Пермский филиал федерального государственного автономного
образовательного учреждения высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет экономики, менеджмента и бизнес-информатики

Чепоков Елизар, Ануфриев Павел, Дроздов Андрей, Кроливецкая Арина

ОПРЕДЕЛЕНИЕ ВЛИЯНИЯ СПОСОБА И РЕЖИМА ЭКСПЛУАТАЦИИ НА ВИДЫ ДАВЛЕНИЯ В СКВАЖИНАХ

Проектная работа

студента образовательной программы «Программная инженерия»

по направлению подготовки *09.03.04 Программная инженерия*

Руководитель:

_____

А.В. Кычкин

Пермь, 2021 год

Оглавление

Введение

Данные о проекте: Технологический процесс добычи нефти. В работе находятся несколько скважин, каждая из которых оснащена насосным оборудованием. Оборудование может работать в различных режимах. Основные технологические параметры о работе оборудования измеряются с дискретностью 1 сутки (раз в 24 часа). В связи с нестабильной связью, особенностями сбора данных (ручной ввод) имеются значительные пропуски в данных.

Цель работы: определить влияние способа и режима эксплуатации на виды давления в скважинах.

Исходные данные: файл Excel с параметрами скважин, результатами измерений, режимами работы оборудования. Каждая вкладка файла содержит данные по 1 месяцу работы всего массива скважин.

Подготовительный этап:

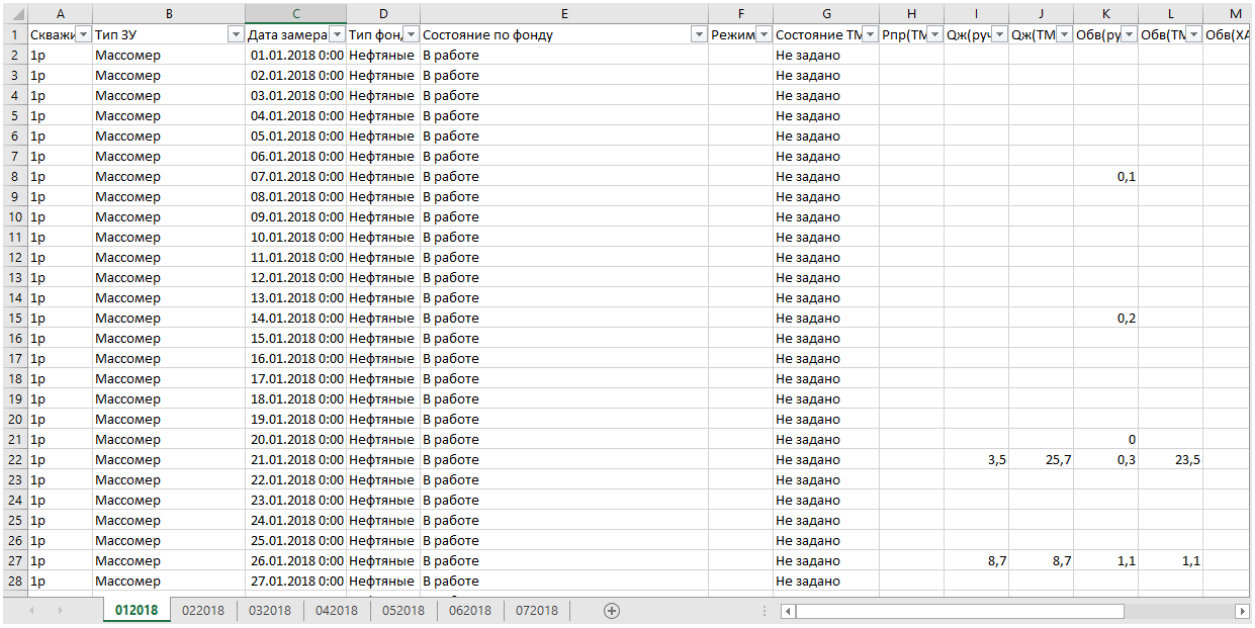
1. Обработка данных для исследований.py - реализует парсинг данных из исходного файла и формирование датафрейма. Формируется новый датафрейм с данными только по одной выбранной скважине. Также в файле реализуется начальная очистка данных (удаление столбцов с пустыми значениями во всех строках) для выбранной скважины.
2. Сохранение_данных_в_CSV_для_Influx.py - реализует парсинг данных из исходного файла, подготовку датафрейма для экспорта и сам экспорт в csv файл. Далее можно использоваться скрипт CSV2Influx.py для экспорта в БД InfluxDB.

Постановка задачи

1. **Осуществить выборку данных:**
   1.1. Исследовать файл «Данные для исследований.xlsx» с информацией о скважинах за 7 месяцев;
   1.2. Выбрать данные в соответствии с номером проекта (100 скважин на проект);
   1.3. Выполнить фильтрацию данных в соответствии с задачами проекта (выбрать столбцы: Скважина, Дата замера, Способ эксплуатации, Режим, и все столбцы с давлениями);
2. **Выполнить очистку данных:**
   2.1. Заполнить пропуски;
   2.2. Удалить аномалии;
3. **Провести трансформацию данных:**
   3.1. Сгруппировать данные по скважинам;
   3.2. Выполнить сглаживание в окне скользящего среднего;
   3.3. Значения давлений привести при помощи нормализации (деления на мах. значение) к диапазону: от 0 до 1;
4. **Анализ данных:**
   4.1. Минимальные, максимальные, средние значения давлений по скважинам, сгруппировать по месяцам и за 7 месяцев;
   4.2. Построить линейную регрессию способа и режима добычи на виды давления по скважинам;
5. **Визуализировать полученные данные.**

Выборка данных

Выборка данных производилась из файла “Данные для исследований.xlsx” в котором хранится информация о скважинах за 7 месяцев. В соответствии с 10 номером нашей группы мы рассматривали скважины, с порядковыми номерами с 1 по 100. На рисунке 1.1 представлен исходный файл до фильтрации данных.

