Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

Институт математики и механики им. Н. И. Лобачевского

Кафедра математического анализа

Направление: 02.03.01 – «Математика и компьютерные науки»

Профиль: математическое и компьютерное моделирование

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Машинное обучение в анализе геофизических данных

Студент 4 курса
Группы 05-503
« ___» _____2019г. _____А.С.Степанов

Научный руководитель
Кандидат физико-математических наук
« ___» _____2019г. _____А.А.Новиков

Заведующий кафедрой математического анализа
Доктор физико-математических наук, профессор
« ___» _____2019г. _____С.Р.Насыров

Казань 2019

Содержание

1 Введение

На сегодняшний момент при геофизических исследованиях скважин набирает популярность применение машинного обучения и нейросетей. К геофизическим исследованиям относят:

- определение разрезов скважин - стратиграфия
- определение осадочных пород, их состава, строения и распространения - литология
- исследование характера насыщения и динамики флюидов в скважине - определение коллекторов
- определение пористости, проницаемости и нефтегазонасыщенности пласта

Целью данной работы является демонстраци возможного применения машинного обучения для определения стратиграфии и литологии в разрезах скважин. Стоит указать, почему именно возможность обучать машину на данный момент является одним из самых перспективных аспектов в области геофизики.

Геофизические исследования начинаются с бурения скважин и замеров различных каротажей в стволе скважины, далее данные с зондов, которые занимались замерами, преобразуются в кривые. На следующем этапе эти кривые исследуются экспертом, который согласно теоретическим данным проставляет значения: если это стратиграфия - отмечает кровлю и подошву горизонта(разрера), если литология - код определяемой породы. Далее, так как скважин бурят достаточно много, после определения экспертных данных, находят схожие аномалии в поведении кривых в остальных скважинах и проставляют экспертные данные у них.

Для ускорения работы применяют конечные алгоритмы, который зачастую опирается на конкретные значения кривой, чтобы определять её поведение. Данный подход отлично работает и отлично переносит экспертные данные на остальные скважины, но имеет ряд недостатков и самый весомый из них - невозможность применения на других месторождениях. И причина здесь в том, что при попытке обработки совсем других данных, нам, скорее всего, придется переписывать весь алгоритм, что приводит к очень долгому анализу новых месторождений.

Относительно новым решением этой проблемы стали машинное обучение и нейросети, которые позволяют переносить одно и тоже решение, меняя при этом только сбор признаков с кривых. Достаточно лишь подобрать новые признаки и подкорректировать выбранную модель, а все остальное сделает машина. В качестве задачи проверим целесообразно ли применять машинное обучение для опрделения стратиграфии и литологии, и насколько ответы машины отличаются от ответов эксперта.

Так как кривая представляет из себя массив значений, определение характера её поведения сводится к различным подходам математического и статистического анализа:

- нахождение зависимостей между кривыми(корреляция)
- определение скорости изменения значений(градиент)
- нахождение экстремумов
- и другие методы изучения кривых

Также для описания поведения можно использовать приближенные решения, такие как:

- интерполяция кривых
- экстраполяция
- сглажка
- разложение кривой в спектр
- и другие методы

Многие кривые имеют определенный физический смысл в той или иной задаче геофизики, подробнее о том, какие кривые и почему использовались для стратиграфии или литологии, опишем далее.

2 Материалы и методы

2.1 Список используемых методов, библиотек и классов

1. numpy - модуль для работы с массивами, а также с численными и статистическими вычислениями
2. lasio - модуль для работы с "*.las" файлами.
3. scipy - интерполяция, сглаживание, статистические и другие функции для анализа.
4. sklearn - библиотека используемых моделей для машинного обучения, функций метрики и оценки
5. pandas - библиотека для анализа данных и чтения "*.xlsx" файлов
6. os - модуль для работы с файлами
7. matplotlib - библиотека для визуализации данных
8. seaborn - библиотека для визуализации данных
9. math - модуль, содержащий математические функции
10. tqdm - модуль для визуализации выполнения кода
11. codecs - работа с файловыми кодировками
12. sys - работа с данными о текущей используемой системе
13. re - регулярные выражения
14. pickle - модуль сохранения и загрузки Python объектов в файлы
15. stats - класс статистики
16. LabelEncoder - класс преобразования истинных предсказаний
17. GradientBoostingClassifier - классификационная модель
18. SVC - классификационная модель
19. AdaBoost - классификационная модель
20. LogisticRegression - классификационная модель

2.2 Используемое оборудование

Язык разработки: Python 3.7
Используемые IDE: PyCharm Professional 2019.1.1, Jupyter Notebook 5.7.8
Данные о системе:
Тип системы: 64-разрядная операционная система, процессор x64
Процессор: Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz 1.70 GHz
Видеоадаптер: NVIDIA GeForce 820M - 2Гб / DDR3 / 1800 МГц
ОЗУ: 4 Гб
Файл подкачки: 8 Гб

3 Предобработка данных

3.1 Необходимость предобработки

Перед анализом данных и посроением модели требуется обработать входные данные. Файлы с информацией о кривых по каждой скважине обычно поступают в разрозненном виде, где в худшем случае - кривая разбита не несколько файлов, которые суммарно образуют несколько, возможно пересекающихся кривых. Данные в таком виде нельзя применять в моделях стратиграфии или литологии, так как ценность информации о раздробленной кривой, стремится к нулю из за малого интервала анализа. Данные приходят в формате ".las а обработка файлов происходит в два этапа:

1. Объединение разрозненных файлов в одну скважину.
2. Сшивка кривых по имени в одну целую.

3.2 Загрузка, отладка и объединение файлов в скважины

Разберем каждый этап более подробно. На первом этапе возникает проблема с кодировкой файлов, а также, возможно, с их неправильной семантикой. Для решения проблемы с кодировкой, использовались регулярные выражения для поиска неопознанных символов в файле, а также был найден список всех доступных в Python 3.+ кодировок(Рис. 1).

```
#Доступные Python кодировки
encodincs = ['ascii', 'utf_8', 'cp1251', 'cp866', 'utf_8_sig', 'cp1252', 'big5',
             'big5hkscs', 'cp037', 'cp273', 'cp424', 'cp437',
             'cp500', 'cp720', 'cp737', 'cp775', 'cp850', 'cp852', 'cp855', 'cp856',
             'cp857', 'cp858', 'cp860', 'cp861', 'cp862', 'cp863', 'cp864', 'cp865',
             'cp869', 'cp874', 'cp875', 'cp932', 'cp949', 'cp950', 'cp1006',
             'cp1026', 'cp1125', 'cp1140', 'cp1250', 'cp1253',
             'cp1254', 'cp1255', 'cp1256', 'cp1257', 'cp1258', 'cp65001', 'euc_jp',
             'euc_jis_2004', 'euc_jisx0213', 'euc_kr', 'gb2312', 'gbk', 'gb18030',
             'hz', 'iso2022_jp', 'iso2022_jp_1', 'iso2022_jp_2', 'iso2022_jp_2004',
             'iso2022_jp_3', 'iso2022_jp_ext', 'iso2022_kr', 'latin_1', 'iso8859_2',
             'iso8859_3', 'iso8859_4', 'iso8859_5', 'iso8859_6', 'iso8859_7', 'iso8859_8',
             'iso8859_9', 'iso8859_10', 'iso8859_11', 'iso8859_13', 'iso8859_14',
             'iso8859_15', 'iso8859_16', 'johab', 'koi8_r', 'koi8_t', 'koi8_u', 'kz1048',
             'mac_cyrillic', 'mac_greek', 'mac_iceland', 'mac_latin2', 'mac_roman',
             'mac_turkish', 'ptcp154', 'shift_jis', 'shift_jis_2004', 'shift_jisx0213',
             'utf_32', 'utf_32_be', 'utf_32_le', 'utf_16', 'utf_16_be', 'utf_16_le', 'utf_7']
```

