валидация для нахождения наилучших показателей у моделей для разных гиперпараметров.

В процессе работы было произведено знакомство с инструментом для работы с машинным обучением в Apache Spark, а также изучены методы линейной регрессии и классификации при помощи случайного леса. При проведении кросс-валидации были выявлены лучшие параметры для моделей регрессии и классификации, а также обучены на основе данных параметров.

ЗАКЛЮЧЕНИЕ

Таким образом в данной работе было проведено исследование Daily dataset с использованием технологии больших данных. В данной работе был проведен разведочный анализ данных на датасете с помощью PySpark.

Разведочный анализ включал в себя определение типов признаков в датасете, определение пропущенных значений и их устранение, определение выбросов и их устранение, расчет статистических показателей признаков, вывод корреляции между признаками, визуализации распределения признаков.

Также было проведен анализ обработанных данных датасета с помощью двух алгоритмов машинного обучения – задачи регрессии (LinearRegression), и задачи бинарной классификации (RandomForest). Эффективность полученных моделей была рассмотрена с помощью расчета метрик классификации, а именно матрицы ошибок и площади под кривой ROC (AUR), и метрик регрессии, а именно среднеквадратическая ошибка (RMSE) и коэффициент детерминации (R2).

Также для улучшения эффективности моделей, был выполнен подбор гиперпараметров модели по сетке. Улучшение эффективности было доказано с помощью повторного расчета метрик, описанных выше, и сравнение их метриками, рассчитанными для изначальной модели.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Smart meters in London [Электронный ресурс] Режим доступа : https://www.kaggle.com/datasets/jeanmidev/smart-meters-in-london?select=daily_dataset (дата обращения : 23.12.2022);

2 Выброс (Outlier) [Электронный ресурс] Режим доступа : https://www.helenkapatsa.ru/vybros/ (дата обращения : 25.12.2022)

3 CrossValidator [Электронный ресурс] Режим доступа : https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.tuning.CrossValidator.html (дата обращения : 03.01.2023);

4 Перекрестная проверка: оценка производительности [Электронный ресурс] Режим доступа : https://scikit-learn.ru/3-1-cross-validation-evaluating-estimator-performance/ (дата обращения : 04.01.2023);

5 20 популярных метрик машинного обучения. Часть 1. Метрики классификации и регрессионной оценки [Электронный ресурс] Режим доступа : https://machinelearningmastery.ru/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce (дата обращения : 04.01.2023).