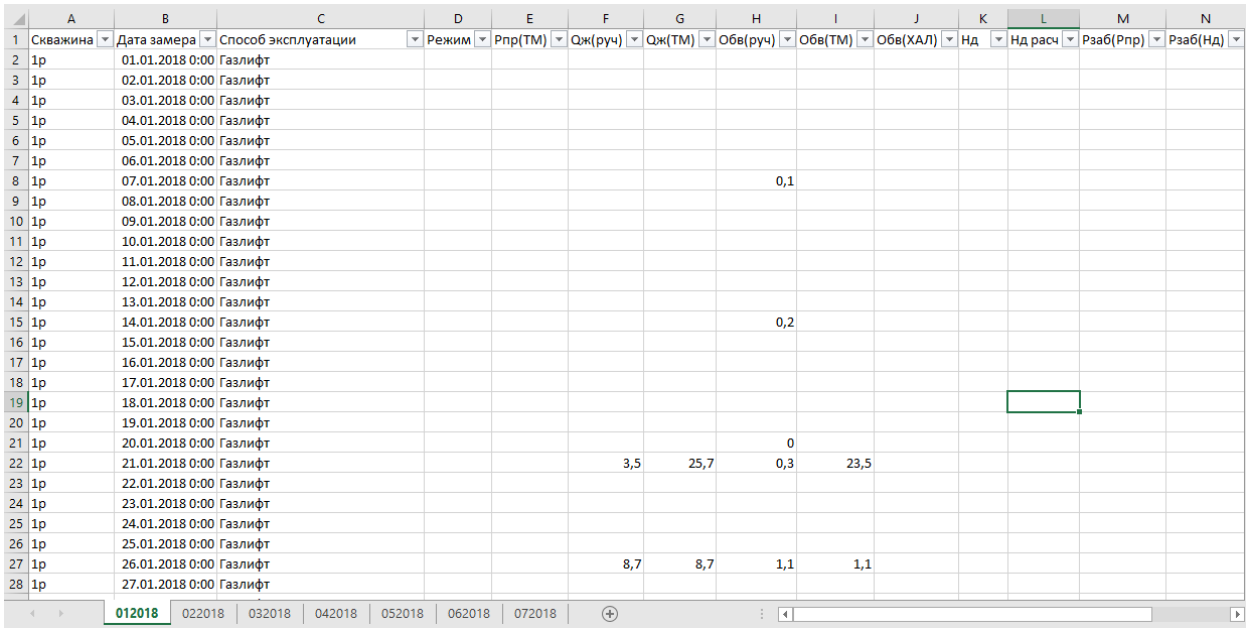


	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Скважи	Тип ЗУ	Дата замера	Тип фон	Состояние по фонду	Режим	Состояние ТМ	Рпр(ТМ	Qж(руч	Qж(ТМ	Обв(ру	Обв(ТМ	Обв(ХА
2	1р	Массомер	01.01.2018 0:00	Нефтяные	В работе		Не задано						
3	1р	Массомер	02.01.2018 0:00	Нефтяные	В работе		Не задано						
4	1р	Массомер	03.01.2018 0:00	Нефтяные	В работе		Не задано						
5	1р	Массомер	04.01.2018 0:00	Нефтяные	В работе		Не задано						
6	1р	Массомер	05.01.2018 0:00	Нефтяные	В работе		Не задано						
7	1р	Массомер	06.01.2018 0:00	Нефтяные	В работе		Не задано						
8	1р	Массомер	07.01.2018 0:00	Нефтяные	В работе		Не задано				0,1		
9	1р	Массомер	08.01.2018 0:00	Нефтяные	В работе		Не задано						
10	1р	Массомер	09.01.2018 0:00	Нефтяные	В работе		Не задано						
11	1р	Массомер	10.01.2018 0:00	Нефтяные	В работе		Не задано						
12	1р	Массомер	11.01.2018 0:00	Нефтяные	В работе		Не задано						
13	1р	Массомер	12.01.2018 0:00	Нефтяные	В работе		Не задано						
14	1р	Массомер	13.01.2018 0:00	Нефтяные	В работе		Не задано						
15	1р	Массомер	14.01.2018 0:00	Нефтяные	В работе		Не задано				0,2		
16	1р	Массомер	15.01.2018 0:00	Нефтяные	В работе		Не задано						
17	1р	Массомер	16.01.2018 0:00	Нефтяные	В работе		Не задано						
18	1р	Массомер	17.01.2018 0:00	Нефтяные	В работе		Не задано						
19	1р	Массомер	18.01.2018 0:00	Нефтяные	В работе		Не задано						
20	1р	Массомер	19.01.2018 0:00	Нефтяные	В работе		Не задано						
21	1р	Массомер	20.01.2018 0:00	Нефтяные	В работе		Не задано				0		
22	1р	Массомер	21.01.2018 0:00	Нефтяные	В работе		Не задано		3,5	25,7	0,3	23,5	
23	1р	Массомер	22.01.2018 0:00	Нефтяные	В работе		Не задано						
24	1р	Массомер	23.01.2018 0:00	Нефтяные	В работе		Не задано						
25	1р	Массомер	24.01.2018 0:00	Нефтяные	В работе		Не задано						
26	1р	Массомер	25.01.2018 0:00	Нефтяные	В работе		Не задано						
27	1р	Массомер	26.01.2018 0:00	Нефтяные	В работе		Не задано		8,7	8,7	1,1	1,1	
28	1р	Массомер	27.01.2018 0:00	Нефтяные	В работе		Не задано						

012018 | 022018 | 032018 | 042018 | 052018 | 062018 | 072018

Рисунок 1.1 – файл «Данные для исследований.xlsx»

Мы провели фильтрацию данных в соответствии с задачами проекта и выбрали столбцы в соответствии с поставленной задачей: Скважина, Дата замера, Способ эксплуатации, Режим, и все столбцы с давлениями. Результат представлен на рисунке 1.2.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Скважина	Дата замера	Способ эксплуатации	Режим	Рпр(ТМ)	Qж(руч)	Qж(ТМ)	Обв(руч)	Обв(ТМ)	Обв(ХАЛ)	Нд	Нд расч	Рзаб(Рпр)	Рзаб(Нд)
2	1р	01.01.2018 0:00	Газлифт											
3	1р	02.01.2018 0:00	Газлифт											
4	1р	03.01.2018 0:00	Газлифт											
5	1р	04.01.2018 0:00	Газлифт											
6	1р	05.01.2018 0:00	Газлифт											
7	1р	06.01.2018 0:00	Газлифт											
8	1р	07.01.2018 0:00	Газлифт					0,1						
9	1р	08.01.2018 0:00	Газлифт											
10	1р	09.01.2018 0:00	Газлифт											
11	1р	10.01.2018 0:00	Газлифт											
12	1р	11.01.2018 0:00	Газлифт											
13	1р	12.01.2018 0:00	Газлифт											
14	1р	13.01.2018 0:00	Газлифт											
15	1р	14.01.2018 0:00	Газлифт					0,2						
16	1р	15.01.2018 0:00	Газлифт											
17	1р	16.01.2018 0:00	Газлифт											
18	1р	17.01.2018 0:00	Газлифт											
19	1р	18.01.2018 0:00	Газлифт											
20	1р	19.01.2018 0:00	Газлифт											
21	1р	20.01.2018 0:00	Газлифт					0						
22	1р	21.01.2018 0:00	Газлифт			3,5	25,7	0,3	23,5					
23	1р	22.01.2018 0:00	Газлифт											
24	1р	23.01.2018 0:00	Газлифт											
25	1р	24.01.2018 0:00	Газлифт											
26	1р	25.01.2018 0:00	Газлифт											
27	1р	26.01.2018 0:00	Газлифт			8,7	8,7	1,1	1,1					
28	1р	27.01.2018 0:00	Газлифт											

012018 | 022018 | 032018 | 042018 | 052018 | 062018 | 072018

Рисунок 1.2 – файл «Data.xlsx»

Данные в файле сгруппированы по скважинам и располагаются в хронологическом порядке выполнения замеров. Для выборки данных из исходного файла был отредактирован представленный в качестве примера преподавателем код на python (см. листинг программы в Приложении А). Так как программное обеспечение InfluxDB не позволяет загрузку отсутствующих значений, файл “Data.xlsx” был дополнен временными данными.

Среди предложенных вариантов заполнения были выявлены следующие:

1. Заполнение последним доступным значением;
2. Заполнение следующим доступным значением;
3. Заполнение нулями;
4. Интерполяция (линейная, полином, сплайн).

Из данных вариантов было выбрано заполнение следующими или последними доступными значениями, так как только данные варианты заполнения подходят в нашем случае. Вариант заполнения интерполяцией был отклонен из-за неэффективности для большого массива данных с большим количеством пропущенных значений. Заполнение нулями невозможно так как из-за большого количества пропущенных значений данные пустые значения серьезно повлияют на все возможные статистические метрики.

Результат заполнения представлен на рисунке 1.3.

	A	B	C	D	E	F	G	H	I
1	Дата замера	Скважина	Время	Способ эк	Режим	Pпр(ТМ)	Pзаб(Pпр)	Pзаб(Нд)	Pзаб(иссл)
2	01.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
3	02.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
4	03.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
5	04.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
6	05.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
7	06.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
8	07.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
9	08.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
10	09.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
11	10.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
12	11.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
13	12.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
14	13.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
15	14.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
16	15.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
17	16.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
18	17.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
19	18.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	68.0
20	19.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	61.5
21	20.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	61.5
22	21.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	61.5
23	22.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	61.5
24	23.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	61.5
25	24.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	61.5
26	25.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	61.5
27	26.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	61.5
28	27.01.2018	105-1Д	0:00:00	1	1	0.0	168.1	118.0	61.5

holes

Рисунок 1.3 – файл «holes.csv»

Заполнение осуществлялось функцией python, fillna, где в зависимости от расположения пропущенного значения выбирался вариант заполнения “bfill” или

“ffill”, после чего данные записывались в новый документ, который будет загружаться в базу данных influx.

Для полученных “сырых” значений был проведен анализ с использованием веб-приложения Grafana, для чего было дополнительно отфильтрованы и загружены в базу данных значения из файла “holes.csv”. Так же были преобразованы данные с помощью языка python (см. Приложение Б). Данные были загружены с типами столбцов представленными на рисунке 1.4.

```
> SHOW FIELD KEYS
name: raw_data
fieldKey            fieldType
--------            ---------
Время               string
Режим               integer
Рзаб(Нд)            float
Рзаб(Рпр)           float
Рзаб(иссл)          float
Рпр(ТМ)             float
Способ эксплуатации integer
```

Рисунок 1.4 – файл «holes.csv»

Результат действий представлен в виде сырых данных на рисунке 1.5 и рисунке 1.6.