Рис. 1: Кодировки

Кодировка считается подходящей - если количество неопознанных символов равно нулю. В противном случае будем использовать кодировку с наименьшим количеством неопознанных символов. Важно уточнить что такое неопознанный символ. Ориентиром являются русскоязычные компании, которые, соответственно, используют в .las файлах русские символы(Рис. 2).

Вообще кодировок, "понимающих"русский язык - всего две: cp1251 и oem866, но бывают случаи, когда разные части файла записаны в разных кодировках, именно для этого существует общий случай. После проверки кодировки, файл можно проверить на наличие ошибок семантики файла.

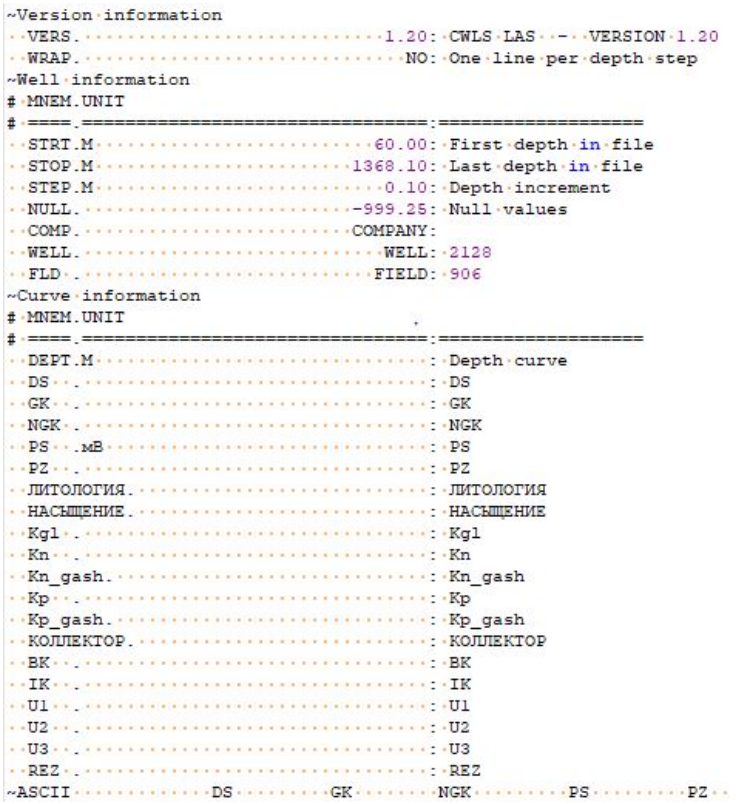

Рис. 2: Пример *.las файла

Так как для считывания данных из файла используется модуль "lasio данные в файле должны быть предоставлены в определенном виде:

1. блок значений кривых начинается с A или ASCII и должен быть одинакового размера для каждой кривой
2. блок данных о скважине должен содержать имя скважины, вся информация представляется в виде <имя>. <данные>:<описание>
3. блок информации о кривых должен содержать столько же кривых, сколько колонок(кривых) в блоке A, информация также должна быть оформлена в виде <имя>. <данные>:<описание>

Ошибки пункта 1 чаще всего не подлежат автоматическому исправлению, так как рискуем получить неверные данные при такой обработке. В то время как пункты 2 и 3 можем исправить, попытавшись привести все строки к виду <имя>. <данные>:<описание> После этого получаем файлы, готовые к объединению в скважины. Эта часть делится на несколько пунктов:

1. Объединение файлов по имени скважины
2. Создание глобальной сетки глубины для каждой скважины
3. Интерполяция кривых на новую сетку глубины

Для объединения файлов используется контрукция словаря в Python, для нахождения глобальной сетки достаточно найти абсолютный минимум и максимум глубины по всем файлам, шаг по глубине равен 0.1м, так как эта величина является стандартом. Третьим пунктом, чтобы не потерять информацию, применяется линейная интерполяция со старой сетки глубины на глобальную, листинг ниже.

```python
# Интерполяция
def newMesh(CRV, DEPT, new_mesh):
    if abs(DEPT[0] - new_mesh[0]) < 1e-5 and len(new_mesh) == len(DEPT):
        return CRV
    else:
        new_crv = np.interp(new_mesh, DEPT, CRV, left=np.nan, right=np.nan)
        return new_crv
```

Рис. 3: Интерполяция кривой на новую сетку глубины

3.3 Увязка кривых в скважине

3.3.1 Построение матрицы отношений

После этого можем приступить ко второму пункту обработки данных - сшивке кривых. После пункта 1 имеем кривые одинаковой длины, сгруппированные по имени скважины. Для того, чтобы построить модель стратиграфии, искомые данные должны иметь одну, наиболее содержательную кривую каждого типа. Для этого требуется произвести увязку кривых. Для наиболее корректной работы увязка производится в два этапа:

1. Увязка пересекающихся частей сгруппированных кривых

2. Сшивка непересекающихся частей сгруппированных кривых

Разберем эти пункты более подробно. Наиболее трудоемким является первый пункт, поясним работу алгоритма.