Рисунок 1.5 – 1 вариант результата

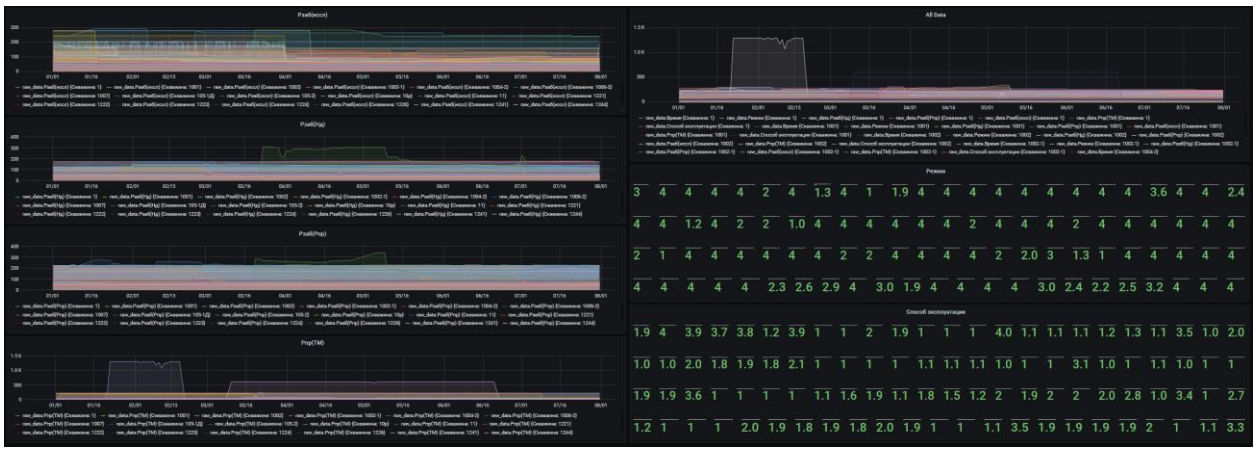


Рисунок 1.6 – 2 вариант результата

Очистка данных

Во время выборки и визуализации сырых данных было выявлено, что часть значений имеет большое отклонение от нормы, в связи с чем был сделан вывод о том, что анализируемые данные содержат аномальные значения.

Существуют различные виды аномалий:

1. Дубли - выбор одного значения
2. Противоречия - числовой результат противоречит реальному состоянию объекта
   2.1. Типы и виды физических процессов, выбор диапазонов измерений
   2.2. Резкое изменение параметров, которое невозможно объяснить

Аномалии можно выявить различными способами:

1. Пороговый фильтр - фильтрация всплесков или провалов
2. Фильтр по доверительному интервалу или стандартному отклонению
3. Фильтр по квантилю, как меры оценки распределения величин

Для определения аномалий нами был выбран метод фильтрации по квантилям уровня 0,01 и 0,99. Квантили рассчитывались для каждого из столбцов массива данных. Для вычисления использовалась библиотека pandas, код представлен ниже.

```
# -*- coding: utf-8 -*-

# !pip install influxdb
import pandas as pd
import numpy as np

df = pd.read_csv('holes.csv', sep=';', engine='python')
q_low = df.quantile(.01)
q_high = df.quantile(.99)
print(q_low)
print(q_high)

df = df[(df['Режим'] >= q_low['Режим']) & (df['Режим'] <= q_high['Режим'])]
df = df[(df['Рпр(ТМ)'] >= q_low['Рпр(ТМ)']) & (df['Рпр(ТМ)'] <= q_high['Рпр(ТМ)'])]
df = df[(df['Рзаб(Рпр)'] >= q_low['Рзаб(Рпр)']) & (df['Рзаб(Рпр)'] <= q_high['Рзаб(Рпр)'])]
df = df[(df['Рзаб(иссл)'] >= q_low['Рзаб(иссл)']) & (df['Рзаб(иссл)'] <= q_high['Рзаб(иссл)'])]

df.to_csv('clear_holes.csv', index=False, sep=';')
```

На рисунке 2.1 представлены значения квантилей.

Рисунок 2.1 – Квантили массива данных

На рисунке 2.2 представлено количество строк до и после срабатывания скрипта для удаления аномалий и соответственно на рисунке 2.3 представлен результат удаления аномалий.

Рисунок 2.2 – количество строк данных

Рисунок 2.3 – Результат удаления аномалий

Трансформация

Для текущей задачи стоит выделить 3 вида группировки:

1. Группировка по времени - разделение временного ряда по неделям / месяцам
2. Группировка по скважинам - разделение всех метрик по скважинам с различными ID
3. Группировка по способам добычи, по режимам работы и другим метрикам

Так как стоит отделить разные скважины стоит произвести группировку по скважинам.

Фильтрация нужна для того, чтобы:

1. устранить высокочастотные составляющие, всплески и провалы
2. устранить низкочастотные составляющих, сезонность и тренд

Существуют различные виды фильтров:

1. фильтр сглаживающий, простого скользящего среднего с окном 5 значений
2. фильтр экспоненциального сглаживающего среднего, веса значений уменьшаются по мере удаления от центральной точки, взять окно равное 5
3. фильтр медианный
4. фильтр Калмана и другие фильтры с адаптацией к сигналу

Для данных был применен сглаживающий фильтр скользящего среднего. Данный алгоритм формирует набор данных, который служит для построения модели прогнозирования. Код алгоритма представлен ниже.

```
# -*- coding: utf-8 -*-

import numpy as np
import pandas as pd

data = pd.read_csv('clean_holes.csv', sep=';', engine='python')
df = pd.DataFrame(data)

print(f"DataFrame:\n{df}\n")
```

```
print(f"column types:\n{df.dtypes}")

holes_list = []

col_List= df['Скважина'].tolist()
num = 1
for i in range(len(col_List)):
   if (i == 0) or (col_List[i] != col_List[i-1]):
      holes_list.append(col_List[i])
      num += 1

columns = ['Рпр(ТМ)', 'Рзаб(Рпр)', 'Рзаб(Нд)', 'Рзаб(иссл)']

for hole in holes_list:
   df1 =  df[lambda df: df['Скважина'] == hole]

   for col in columns:
      col_List= df1[col].tolist()

      for i in range(len(col_List)):
         delta = len(col_List) - i
         left = col_List[i-1]+col_List[i-2]+col_List[i-3]+col_List[i-4]+col_List[i-5]
         if delta == 1:
            right = col_List[0]+col_List[1]+col_List[2]+col_List[3]+col_List[4]
         if delta == 2:
            right = col_List[i+1]+col_List[0]+col_List[1]+col_List[2]+col_List[3]
         if delta == 3:
            right = col_List[i+1]+col_List[i+2]+col_List[0]+col_List[1]+col_List[2]
         if delta == 4:
            right = col_List[i+1]+col_List[i+2]+col_List[i+3]+col_List[0]+col_List[1]
         if delta == 5:
            right = col_List[i+1]+col_List[i+2]+col_List[i+3]+col_List[i+4]+col_List[0]
         if delta >= 6:
            right = col_List[i+1]+col_List[i+2]+col_List[i+3]+col_List[i+4]+col_List[i+5]
         col_List[i] = (left + right) / 10

      df1[col] = col_List
      df[lambda df: df['Скважина'] == hole] = df1

max = df.max()
df['Рпр(ТМ)'] = df['Рпр(ТМ)'].apply(lambda x: round(x / max['Рпр(ТМ)'], 3))
df['Рзаб(Рпр)'] = df['Рзаб(Рпр)'].apply(lambda x: round(x / max['Рзаб(Рпр)'], 3))
df['Рзаб(Нд)'] = df['Рзаб(Нд)'].apply(lambda x: round(x / max['Рзаб(Нд)'], 3))
df['Рзаб(иссл)'] = df['Рзаб(иссл)'].apply(lambda x: round(x / max['Рзаб(иссл)'], 3))

df.to_csv('normal_holes.csv', index=False, sep=';', encoding='cp1251')
```

После фильтрации данные были нормализованы делением каждого значения на максимальное к диапазону от 0 до 1. (рис. 3.1)

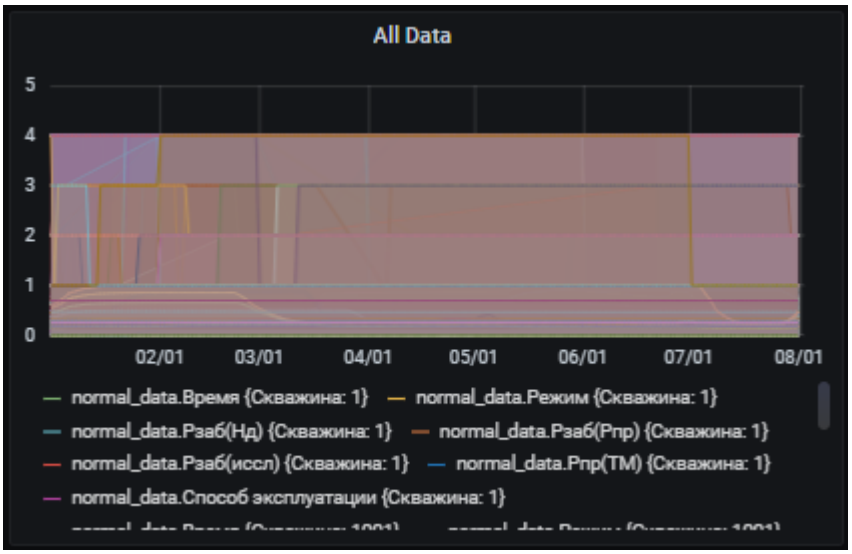


Рисунок 3.1 – Нормализованные данные

На рисунках 3.2 и 3.3 показаны результаты сглаживания по некоторым скважинам.

Рисунок 3.2 – Результат

Рисунок 3.3 – Результат

По представленным данным видны резкие перепады во время замера датчиков без сглаживания (слева) и нормализованные данные со сглаживанием (справа).

Сглаживание дополняет замеры и помогает получить примерные данные при отсутствии реальных замеров, а так же помогает в визуализации значений на более коротком промежутке, без сглаживания на панели отображалось отсутствие данных, которое неподготовленный человек воспринял бы за ошибку.

Анализ

На данном этапе представлен анализ данных с наглядной демонстрацией. В ходе выполнения заданий было создано 3 скрипта для заполнения, очистки и сглаживания значений, а также был модифицирован скрипт для загрузки данных в базу данных Influx. Для выполнения задания потребовалось создать одну базу данных и внести 3 измерения «raw_data» - список неотфильтрованных значений для визуализыции сырых данных «clean_data» - список с удалением всех аномалий повторяющихся данных «normal_data» - нормализованный список с фильтрацией данных.

На рисунке 4.1 представлен формат размещения всех 3 типов данных на панелях Grafana.

Рисунок 4.1 – Данные

На рисунках 4.2 представлены панели графиков с датчиками.

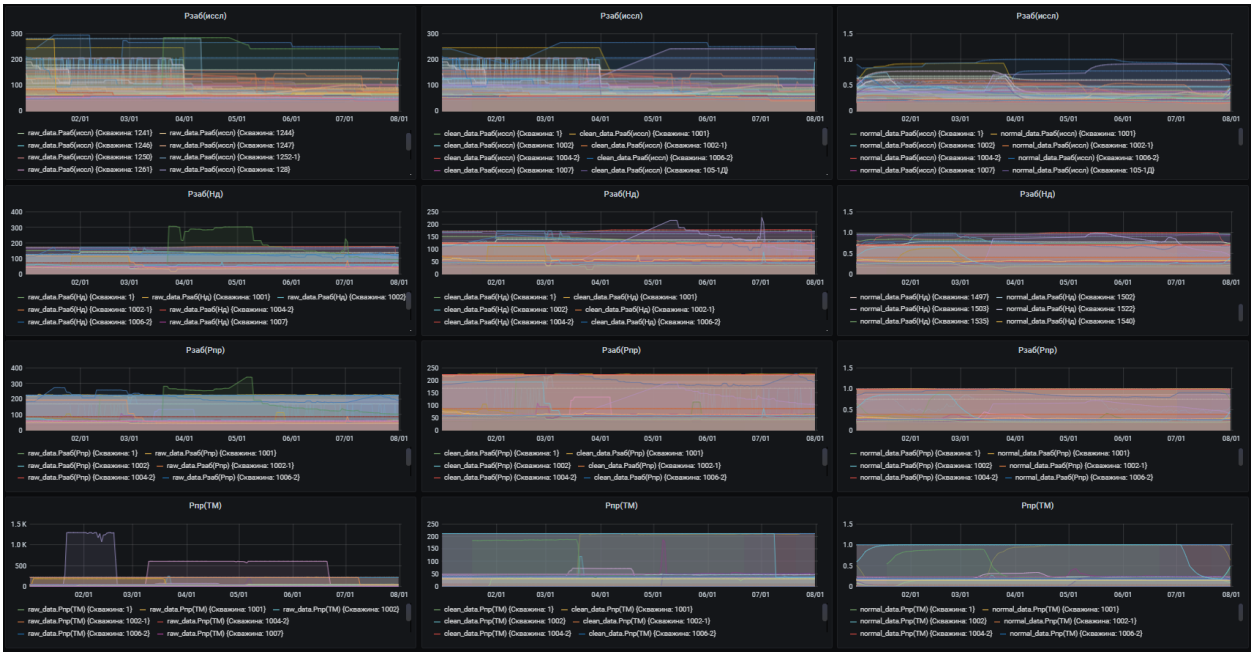


Рисунок 4.2 – Данные

На рисунках 4.3 представлены значения режимов и способов эксплуатации

Рисунок 4.3 – Данные

На рисунке 4.4 представлены максимальные значения датчиков

Рисунок 4.4 – Данные

На рисунке 4.5 представлены средние значения датчиков

Рисунок 4.5 – Данные

На рисунке 4.6 представлены минимальные значения датчиков

Рисунок 4.6 – Данные

На рисунках 4.7, 4.8, 4.9, 4.10 представлены графики изменений среднего, максимального и минимального значений, сгруппированных по неделям.

Рисунок 4.7 – Данные

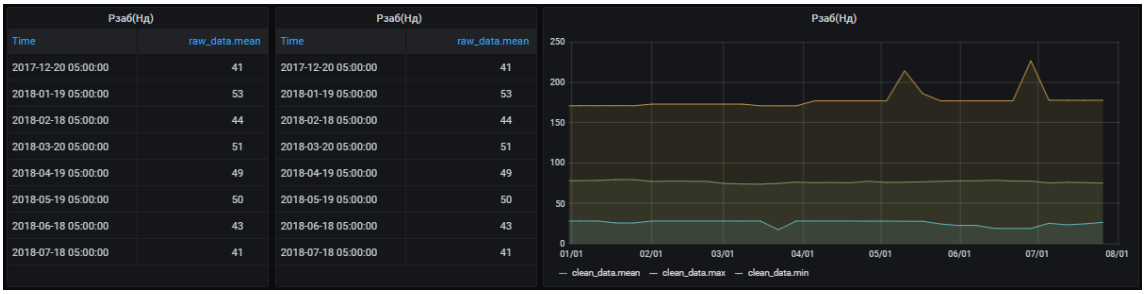


Рисунок 4.8 – Данные

Рисунок 4.9 – Данные

Рисунок 4.10 – Данные

На рисунках 4.11 и 4.12 изображены диаграммы датчиков Рзаб(нд) и Рзаб(иссл). При зависимости способа эксплуатации и режима (в данном случае способа эксплуатации 1 и режима 4) зависит давление в датчике Рзаб(нд). Таким образом совокупность значений способа эксплуатации и режима прямо пропорционально значению давления датчика. Основываясь на графиках можно заметить, что давления Рпр(Тм), Рзаб(Рпр) и Рзаб(Нд) обратно пропорционально способу эксплуатации, т.е. с увеличением номера способа эксплуатации значения давлений уменьшаются.

Рисунок 4.11 – Данные датчика Рзаб(Нд)

Рисунок 4.12 – Данные датчика Рзаб(иссл)

На рисунках 4.13-4.16 изображены средние значения давлений датчиков, сгруппированных по неделям на протяжении всего периода.