В начале цикла строится квадратная матрица размером m, каждый элемент которой a_{ij} хранит информацию о интервале пересечения кривых с номерами i и j, код на рисунках 4 и 5.

```
def takeMatrix(self,wellName, curvesToMerge):
    dept = self.wells[wellName].curves['DEPT']
    a = []
    #Создать матрицу
    for i in range(0,len(curvesToMerge)):
        mas = []
        for k in range(0,i+1):
            mas.append(IntersectionInformation(None,None,None,None,True,None,None))
        for j in range(i + 1, len(curvesToMerge)):
            try:
                curveUpper = None
                curveDown = None
                dept1 = dept[np.isfinite(curvesToMerge[i].dataX)]
                dept2 = dept[np.isfinite(curvesToMerge[j].dataX)]
                if len(dept1) != 0 and len(dept2) != 0:
                    if dept1[0] <= dept2[0]:
                        if dept1[0] == dept2[0]:
                            if dept1[-1] >= dept2[-1]:
                                curveUpper = curvesToMerge[i]
                                curveDown = curvesToMerge[j]
                            else:
                                curveUpper = curvesToMerge[j]
                                curveDown = curvesToMerge[i]
                        else:
                            curveUpper = curvesToMerge[i]
                            curveDown = curvesToMerge[j]
                    else:
                        curveUpper = curvesToMerge[j]
                        curveDown = curvesToMerge[i]
                    intersection, length = self.takeIntersection(curveUpper, curveDown,dept)
                    corr = None
                    k = None
                    if length != None:
                        corr, k = self.takeCorrelationAndCoef(curveUpper, curveDown, dept, intersection)
                        isNone = False
                    else:
                        isNone = True
                    mas.append(IntersectionInformation(k, corr, length, intersection, isNone, curveUpper, curveDown))
                else:
                    mas.append(IntersectionInformation(None, None, None, None, True, None, None))
            except Exception as e:
                print(e)
                print('Fail in takeMatrix, well - ', wellName)
        if len(mas) > 0:
            a.append(mas)
    return a
```

Рис. 4: Функция нахождения матрицы

```
class IntersectionInformation():
    def __init__(self,k,corr,length,intersection,isNone,curveUpper,curveDown):
        self.k = k
        self.corr = corr
        self.intersection = intersection
        self.length = length
        self.tested = False
        self.isNone = isNone
        self.curveUpper = curveUpper
        self.curveDown = curveDown
```

Рис. 5: Элемент матрицы

Далее в матрице нужно найти элемент a_{ij} с положительной корреляцией и ближайшим к 1 коэффициетом отношения.

3.3.2 Увязка кривых в пересечении и необходимые условия

Для увязки пересекающихся кривых необходимо выполнение двух условий:

1. Коэффициент отношения кривых в интервале пересечения находится в интервале (0.9;1.1)

2. Корреляция кривых в интервале пересечения - число строго положительное

```
#Основной цикл
indexes, bestK = self.takeBestK(a,trueName)
while bestK != None:
    newCurve, appendLog = self.mergePls(bestK,trueName,wellName)
    if len(newCurve) == 1:
        newCurve = newCurve[0]
        c = bestK.curveUpper
        self.curvesToDelete.append(curvesToMerge.pop(curvesToMerge.index(bestK.curveUpper)))
        self.curvesToDelete.append(curvesToMerge.pop(curvesToMerge.index(bestK.curveDown)))
        if mergeCount > 0:
            newCurve.name = trueName + str(mergeCount)
            newCurve.original_mnemonic = trueName
        else:
            newCurve.name = trueName
            newCurve.original_mnemonic = trueName
        appendLog[-1] = "Сшиты в " + newCurve.name
        self.mergeLog.append(appendLog)
        mergeCount = mergeCount + 1
        newCurve.unit = c.unit
        newCurve.descr = c.descr
        curvesToMerge.append(newCurve)
        a = self.takeMatrix(wellName, curvesToMerge)
    else:
        appendLog[-1] = "Не сшиты"
        self.mergeLog.append(appendLog)
        a[indexes[0]][indexes[1]].tested = True
    indexes, bestK = self.takeBestK(a,trueName)
for i in range(0,len(a)):
    for j in range(i+1,len(a[i])):
        self.mergeLog.append([wellName,a[i][j].curveUpper.name,a[i][j].curveDown.name,
                              str(a[i][j].length),str(a[i][j].corr),str(a[i][j].k),
                              "","","",""])
for curve in curvesToMerge:
    curvesToExit.append(curve)
return curvesToExit
```

Рис. 6: Основная часть цикла увязки

В случае, когда условие 1 не выполняется, но коэффициент отношения находится в интервале от 0.4 до 1.6 - делим интервал пересечения на отрезки по 2м, на каждом находим коэффициент отношения, и берем последний, удовлетворяющий условию 1). Если сшивка не произошла, помечаем элемент a_{ij} как протестированный и более его не трогаем на данном шаге цикла. Если сшивка все таки произошла - необходимо удалить кривые i j из матрицы(Рис. 6), вместо них добавив новую - сшитую кривую, перестроить матрицу и повторить шаг цикла. Так пробегаем по всей матрице, пока есть элементы с положительной корреляцией, отношением в интервале от 0.4 до 1.6 и еще не тестированные элементы. На выходе получим кривые, участвовавшие и нет в увязке, после этого стоит приступить ко второму пункту сшивки кривых - объединение частей без пересечения.

3.3.3 Объединение кривых без пересечения

```
#Основная часть сшивки без пересечения
def mergeNoIntersectionCurves(self,wellName, curvesToMerge, trueName):
    curvesToExit = []
    self.curvesToDelete = []
    mergeCount = 0
    i = 0
    while i < len(curvesToMerge):
        curvei = curvesToMerge[i]
        curveiMerged = False
        haveIntersection1 = self.anyIntersections(curvei, curvesToMerge, wellName)
        if haveIntersection1 == False:
            j = i + 1
            while j < len(curvesToMerge):
                curvej = curvesToMerge[j]
                curvejMerged = False
                haveIntersection2 = self.anyIntersections(curvej,curvesToMerge,wellName)
                if haveIntersection2 == False:
                    mergeCount += 1
                    curveiMerged = True
                    curvejMerged = True
                    newCurve = self.mergeWithoutIntersection(curvei,curvej,wellName)
                    c = curvesToMerge.pop(curvesToMerge.index(curvei))
                    self.curvesToDelete.append(curvesToMerge.pop(curvesToMerge.index(curvej)))
                    self.curvesToDelete.append(c)

                    self.mergeLog.append([wellName,curvei.name,curvej.name,"Сшиты без пересечения"])
                    newCurve.name = trueName + '__' + str(mergeCount)
                    newCurve.descr = c.descr
                    newCurve.unit = c.unit
                    newCurve.original_mnemnic = trueName
                    curvesToMerge.append(newCurve)
                    i = 0
                    break
                else:
                    j += 1
        else:
            i += 1
        if curveiMerged == False:
            i += 1
    for curve in curvesToMerge:
        curvesToExit.append(curve)
    return curvesToExit
```

Рис. 7: Основная часть цикла сшивки кривых без пересечения

На этом этапе необходимо выполнение только одного условия - части, которые нужно объединять, не должны пересекаться с остальными кривыми с соответствующим именем, код на рисунке 7.