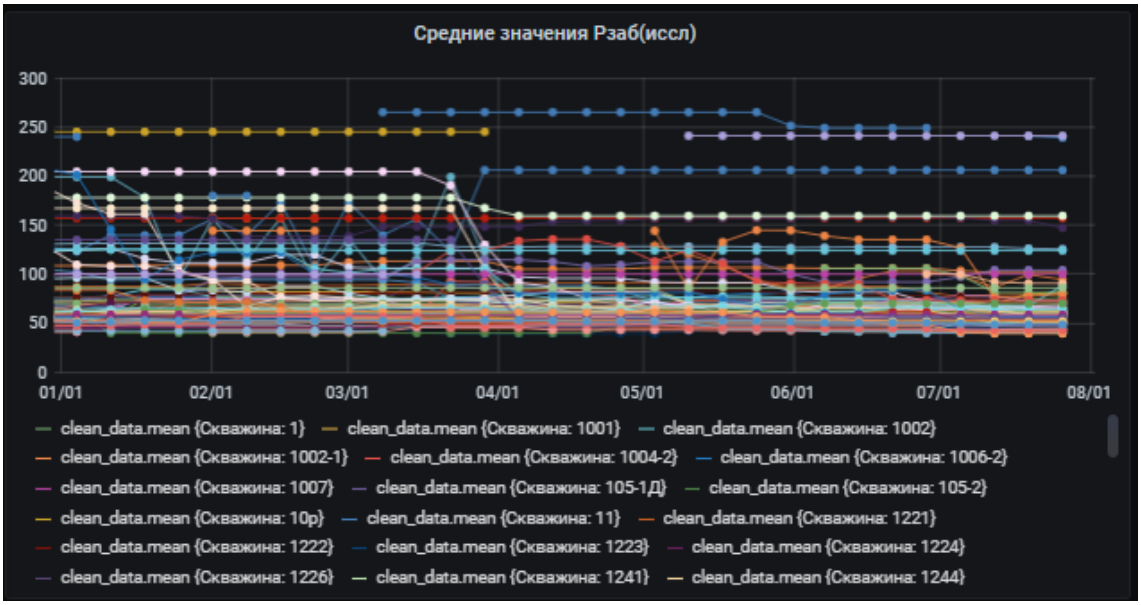


Рисунок 4.13 – Данные датчика Рзаб(иссл)

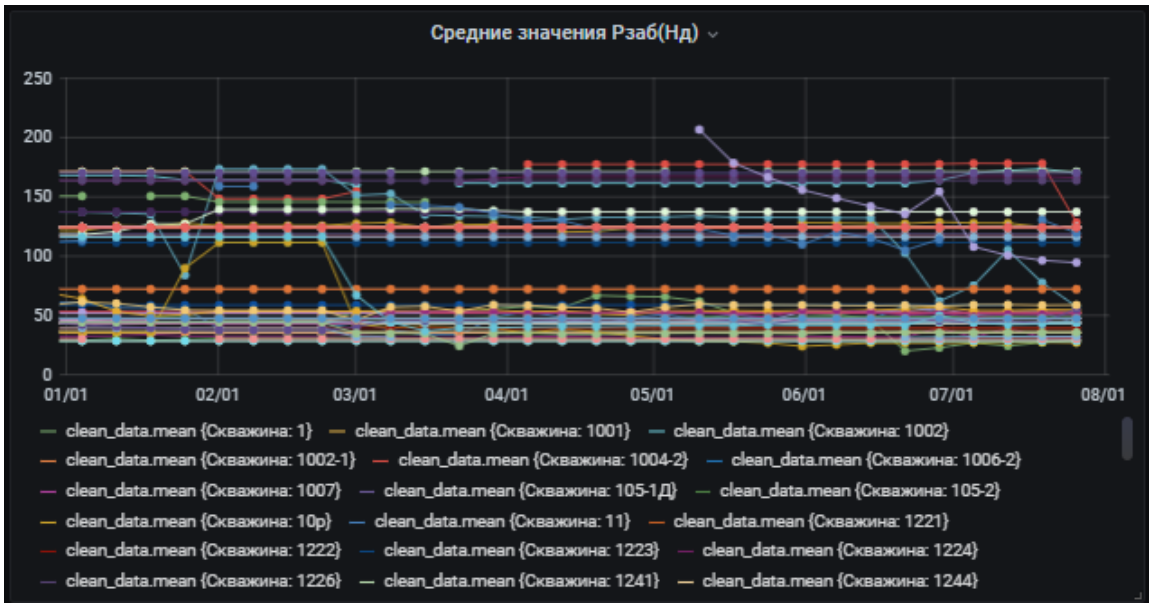


Рисунок 4.14 – Данные датчика Рзаб(Нд)

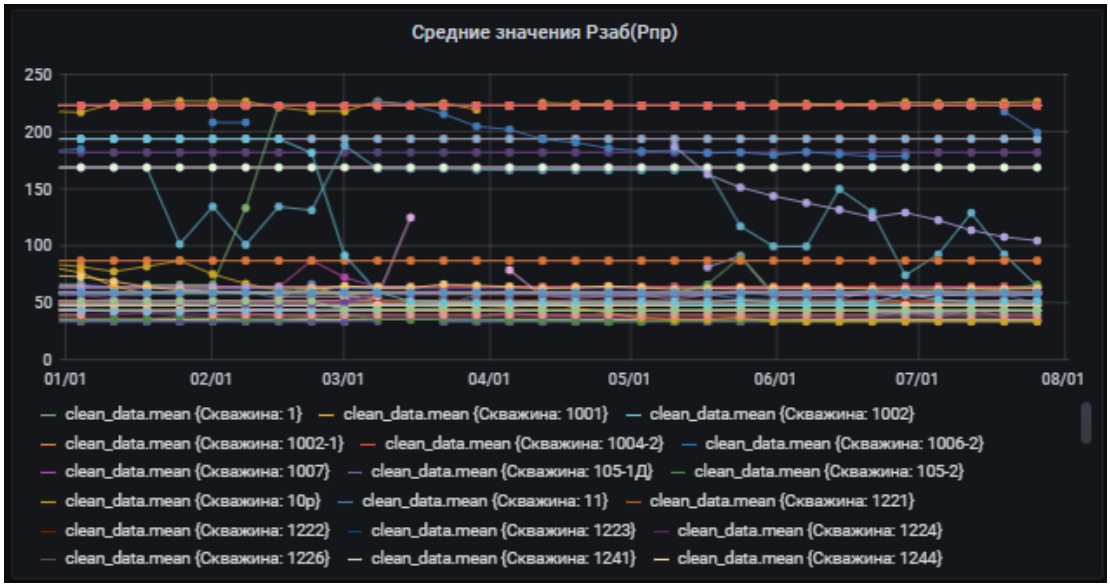


Рисунок 4.15 – Данные датчика Рзаб(Рпр)

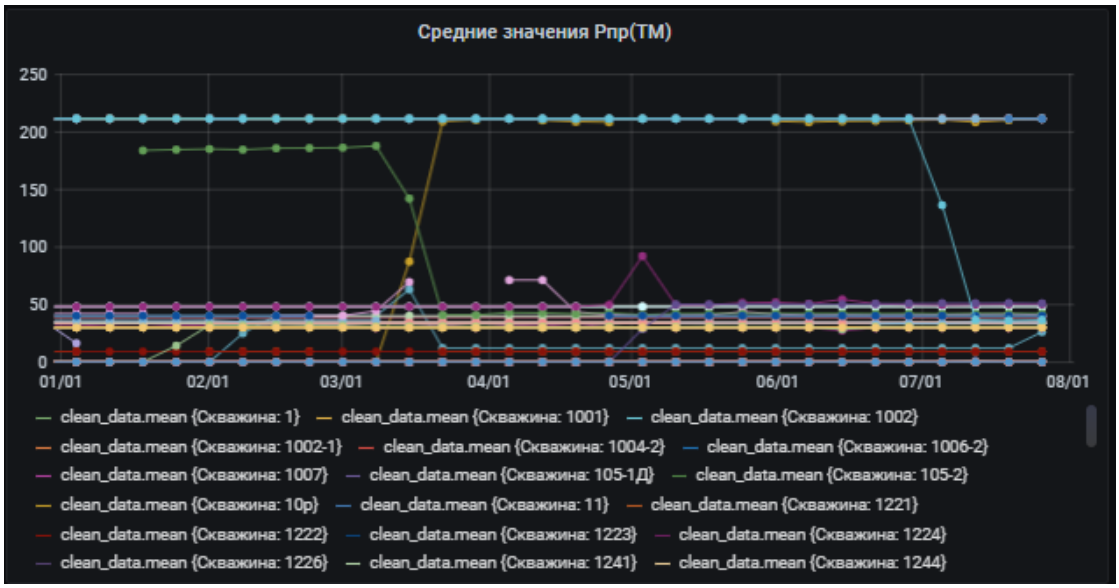


Рисунок 4.16 – Данные датчика Рпр(ТМ)

На рисунках 4.17-4.18 изображены все данные по скважине 704. В ходе анализа было выявлено, что от изменения режима с 1 на 3, изменились в положительную сторону показатели датчиков рзаб(рпр), рпр(тм), рзаб(нд), однако показатели датчика рзаб(иссл) ухудшились.

Рисунок 4.16 – Данные скважины 704

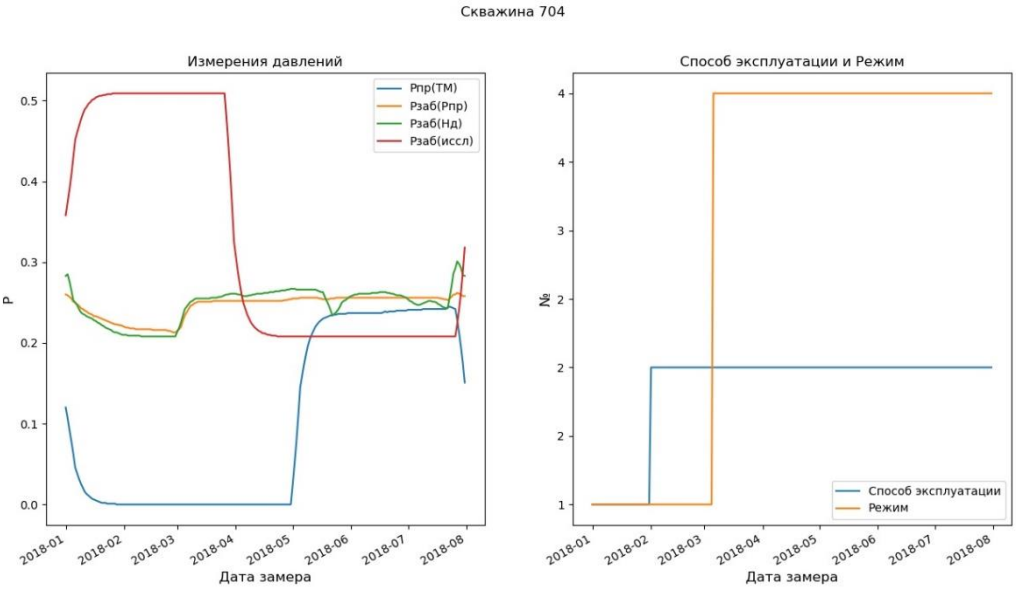


Рисунок 4.17 – Данные скважины 704

Заключение

В процессе работы из 500 предложенных скважин были выбраны первые 100. Для выбранных данных были произведены: заполнение следующими и последними доступными значениями пропусков и удаление аномальных значений, трансформация методом скользящего окна и нормализация путем деления всех значений на максимальное значение. Были построены графики, отражающие средние значения датчиков, режимов, соотношение максимальных минимальных значений датчиков сгруппированных по 7 дней.

Если посмотреть на графики зависимости от режима давления, то график давления Рпр(Тм) обратно пропорциональны, т.е с увеличением режима значение давления падает. Что касается остальных давлений, их графики имеют параболистический вид, из-за чего сложно характеризовать зависимость данными параметрами. Давления на 1 и 4 режиме находятся примерно на одном уровне, в то время как к 3 режиму значение давлений падает.