3.4 Экспорт готовых файлов

После этих операций имеем кривые, готовые к анализу и последующей обработке в моделях стратиграфии и литологии, для надежности и дальнейшего использования формируем из имеющихся объектов новые ".las"файлы, код ниже.

```
for wellName in tqdm(wellsToMerge.keys()):
    lasFile = lasio.LASFile()
    lasFile.well['WELL'].value = wellName
    lasFile.well['NULL'].value = -999.25
    lasFile.add_curve('DEPT', data=wellsToMerge[wellName].curves['DEPT'])
    for curve in wellsToMerge[wellName].curvesList:
        if curve.name != 'DEPT':
            lasFile.add_curve(curve.name, data=curve.dataX)
    with codecs.open(converter.replaceSlash('lasFiles/' + wellName + '.las'), 'w', encoding='utf_8') as f:
            lasFile.write(f, version=2, STEP=np.round(lasFile.index[1] - lasFile.index[0], 2))
```

Рис. 8: Выгрузка файлов

4 Стратиграфия

4.1 Постановка задачи и объяснение с точки зрения геофизики

4.1.1 Стратиграфия как наука

Стратиграфия — геологическая дисциплина, которая изучает пространственные и временные соотношения пластующихся толщ горных пород Земной коры. Элементарным объектом изучения стратиграфии является слой. Стратиграфические исследования включают:

1. стратиграфическое расчленение
2. стратиграфическую корреляцию

Одной из задач стратиграфического расчленения является определение выделение в конкретном разрезе отдельных толщ и слоев, которые отличаются определенными признаками - именно эту задачу и будем преследовать.

Для определения стратиграфии используют кривые радио и электро каротажей.

4.1.2 Электрокаротаж в стратиграфии

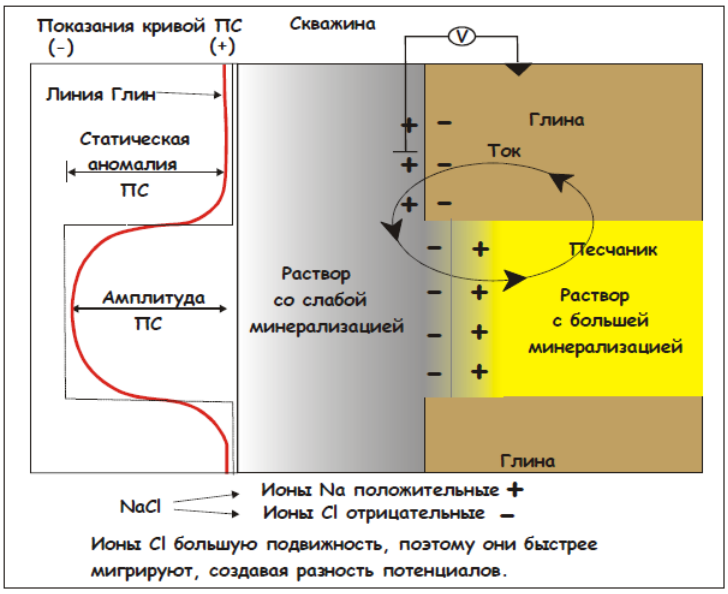

Рис. 9: Пример кривой PS в стратиграфии

Исследование на основе изучения электрокаротажа основываются на том, что при бурении скважины между помывочной жидкостью и пластовыми водами возникают потенциалы спонтанной поляризации, которые непрерывно фиксируются зондом. Измерение кривой PS производится двухэлектродной установкой, один электрод помещается на поверхности, а второй опускается в ствол скважины. Различие значений позволяет диагностировать основные типы глинистых и карбонатных отложений(Рис. 9).

4.1.3 Радиокаротаж в стратиграфии

Наличие глинистого материала ведет к увеличению радиоактивности горных пород в связи с высокой адсорбционной способностью глин. Положительные аномалии ГК могут отмечаться например в обогащенных радиоактивными минералами песчаных породах, пример аномальных значений кривой на рисунке ниже.

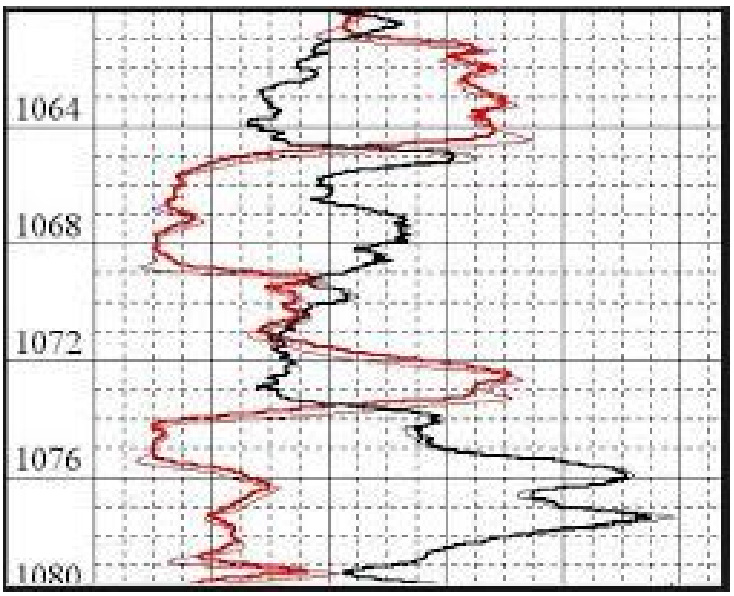

Рис. 10: Пример кривой GK в стратиграфии

Несмотря на то, что в общем виде интенсивность гамма-излучения пород пропорциональна содержанию в них радиоактивных минералов, она зависит и от плотности самой породы, так как с увеличением плотности возрастает поглощение гамма-излучения породой и соответственно уменьшается поток, измеряемый зондом. Наконец, на интенсивность гамма-излучения оказывает влияние радиоактивность пластовых вод. В частности, повышенной радиоактивностью характеризуются высокоминерализованные хлоридно-кальциевые воды.

4.1.4 Загрузка параметров и статистическая оценка пластов

Перед анализом данных считаем имеющиеся файлы скважин а также данные обучающей выборки: искомые значения кровли и подошвы нужного горизонта, листинг ниже.

```
folder = "Full_las"

filePaths = ([joinpath(folder, file) for file in listdir(folder) if (isfile(joinpath(folder, file))
              and file.lower().endswith('las'))])

wells = [Well(filepath) for filepath in tqdm(filePaths[:])]
```

```
100%|██████████| 80/80 [01:05<00:00,  1.99it/s]
```

```
data = pd.read_excel('Akan_Tab.xlsx')
```

```
data.head()
```

	Скважина	Статус	КС	ПС	ДС	ИК	БК	ГК	НК	КС.1	...	GKue	NGKue	Tau	Rk_Speed	Alpha	Top_C2vrT	Top_C2vrK	Top_C2bsh	Bot_C2
0	1892	готово	есть	есть	есть	есть	есть	есть	есть	200	...	1006.0	9600.0	3.0	400.0	0.15	1141.8	1160.9	1187.9	12
1	1899	готово	есть	есть	есть	есть	есть	есть	есть	200	...	955.0	8600.0	3.0	400.0	0.15	1048.8	1069.2	1096.2	11
2	1900	готово	есть	есть	есть	есть	есть	есть	есть	200	...	1950.0	15000.0	3.0	400.0	0.15	1023.0	1041.4	1066.6	11
3	1901	готово	есть	есть	есть	есть	есть	есть	есть	200	...	1904.0	15500.0	3.0	400.0	0.15	1144.0	1163.6	1192.0	12
4	1902	готово	есть	есть	есть	есть	есть	есть	есть	200	...	1001.0	9000.0	3.0	400.0	0.15	1165.6	1187.5	1220.6	12