Приложение. Листинг скрипта

```
# -*- coding: utf-8 -*-

"""
Created on 0 0:00:00 0000

@author: Bloodies
"""

# !pip install influxdb

from influxdb import InfluxDBClient
from pytz import timezone
from openpyxl import Workbook
#from sklearn.linear_model import LinearRegression
import matplotlib.ticker as ticker
#import statsmodels.api as sm
import pandas as pd
import numpy as np
#import seaborn as sns
import matplotlib.pyplot as plt
import csv
import datetime
import os
import shutil

first_hole = 1
last_hole = 100

date = 'Дата замера'
hole = 'Скважина'
x1 = 'Способ эксплуатации'
x2 = 'Режим'
y1 = 'Рпр(ТМ)'
y2 = 'Рзаб(Рпр)'
y3 = 'Рзаб(Нд)'
y4 = 'Рзаб(иссл)'

_host = 'localhost'
_port = 8086

def read_all_sheets(file_name_excel):
    df = pd.DataFrame()
    xls = pd.ExcelFile(file_name_excel)
    for list_excel in xls.sheet_names:
        df = df.append(pd.read_excel(xls, list_excel, parse_dates=[date],
index_col=date))
    return df

def replace_text_values_in_x(df, nameX):
    dict_changes = {}
    _list = pd.unique(df[nameX]).tolist()
    i = 1
    for value in _list:
        if (str(value) != str(np.NaN)):
            df.loc[df[nameX] == value, nameX] = i
            dict_changes[i] = value
            i += 1
    df[nameX] = df[nameX].fillna(len(_list))
    dict_changes[len(_list)] = np.NaN
    return dict_changes
```

```
def convert_csv_to_xlsx(current_file):
    wb = Workbook()
    sheet = wb.active

    CSV_SEPARATOR = ";"

    with open(current_file + '.csv') as f:
        reader = csv.reader(f)
        for r, row in enumerate(reader):
            for c, col in enumerate(row):
                for idx, val in enumerate(col.split(CSV_SEPARATOR)):
                    cell = sheet.cell(row=r+1, column=idx+1)
                    cell.value = val

    wb.save(current_file + '.xlsx')

def separate_export(current_file, current_path):
    file_name = current_file

    df = read_all_sheets(file_name)
    df.sort_index(inplace=True)

    ''' Если появляется ошибка в данных способа эксплуатации и режиме
заменить'''
    #what_replaced_x1 = replace_text_values_in_x(df, x1)
    #what_replaced_x2 = replace_text_values_in_x(df, x2)
    replace_text_values_in_x(df, x1)
    replace_text_values_in_x(df, x2)

    all_data = df.copy()

    if os.path.exists('holes/' + current_path):
        shutil.rmtree('holes/' + current_path)
        os.mkdir('holes/' + current_path)
    else:
        os.mkdir('holes/' + current_path)

    cleaning_map = lambda x: str(x).strip()
    all_data[hole] = all_data[hole].map(cleaning_map)
    all_df_to_influx = all_data.copy()[[hole, x1, x2, y1, y2, y3, y4]]

    influx_file_name = '_data_2018_01-07'
    list_of_holes = pd.unique(all_data[hole]).tolist()
    count_empty_data = 0
    list_empty_data = []
    cleaned_data = pd.DataFrame()
    for _hole in list_of_holes:
        df_to_influx = all_df_to_influx[all_df_to_influx[hole] == _hole][[x1,
x2, y1, y2, y3, y4]]
        df_to_influx.insert(loc=0, column='Время',
value=df_to_influx.index.time[0])
        temp_df = df_to_influx[[y1, y2, y3, y4]].dropna(axis=0, how='all')
        if (not temp_df.empty):
            cleaned_data = cleaned_data.append(df_to_influx)
            df to influx.to csv('holes/' + current path + '/' + str( hole) +
influx_file_name+'.csv', encoding='cp1251', sep=';')
        else:
            list_empty_data.append(_hole)
            count_empty_data += 1
    print('Количество скважен без данных по давлению: '+
str(count_empty_data))
    print('Список скважен без данных по давлению:')
    i=1
```

```
    for _hole in list_empty_data:
        print(str(i)+'. '+_hole)
        i+=1
    print('Сохранение в отдельные файлы ".csv" выполнено!')

def parcing_and_filling():
    file_name = 'data.xlsx'

    def read_all_sheets_f(file_name_excel):
        df = pd.DataFrame()
        xls = pd.ExcelFile(file_name_excel)
        for list_excel in xls.sheet_names:
            df = df.append(pd.read_excel(xls, list_excel, parse_dates=[date],
index_col=date))
        return df

    df = read_all_sheets_f(file_name)
    df.sort_index(inplace=True)

    def replace_text_values_in_x_f(df, nameX):
        dict_changes = {}
        _list = pd.unique(df[nameX]).tolist()
        i = 1
        for value in _list:
            if (str(value) != str(np.NaN)):
                df.loc[df[nameX] == value, nameX] = i
                dict_changes[i] = value
                i += 1
        df[nameX] = df[nameX].fillna(len(_list))
        dict_changes[len(_list)] = np.NaN
        return dict_changes

    ''' Если появляется ошибка в данных способа эксплуатации и режиме
заменить'''
    #what_replaced_x1 = replace_text_values_in_x_f(df, x1)
    #what_replaced_x2 = replace_text_values_in_x_f(df, x2)
    replace_text_values_in_x_f(df, x1)
    replace_text_values_in_x_f(df, x2)


    all_data = df.copy()

    cleaning_map = lambda x: str(x).strip()
    all_data[hole] = all_data[hole].map(cleaning_map)
    all_df_to_influx = all_data.copy()[[hole, x1, x2, y1, y2, y3, y4]]

    list_of_holes = pd.unique(all_data[hole]).tolist()
    count_empty_data = 0
    fill_method = 'bfill'
    list_empty_data = []
    cleaned_data = pd.DataFrame()
    for _hole in list_of_holes[first_hole:last_hole]:
        if count_empty_data > 0:
            fill_method = 'ffill'
        df_to_influx = all_df_to_influx[all_df_to_influx[hole] == _hole][[x1,
x2, y1, y2, y3, y4]]
        df_to_influx.insert(loc=0, column='Время',
value=df_to_influx.index.time[0])
        df_to_influx.insert(loc=0, column='Скважина', value=_hole)
        temp_df = df_to_influx[[y1, y2, y3, y4]].dropna(axis=1, how='all')
        if (not temp_df.empty):
            cleaned_data = cleaned_data.append(df_to_influx)
            cleaned_data[y1].fillna(method=fill_method, inplace=True)
            cleaned_data[y2].fillna(method=fill_method, inplace=True)
```

```
            cleaned_data[y2].fillna(method=fill_method, inplace=True)
            cleaned_data[y3].fillna(method=fill_method, inplace=True)
            cleaned_data[y4].fillna(method=fill_method, inplace=True)

            cleaned_data.to_csv('raw_holes.csv', encoding='cp1251', sep=';')
        else:
            list_empty_data.append(_hole)
            count_empty_data += 1

def INFLUX_INPUT(current_file, current_metric, cycle):
    index = 0
    _databasename = 'holes'
    timecolumn = 'Время'
    datecolumn = 'Дата замера'
    timeformat = '%Y-%m-%d %H:%M:%S'
    datatimezone = 'UTC'
    _delimiter=';'
    batchsize = 5000
    inputfilename = current_file #'holes.csv' #'clear_holes.csv'
#'normal_holes.csv'
    metric = [current_metric] #['raw_data'] #['clean_data'] #['normal_data']
    tagcolumns = ['Скважина']
    fieldcolumns = []

    client = InfluxDBClient(host=_host, port=_port)
    client.create_database(_databasename)
    db_list = client.get_list_database()
    print('существующие базы данных' + db_list)

    if (cycle == 0):
        client.drop_database(_databasename)
#client.drop_database(_databasename)
        client.create_database(_databasename)
#client.create_database(_databasename)

    client.switch_database(_databasename)

    epoch_naive = datetime.datetime.utcfromtimestamp(0)
    epoch = timezone('UTC').localize(epoch_naive)

    def unix_time_millis(dt):
        return int((dt - epoch).total_seconds() * 1000)
    """ Check if data type of field is float """
    def isfloat(value):
            try:
                float(value)
                return True
            except:
                return False

    """ Check if data type of field is int """
    def isinteger(value):
            try:
                if int(value):
                    return True
                else:
                    return False
            except:
                return False

    datapoints1 = []
    count = 0
    with open(inputfilename, 'r') as csvfile:
        reader = csv.DictReader(csvfile, delimiter=_delimiter)
```

```
        fieldcolumns = reader.fieldnames[2:]
        for row in reader:
            datetime_naive = datetime.datetime.strptime(row[datecolumn] + ' '
+ row[timecolumn], timeformat)
            datetime_local = timezone(datatimezone).localize(datetime_naive)

            timestamp = unix_time_millis(datetime_local) * 1000000 # in
nanoseconds
            tags = {}

            for t in tagcolumns:
                if t in row:
                    v = row[t]
                    pass
                tags[t] = v

            #fieldNames = {'Рзаб(Нд)', 'Рзаб(Рпр)'}
            fields = {}
            for f in fieldcolumns:
                v = 0
                if f in row:
                    if isinteger(row[f]):
                        v = int(row[f])
                    else:
                        v = float(row[f]) if isfloat(row[f]) else row[f]
                fields[f] = v

            datapoints1.append({"measurement": metric[0], "time": timestamp,
"fields": fields, "tags": tags})
            count += 1

            if len(datapoints1) % batchsize == 0:
                index += 1
                print('Read %d lines'%count)
                print('Inserting %d datapoints...'%(len(datapoints1)))
                response = client.write_points(datapoints1)

                if response == False:
                    print('Problem inserting points, exiting...')
                    exit(1)
                print("Wrote %d, response: %s" % (len(datapoints1),
response))