5 rows × 43 columns

Рис. 11: Загрузка файлов и параметров

Для начала статистически определим минимальный и максимальный размер горизонта, на этом этапе можем выявить аномальные скважины(Рис. 12), которые не могут быть проанализированы. Имеем 78 скважин, из которых после данной проверки выявим 2 аномальные.

```
m1 = np.array([well.tops[0] for well in wells])
m2 = np.array([well.tops[-2] for well in wells])
mx = np.nanmin(m2-m1)
mn = np.nanmax(m2-m1)
s = np.nanmean(m2-m1)
plt.plot(m2-m1)
print(mx,mn,s)
```

```
43.0 142.70000000000005 51.03587500000002
```

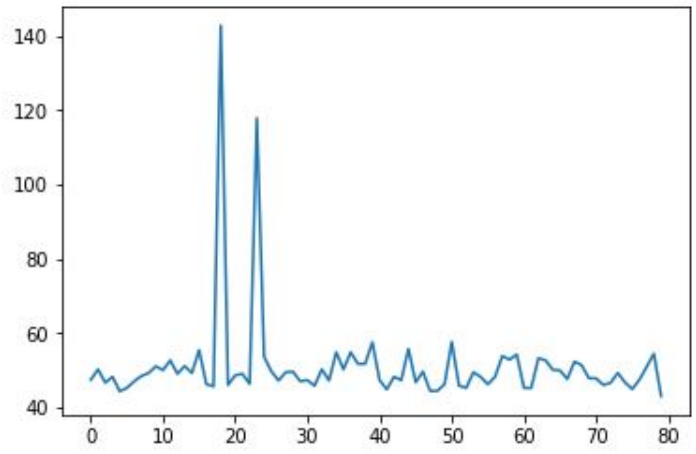

Рис. 12: Распределение мощностей пластов

4.2 Анализ данных

Приступим к анализу и определению закономерностей для выявления шаблона. Для анализа будем использовать кривую гаммакаротажа - GK и электрокаротажа PS.

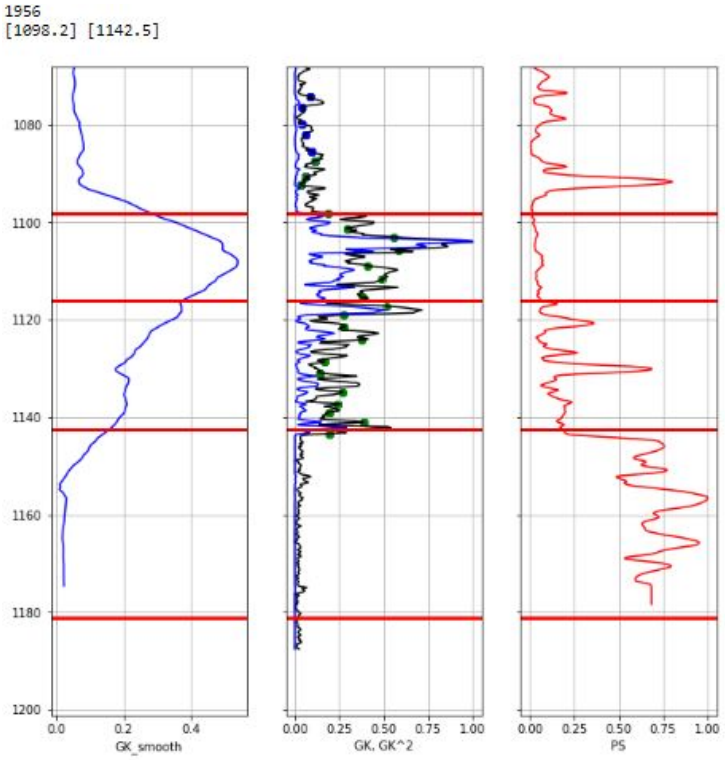

Рис. 13: Кривые для анализа

Для более явного определения возведем кривую GK во вторую степень, а также сгладим методом скользящего среднего. На рисунке 13 можно заметить следующие закономерности:

1. На кривой GK и её квадрате в первой трети искомого пласта заметен ярковыраженный максимум
2. На сглаженной кривой GK в окрестностях второй точки в 10 метрах до точки заметны слабо изменяющиеся средние значения
3. На кривой GK и её сглаженной вариации в окрестностях второй точки в 10 метрах после заметно резкое уменьшение значений вплоть до минимума
4. На кривой PS в окрестностях второй точки в 10 метрах после кривая имеет большое значение среднеквадратичного отклонения

В качестве признаков для каждого кандидата $[p_1,p_2]$ будем использовать следующие признаки:

1. На первой трети кандидата у кривой GK высчитаем сумму среднего и максимального значений на отрезке
2. На отрзке 4 метра до первой точки у кривой GK высчитаем среднее значение
3. На первой трети кандидата у квадрата кривой GK высчитаем сумму среднего и максимального значений на отрезке
4. На отрезке 4 метра до первой точки у квадрата кривой GK высчитаем среднее значение
5. На отрезке за 2 метра до второй точки у кривой GK найдем максимум
6. На отрезке спустя 2 метра после второй точки у кривой GK найдем минимум
7. На отрезке спустя 10 метров после второй точки у кривой GK найдем минимум

8. На отрезке 3 метра до и 3 метра после второй точки найдем полусумму максимума и минимума сглаженной кривой GK

9. На отрезке спустя 10 метров после второй точки найдем среднеквадратичное отклонение сглаженной кривой GK

10. На отрезке спустя 10 метров после второй точки найдем среднеквадратичное отклонение кривой GK

11. На отрезке спустя 10 метров после второй точки найдем среднеквадратичное отклонение кривой PS

4.3 Разделение на обучающую и тестовую выборки, определение задачи и модели

4.3.1 Разделение на обучающую и тестовую выборки

Определимся с классом задачи, так как нужно установить однозначное соответствие "горизонт"или "не горизонт". Будем рассматривать стратиграфию как задачу классификации на два типа 0,1. Оставшиеся 76 скважин поделим на 60 скважин для обучения и 16 для тестовой проверки, код на рисунке ниже.

```
print(len(wells))
wells_train = wells[:60]
wells_test = wells[60:]
```

```
78
```

```
print(len(wells_train) , len(wells_test))
```

```
60 18
```

Рис. 14: Разделение выборки

4.3.2 Определение задачи и модели

Для классификации будем использовать машинное обучение, в качестве модели возьмем метод классификации опорных векторов(Рис. 15), основанный на построении гиперплоскости и разделении множества элементов на два класса. При тестировании модели получаем результат 78 процентов попаданий.

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
sv = SVC(probability=True, C=2, class_weight='balanced', cache_size=2000,
             verbose=True, kernel='rbf',gamma = 'auto', degree=7, shrinking=True)
```

```
sv.fit(features_train, labels_train)
```

```
[LibSVM]

SVC(C=2, cache_size=2000, class_weight='balanced', coef0=0.0,
  decision_function_shape='ovr', degree=7, gamma='auto', kernel='rbf',
  max_iter=-1, probability=True, random_state=None, shrinking=True,
  tol=0.001, verbose=True)
```

```
features_test, labels_test = get_TrainingSet(wells_test)
```

Рис. 15: Модель класиификации методом опорных векторов

Для сравнения другие модели имели показатель, не превышающий модель SVC, наглядная иллюстрация ниже.

```
Scores : %
SVC : 0.78
Ada : 0.67
Linear : 0.56
Gradient : 0.78
```

Рис. 16: Сравнение показателей разных моделей

5 Литология

5.1 Постановка задачи и объяснение с точки зрения геофизики

5.1.1 Литология как наука

После определния стратиграфии, можно приступить к определению литотипов. Литология - наука о составе, структурах и расположении осадочных пород, таких как песчаник, глина, руды и прочее. С помощью литологии также можно определить местоположение жидкости в скважине, что нам и нужно. В общем случае, будем рассматривать литологию как разделение всего ствола скважины на три класса: песчаник, глинистые отложения и карбонаты. Для определения обычно используют кривые электро и радио каротажей. Будем использовать следующие кривые для определения литологии.

5.1.2 Кривая NGK в литологии

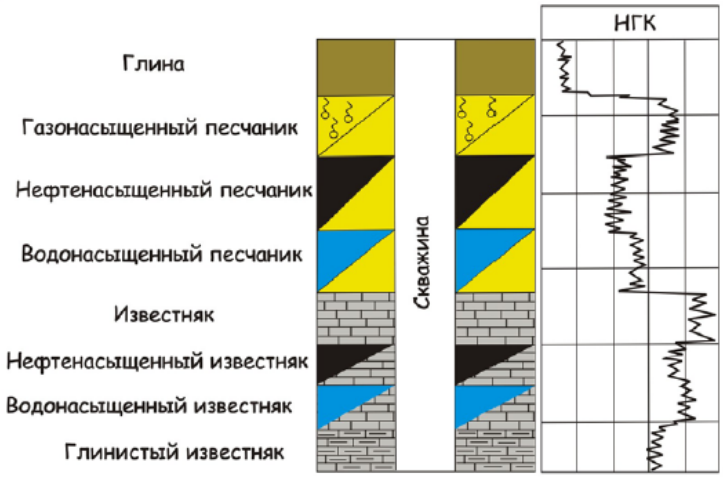

Рис. 17: Пример кривой NGK для определения литологии

NGK - нейтронный гамма каротаж(Рис. 17), показывает исскуственную радиоактивность пород. Суть данного метода основывается на том, что ядра водорода являются своеобразным замедлителем искусственного "ответного"излучения. При изучении показаний опираются на то, что для плотных пород с низким водородосодержанием (плотные известняки, карбонатизированные песчаники и др.) наблюдаются повышенные значения показаний НГК, в то время как для глинистых пород, обладающих максимальной водонасыщенностью характерны минимальные показания.

5.1.3 Кривая GK в литологии

GK - гамма каротаж(Рис. 18), показывает естественную радиоактивность пород в скважине, образуемую за счёт радиоактивных изотопов глинистых минералов: слюды, некоторых фосфатов и полевого шпата. Суть в том, что такие радиоактивные элементы как калий 40, уран и торий в большем количестве находятся в глинистых слоях, но помимо глин, значительной радиоактивностью обладают полимиктовые песчаники - породы, состоящие из более чем двух пород сразу, одними из представителей которых являются аркозовые песчаники - песчаники, с заметным преобладанием полевых шпатов, в состав которых также нередко входит слюда, о радиоактивности которых уже говорилось выше.

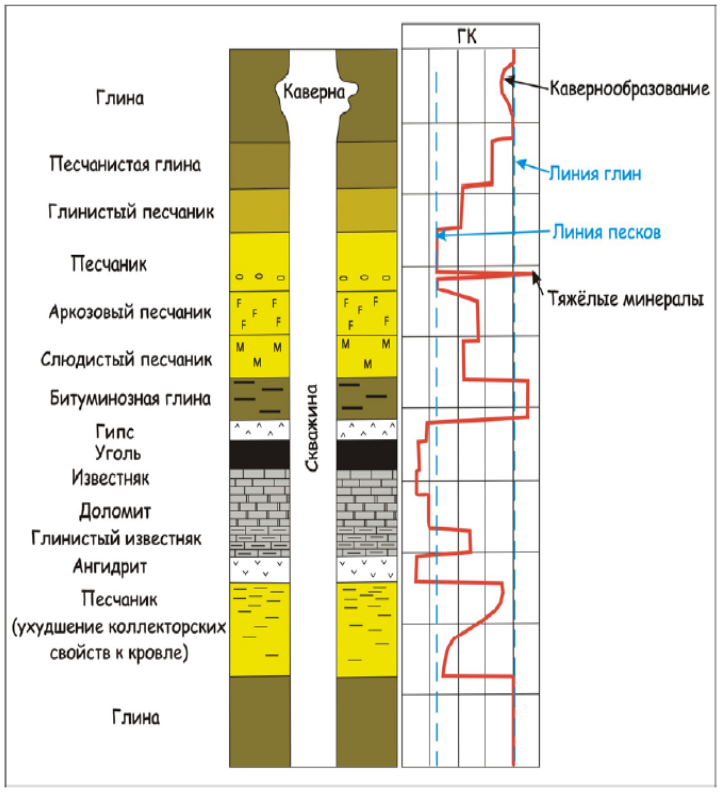

Рис. 18: Пример кривой ГК для определения литологии

5.2 Загрузка параметров и анализ данных

5.2.1 Загрузка файлов и параметров

Первое что следует сделать - загрузить новые данные для обучения нашей модели, константы корректировки, экспертные данные. После отсева скважин на наличие нужных кривых, имеем 76 готовых к анализу скважин, листинг ниже.

```
folder = "las1"
filePaths = ([joinpath(folder, file) for file in listdir(folder) if (isfile(joinpath(folder, file))
             and file.lower().endswith('las'))])

data = pd.read_excel('Table.xlsx')

wells = [Well(filepath) for filepath in tqdm(filePaths[:])]
```

Рис. 19: Загрузка параметров и файлов

5.2.2 Корректировка кривых и построение вспомогательных

Произведем корректировку кривых GK и NGK и на их основе заполним вспомогательные кривые Kp и Carern(Рис. 20). Производить "обрезку"кривых по нужному горизонту не стоит, для предсказаний важно учитывать не только закономерности самого шаблона, но и закономерности ближайший окрестностей, как это использовалось в определении стратиграфии.