                datapoints1 = []
        print(index)

    if len(datapoints1) > 0:
        print('Read %d lines'%count)
        print('Inserting %d datapoints...'%(len(datapoints1)))
        response = client.write_points(datapoints1)

        if response == False:
            print('Problem inserting points, exiting...')
            exit(1)

        print("Wrote %d, response: %s" % (len(datapoints1), response))

    print('Done')

def cleaning():
    df = pd.read_csv('raw_holes.csv', sep=';', engine='python')
    q_low = df.quantile(.01)
    q_high = df.quantile(.99)
    print(q_low)
    print(q_high)
```

```
    df = df[(df['Режим'] >= q_low['Режим']) & (df['Режим'] <=
q_high['Режим'])]
    df = df[(df['Рпр(ТМ)'] >= q_low['Рпр(ТМ)']) & (df['Рпр(ТМ)'] <=
q_high['Рпр(ТМ)'])]
    df = df[(df['Рзаб(Рпр)'] >= q_low['Рзаб(Рпр)']) & (df['Рзаб(Рпр)'] <=
q_high['Рзаб(Рпр)'])]
    df = df[(df['Рзаб(иссл)'] >= q_low['Рзаб(иссл)']) & (df['Рзаб(иссл)'] <=
q_high['Рзаб(иссл)'])]

    df.to_csv('clean_holes.csv', index=False, sep=';', encoding='cp1251')

def normalization():
    data = pd.read_csv('clean_holes.csv', sep=';', engine='python')
    df = pd.DataFrame(data)

    print(f"DataFrame:\n{df}\n")
    print(f"column types:\n{df.dtypes}")

    holes_list = []

    col_List= df['Скважина'].tolist()
    num = 1
    for i in range(len(col_List)):
        if (i == 0) or (col_List[i] != col_List[i-1]):
            holes_list.append(col_List[i])
            num += 1

        columns = ['Рпр(ТМ)', 'Рзаб(Рпр)', 'Рзаб(Нд)', 'Рзаб(иссл)']

    for hole in holes_list:
        df1 =  df[lambda df: df['Скважина'] == hole]

        for col in columns:
            col_List= df1[col].tolist()

            for i in range(len(col_List)):
                delta = len(col_List) - i
                left = col_List[i-1]+col_List[i-2]+col_List[i-3]+col_List[i-
4]+col_List[i-5]
                if delta == 1:
                    right =
col_List[0]+col_List[1]+col_List[2]+col_List[3]+col_List[4]
                if delta == 2:
                    right =
col_List[i+1]+col_List[0]+col_List[1]+col_List[2]+col_List[3]
                if delta == 3:
                    right =
col_List[i+1]+col_List[i+2]+col_List[0]+col_List[1]+col_List[2]
                if delta == 4:
                    right =
col_List[i+1]+col_List[i+2]+col_List[i+3]+col_List[0]+col_List[1]
                if delta == 5:
                    right =
col_List[i+1]+col_List[i+2]+col_List[i+3]+col_List[i+4]+col_List[0]
                if delta >= 6:
                    right =
col_List[i+1]+col_List[i+2]+col_List[i+3]+col_List[i+4]+col_List[i+5]
                col_List[i] = (left + right) / 10

            df1[col] = col_List
            df[lambda df: df['Скважина'] == hole] = df1

    max = df.max()
```

```
    df['Рпр(ТМ)'] = df['Рпр(ТМ)'].apply(lambda x: round(x / max['Рпр(ТМ)'],
3))
    df['Рзаб(Рпр)'] = df['Рзаб(Рпр)'].apply(lambda x: round(x /
max['Рзаб(Рпр)'], 3))
    df['Рзаб(Нд)'] = df['Рзаб(Нд)'].apply(lambda x: round(x /
max['Рзаб(Нд)'], 3))
    df['Рзаб(иссл)'] = df['Рзаб(иссл)'].apply(lambda x: round(x /
max['Рзаб(иссл)'], 3))

    df.to_csv('normal_holes.csv', index=False, sep=';', encoding='cp1251')

def panels_export(current_file, current_path):
    file_name = current_file
    #xls = pd.ExcelFile('Данные для исследований.xlsx')
    #print(xls.sheet_names) # имена листов в считанном файле
    #print(len(xls.sheet_names)) # количество листов в считанном файле

    #df = pd.read_excel('Данные для исследований.xlsx')
    #df = df.set_index('Дата замера')

    df = read_all_sheets(file_name)
    #df_copy = df.copy()
    df.copy()
    df.sort_index(inplace=True)

    ''' Если появляется ошибка в данных способа эксплуатации и режиме
заменить'''
    #what_replaced_x1 = replace_text_values_in_x(df, x1)
    #what_replaced_x2 = replace_text_values_in_x(df, x2)
    replace_text_values_in_x(df, x1)
    replace_text_values_in_x(df, x2)

    """ Здесь порядок другой!!!
    см. текущий порядок в возвращаемом словаре

    df.loc[df['Способ эксплуатации'] == 'Газлифт', 'Способ эксплуатации'] = 1
    df.loc[df['Способ эксплуатации'] == 'Фонтанный', 'Способ эксплуатации'] =
2
    df.loc[df['Способ эксплуатации'] == 'Электропогружным насосом', 'Способ
эксплуатации'] = 3
    df.loc[df['Способ эксплуатации'] == 'По НКТ с пакером', 'Способ
эксплуатации'] = 4
    df.loc[df['Способ эксплуатации'] == 'По НКТ с воронкой', 'Способ
эксплуатации'] = 5
    df.loc[df['Способ эксплуатации'] == 'Установка электроцентробежная для
подачи воды', 'Способ эксплуатации'] = 6
    df.loc[df['Способ эксплуатации'] == 'Прочие способы эксплуатации',
'Способ эксплуатации'] = 7
    df['Способ эксплуатации'] = df['Способ эксплуатации'].fillna(8)

    df.loc[df['Режим'] == 'АПВ', 'Режим'] = 1
    df.loc[df['Режим'] == 'ПДФ', 'Режим'] = 2
    df.loc[df['Режим'] == 'ПКВ', 'Режим'] = 3
    df['Режим'] = df['Режим'].fillna(4)
    """

    all_data = df.copy()

    if os.path.exists('panels/' + current_path):
        shutil.rmtree('panels/' + current_path)
        os.mkdir('panels/' + current_path)
    else:
        os.mkdir(current_path)
```

```
    # Чистим лишние пробулы в столбце "Скважина"
    # Это классная команда для простого преобразования данных. Определяете
словарь,
    # в котором «ключами» являются старые значения, а «значениями» – новые
значения:
    cleaning_map = lambda x: str(x).strip()
    all_data[hole] = all_data[hole].map(cleaning_map)

    all_data = all_data[[hole, x1, x2, y1, y2, y3, y4]]

    def draw_plot(name_hole, data_to_draw):
        formatter = ticker.FormatStrFormatter("%.0f")
        fig, axes = plt.subplots(1,2)
        axes[0].plot(data_to_draw[y1], label=y1)
        axes[0].plot(data_to_draw[y2], label=y2)
        axes[0].plot(data_to_draw[y3], label=y3)
        axes[0].plot(data_to_draw[y4], label=y4)
        axes[0].legend()
        axes[0].set_title('Измерения давлений')
        axes[0].set_xlabel(date, fontsize=12)
        axes[0].set_ylabel('P', fontsize=12)
        axes[1].plot(data_to_draw[x1], label=x1)
        axes[1].plot(data_to_draw[x2], label=x2)
        axes[1].legend()
        axes[1].set_title(x1+' и '+x2)
        axes[1].set_xlabel(date, fontsize=12)
        axes[1].set_ylabel('№', fontsize=12)
        axes[1].yaxis.set_major_formatter(formatter)
        fig.set_figwidth(15)
        fig.set_figheight(8)
        fig.suptitle(hole + ' ' + str(name_hole))
        fig.autofmt_xdate()
        plt.legend()
        #plt.show()
        plt.savefig('panels/' + current_path + '/gridJ_'+ name_hole +'.png',
dpi=100, format='png')

    list_of_holes = pd.unique(all_data[hole]).tolist()
    count_empty_data = 0
    list_empty_data = []
    cleaned_data = pd.DataFrame()
    for _hole in list_of_holes:
        df_to_draw = all_data[all_data[hole] == _hole][[x1, x2, y1, y2, y3,
y4]]
        temp_df = df_to_draw[[y1, y2, y3, y4]].dropna(axis=0, how='all')
        if (not temp_df.empty):
            cleaned_data = cleaned_data.append(df_to_draw)
            draw_plot(_hole, df_to_draw)
            """ Эта часть кода вынесена в функцию
            fig, axes = plt.subplots(1,2)
            axes[0].plot(df_to_draw[y1], label=y1)
            axes[0].plot(df_to_draw[y2], label=y2)
            axes[0].plot(df_to_draw[y3], label=y3)
            axes[0].plot(df_to_draw[y4], label=y4)
            axes[0].legend()
            axes[0].set_title('Измерения давлений')
            axes[1].plot(df_to_draw[x1], label=x1)
            axes[1].plot(df_to_draw[x2], label=x2)
            axes[1].set_title('Настройка')
            axes[1].legend()
            fig.set_figwidth(15)
            fig.set_figheight(8)
            plt.title(_hole)
            plt.show()
```

```
            """
        else:
            #print('Для скважины '+ _hole +' нет данных по давлению!')
            list_empty_data.append(_hole)
            count_empty_data += 1
    print('Количество скважен без данных по давлению: '+
str(count_empty_data))
    """
    print('Список скважен без данных по давлению:')
    i=1
    for _hole in list_empty_data:
        print(str(i)+'. '+_hole)
        i+=1
    """

if os.path.exists('holes'):
    separate_export('Данные для исследований.xlsx','0_null_data_holes')
else:
    os.mkdir('holes')
    separate_export('Данные для исследований.xlsx','0_null_data_holes')

if os.path.exists('panels'):
    panels_export('Данные для исследований.xlsx','0_null_data_panels')
else:
    os.mkdir('panels')
    panels_export('Данные для исследований.xlsx','0_null_data_panels')
print("вывод null data завершен")

parcing_and_filling()
print("данные raw_data заполены")
INFLUX_INPUT('raw_holes.csv', 'raw_data', 0)
print("вывод raw_data в Influx завершен")
convert_csv_to_xlsx('raw_holes')
print("raw_data еонвертирован")

separate_export('raw_holes.xlsx','1_raw_data_holes')
panels_export('raw_holes.xlsx','1_raw_data_panels')
print("вывод raw_data завершен")

cleaning()
print("данные clean_data заполены")
INFLUX_INPUT('clean_holes.csv', 'clean_data', 1)
print("вывод clean_data в Influx завершен")
convert_csv_to_xlsx('clean_holes')
print("clean_data еонвертирован")

separate_export('clean_holes.xlsx','2_clean_data_holes')
panels_export('clean_holes.xlsx','2_clean_data_panels')
print("вывод clean_data завершен")

normalization()
print("данные normal_data заполены")
INFLUX_INPUT('normal_holes.csv', 'normal_data', 1)
print("вывод normal_data в Influx завершен")
```