```
well.curves["AG_auto"] = (well.curves["GK"]  - asupps_gk[0]) / (asupps_gk[1] - asupps_gk[0])
well.curves["ANG_auto"] = (well.curves["NGK"] - asupps_nk[0]) / (asupps_nk[1] - asupps_nk[0])
well.curves["Cavern"] = well.curves["DS"]- well.wellData['dNom']

if well.wellData["haveDS"]:
    well.curves["H_ClayCrust"] = (well.wellData["dNom"] - well.curves["DS"]) / 2
    well.curves["H_ClayCrust"][well.curves["H_ClayCrust"] < 0] = 0

well.curves["NK_min"] = np.ones(np.shape(well.curves["DEPT"])) * well.asupps_nk[0]
well.curves["NK_max"] = np.ones(np.shape(well.curves["DEPT"])) * well.asupps_nk[1]
well.curves["GK_min"] = np.ones(np.shape(well.curves["DEPT"])) * well.asupps_gk[0]
well.curves["GK_max"] = np.ones(np.shape(well.curves["DEPT"])) * well.asupps_gk[1]

well.curves['Kp_auto'][ind_vr] = Calc_Phi(well.curves["AG_auto"][ind_vr],
                                          well.curves["ANG_auto"][ind_vr],
                                          well.wellData["counterType"], "vr")

well.curves['Kp_auto'][ind_bash] = Calc_Phi(well.curves["AG_auto"][ind_bash],
                                          well.curves["ANG_auto"][ind_bash],
                                          well.wellData["counterType"], "bash")
```

Рис. 20: Построение кривых Cavern и Kp

Сегментируем и построим новые кривые, которые в дальнейшем будем использовать для сбора признаков(код на рисунке ниже):

1. Разделим кривую на отрезки не менее 5м по точкам локальных экстремумов кривой
2. Каждый такой отрезок сгладим методом скользящего среднего
3. Всему отрезку присвоим среднее значение на отрезке

```
def CalculationNewMesh(WELLS_i):
    def GetCand(_bnd, dpt, tops):
        cand = []
        bnd = list(set(_bnd))
        bnd.append(tops[1])
        bnd.append(tops[3])
        bnd = np.sort(bnd)
        bnd = bnd[np.logical_and(bnd >= tops[1], bnd <= tops[3])]
        i = 0
        while i < len(bnd)-1:
            il = np.where(abs(dpt - bnd[i])   < 1e-3)[0][0]
            ir = np.where(abs(dpt - bnd[i+1]) < 1e-3)[0][0]
            cand.append([il, ir])
            i += 1
        res = []
        if abs(cand[0][0] - cand[0][1]) < 5:
            cand[1] = [cand[0][0], cand[1][1]]
        else:
            res.append(cand[0])
        for c in range(1, len(cand) - 1):
            if abs(cand[c][0] - cand[c][1]) < 5:
                if abs(cand[c-1][0] - cand[c-1][1]) < abs(cand[c+1][0] - cand[c+1][1]):
                    res[-1] = [res[-1][0], cand[c][1]]
                else:
                    cand[c+1] = [cand[c][0], cand[c+1][1]]
            else:
                res.append(cand[c])
        if abs(cand[-1][0] - cand[-1][1]) < 5:
            res[-1] = [res[-1][0], cand[-1][1]]
        else:
            res.append(cand[-1])
        return np.array(res, dtype = int)

    well = copy.deepcopy(WELLS_i)
    seg_name = ["AG_auto"]
    well.boundaries = {}
    for key in seg_name:
        MakeSegs(well, key, 3, 1, well.tops[0]-1, well.tops[3] + 1)
    cand = GetCand(well.boundaries["AG_auto"], well.curves["DEPT"], well.tops)
    return cand
```

Рис. 21: Сегментация кривой

В итоге получим кривые ступенчатого вида, каждая "ступень"которой будет определенным классом. Далее заполним кривые экспертных данных и начнем анализ имеющихся кривых.

5.3 Анализ данных

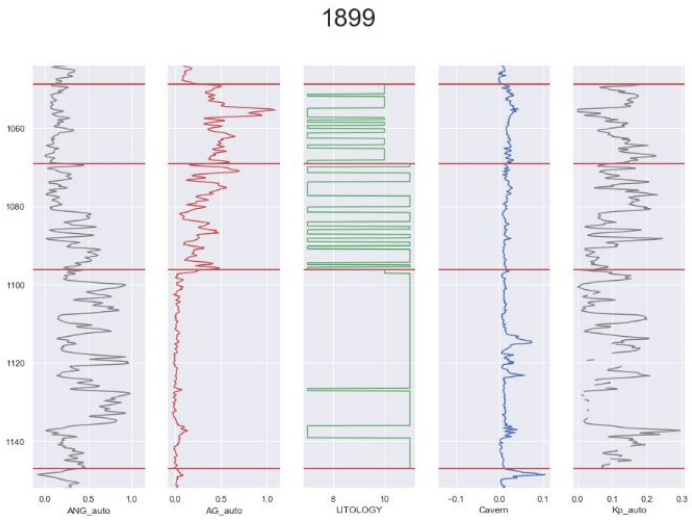

Рис. 22: Кривые для анализа и экспертная кривая литологии

На рисунке 22 заметны следующие закономерности:

1. Кривая GK(AG)
   (a) Заметим, что в максимуме GK наблюдаются нулевые значения экспертных данных.
   (b) На минимуме же значение кривой экспертной литологии равно 12.
   (c) При средних значениях - 10.
2. Кривая NGK(ANG)
   (a) На минимуме значений у кривой экспертных данных также значение равно нулю.
   (b) В точках локального максимума - 12
   (c) При больших значениях значение экспертных данных также равно 12
3. Кривая Kp - от одного горизонта к другому наблюдается смена корреляции между экспертными данными и кривой Kp.
   (a) Первый горизонт - на минимуме наблюдаются нулевые значения литологии
   (b) Второй горизонт - на минимуме - 12
   (c) Третий горизонт - в области резкого изменения значений - нулевые значения.

Так как для всех кривых, участвующих в обучении, используются одни и те же признаки, опишем их только для одной кривой:

1. Для текущего кандидата вычислим среднее значение кривой на отрезке
2. Для текущего кандидата вычислим минимальное значение кривой на отрезке
3. Для текущего кандидата вычислим максимальное значение кривой на отрезке
4. Для текущего кандидата вычислим среднее значение градиента кривой на отрезке
5. Для текущего кандидата вычислим среднее значение квадрата кривой на отрезке
6. Для предыдущего кандидата вычислим среднее значение кривой на отрезке
7. Для предыдущего кандидата вычислим минимальное значение кривой на отрезке
8. Для предыдущего кандидата вычислим максимальное значение кривой на отрезке
9. Для предыдущего кандидата вычислим среднее значение градиента кривой на отрезке
10. Для предыдущего кандидата вычислим среднее значение квадрата кривой на отрезке

11. Для следующего кандидата вычислим среднее значение кривой на отрезке
12. Для следующего кандидата вычислим минимальное значение кривой на отрезке
13. Для следующего кандидата вычислим максимальное значение кривой на отрезке
14. Для следующего кандидата вычислим среднее значение градиента кривой на отрезке
15. Для следующего кандидата вычислим среднее значение квадрата кривой на отрезке

5.4 Разделение на обучающую и тестовую выборки, определение задачи и модели

5.4.1 Разделение на обучающую и тестовую выборки

Так как значения и поведение кривых от класса к классу заметно различаются, и имеют резкие перепады, будем использовать в качестве признаков квадраты и градиенты кривых на каждом кандидате, которыми у нас являются сегменты, построенные ранее, а также средние значения функции, минимумы и максимумы. Суммарно будем использовать 60 признаков со всех нужных кривых.

5.4.2 Определение задачи и выбор модели

В качестве модели для классификации на 3 типа идеально подойдет Gradient Boosting Classifier, все параметры модели подбираем путем подбора на основе ошибок. В качестве результата на 16-ти тестовых скважинах имеем 96% верных предсказаний, пример работы модели на рисунке ниже.

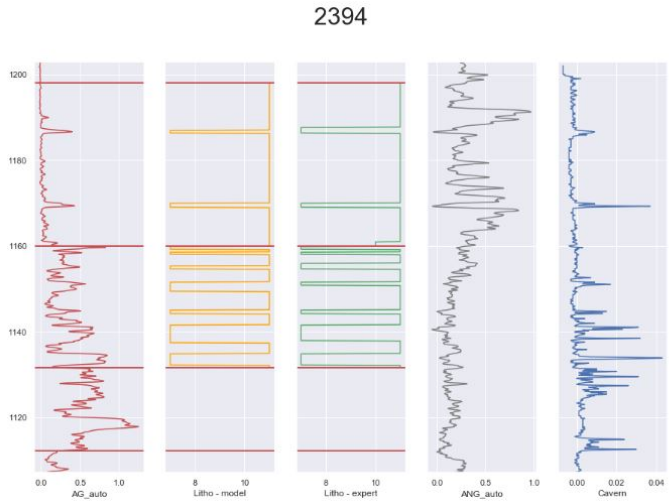

Рис. 23: Демонстрация предсказаний

Другие модели показали себя хуже:

- Gradient - 96.51%
- Ada - 79.73%
- Logistic - 66.0%
- SVC - 82.46%

6 Обсуждение

Во время определения стратиграфии возникали трудности с описанием кривых, так как, обычно, для обучения используют выборки гораздо больших размеров, и это связано с тем, что поведение кривой разнится от скважины к скважине. Из за этого модель не может уловить все возможные варианты, и имея всего 60 верных значений, вероятность ошибки возрастает с увеличением общего количества кандидатов, так на восьмидесяти тысячах наблюдалось всего 56% верных предсказаний. Результат улучшали путем более сильного осреднения кривой, добавляя ограничение на мощность пласта, что снижало количество кандидатов, менял параметры моделей, пробовал различные признаки: среднее значение, квадратичное отклонение, градиент, минимумы и максимумы.

Так или иначе, на практике, при большом количестве скважин, даже относительно небольшой процент верных попаданий - 70%, ускорял обработку данных людьми и позволял применять одни и те же модели в разных месторождениях, изменяя только собираемые признаки.

При этом все равно в обучающей выборке были скважины, отличающиеся поведением в заданных мною окрестностях точек, что приводило либо к неверным результатам, либо переобучению моих моделей.

7 Заключение

В работе были представлены примеры определения литологических разрезов(стратиграфии) и литологии. Нужно заметить, что признаки, которые использовались для обучения и тестирования, подходят исключительно для показанных данных, а параметры моделей выбирались методом подбора, с целью получения наилучшего процента верных попаданий. При этом, для использования похожих методов обычно берут гораздо большую обучающую выборку, количество скважин в которых превышает 200, это позволяет отловить много аномальных случаев, которые были замечены в данной выборке. При использовании выборок малого количества, обучающей и тестовой выборкой обычно выбирают весь набор данных, т.к. такое малое количество зачастую является эксперементальными данными.

Благодарности

Исмагилов Амир Равилевич - КФУ, НОЦ "Моделирование ТРИЗ за помощь в вопросах обучения моделей с точки зрения геофизики. - `https://kpfu.ru/main?p_id=41656`

Судаков Владислав Анатольевич - КФУ, НОЦ "Моделирование ТРИЗ за возможность участвовать в команде и учиться применять знания анализа данных на практике. - `https://kpfu.ru/vladislav.sudakov`

Список литературы

[1] Балабанов, Ю. П. Геофизические методы изучения геопромысловых характеристик продуктивных пластов / Ю. П. Балабанов, И. П. Зинатуллина. // Казань. - 2016 - 47с. – http://dspace.kpfu.ru/xmlui/bitstream/handle/net/109860/metod_GIS-2-2.pdf?sequence=1&isAllowed=y

[2] Косков, В. Н. Геофизические исследования скважин и интерпретация данных ГИС"/ В. Н. Косков, Б. В. Косков // Пермь: Изд-во Пермского гос. технического ун-та. - 2007 - 315с. – https://search.rsl.ru/ru/record/01004240028

[3] Мартынова, В.Г. Геофизические исследования скважин: справочникмастера по промысловой геофизике / В.Г.Мартынова, Н.Е.Лазуткина, М.С.Хохлова // Москва: Инфра-Инженерия. - 2009 - 960с. – http://www.geokniga.org/bookfiles/geokniga-geofizicheskie-issledovaniya-skvazhin-spravochnik-mastera-po-promyslovoy-geofiz.pdf

[4] Эндрю Ын Machine Learning Yearning / Эндрю Ын // США. - 2018 - 118с. – https://www.deeplearning.ai/machine-learning-yearning/

[5] Стивен Марслэнд Machine Learning: An Algorithmic Perspective / Стивен Марслэнд // США. - 2015 - 452с. – https://seat.massey.ac.nz/personal/s.r.marsland/MLBook.html

[6] Официальная документация по языку Python - https://docs.python.org/3/

[7] Официальная документация по библиотеке SciPy – https://docs.scipy.org/doc/numpy/reference/

[8] Официальная документация по библиотеке NumPy – https://www.numpy.org/

[9] Официальная документация по библиотеке Scikit-learn – https://scikit-learn.org/stable/documentation.html

Приложения

https://github.com/matrixalex/Downloading - исходный код предобработки данных в Jupyter Notebook, обновление от 25.05.2019

https://github.com/matrixalex/Stratigraphy - исходный код стратиграфии в Jupyter Notebook, обновление от 06.06.2019

https://github.com/matrixalex/Lithology - исходный код литологии в Jupyter Notebook, обновление от 25.05.2019