МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет
имени К.И. Сатпаева

Институт автоматики и информационных технологий

Кафедра «Программная инженерия»

Көшкін Әсет Төлегенұлы

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

Образовательная программа 6B06102 - Computer Science

Алматы 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К.И.Сатпаева

Институт автоматики и информационных технологий

Кафедра «Программная инженерия»

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ПИ
канд. тех. наук, ассоц.профессор
_____ Ф.Н. Абдолдина
"27" 05 2025 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

На тему: «Сравнение эффективности различных методов машинного обучения для классификации геофизических данных»

Образовательная программа: 6B06102 Computer Science

Выполнил Көшкін Әсет Төлегенұлы

Рецензент
PhD доктор, ассоциированный профессор
_____ Темирбеков А. Н.
"27" 05 2025 г.

Научный руководитель
канд. техн. наук, ассоциированный профессор
_____ Абдолдина Ф. Н.
"23" 05 2025 г.

Алматы 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К.И.Сатпаева

Институт автоматики и информационных технологий

Кафедра «Программная инженерия»

УТВЕРЖДАЮ
Заведующий кафедрой ПИ канд. тех. наук, ассоц.профессор
_____Ф.Н. Абдолдина
«27» 05 2025 г.

ЗАДАНИЕ
на выполнение дипломного проекта

Обучающемуся: *Көшкін Әсет Төлегенұлы*
Тема: Сравнение эффективности различных методов машинного обучения для классификации геофизических данных
Утверждена приказом проректора по академической работе:
№ 1803-до

	от «25» ноября 2024 г.
Срок сдачи законченного проекта	*«26» 05 2025 г.*

Исходные данные к дипломному проекту:
А) Анализ предметной области;
Б) Анализ аналогичных методов;
В) Разработка технического задания;
Г) Сбор и предобработка набора данных;
Д) Разработка и тестирование моделей;
Перечень подлежащих разработке в дипломном проекте вопросов: (с точным указанием обязательных чертежей): *представлены* 24 *слайда презентации.*
Рекомендуемая основная литература: *из* 14 *наименований.*

Алматы 2025

ГРАФИК

подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	**Сроки представления научному руководителю и консультантам**	**Примечание**
1. Анализ предметной области, разработка технического задания	30.11.2024	Выполнено
2. Сбор и предобработка набора данных	25.12.2024	Выполнено
3. Разработка моделей	18.02.2025	Выполнено
4. Тестирование и оптимизация моделей	25.04.2025	Выполнено
5. Написание пояснительной записки к дипломному проекту	08.05.2025	Выполнено

Подписи

консультантов и нормоконтролера на законченную дипломный проект
с указанием относящихся к ним разделов проекта

Наименования разделов	**Консультанты, И.О.Ф. (уч. степень, звание)**	**Дата подписания**	**Подпись**	**Оценка**
Программное обеспечение	Шаханов Ә. Р. ст. преподаватель	23.05.2025		90
Нормоконтролер	Имаматдинова К. Ф. ст. преподаватель, магистр.	23.05.2025		

Научный руководитель_____Абдолдина. Ф.Н.

Задание принял к исполнению обучающийся Көшкін Әсет Төлегенұлы

Дата «26» 05 2025 г.

Алматы 2025

АҢДАТПА

Дипломдық жоба ұңғымалардың геофизикалық каротаж деректері негізінде литологиялық жыныс түрлерін автоматты түрде жіктеуге арналған машиналық оқыту жүйесін әзірлеуге арналған. Тақырыптың өзектілігі геологиялық деректердің үлкен көлемін дәл және жылдам интерпретациялау қажеттілігімен байланысты, бұл әсіресе геологиялық барлау және бұрғылау міндеттері үшін маңызды.

Жоба аясында машиналық оқытудың бірнеше алгоритмдері: Decision Tree, Random Forest, XGBoost, LightGBM, CatBoost және нейрондық желілер салыстырмалы түрде талданды. Деректерді алдын ала өңдеуден бастап модельдерді оқыту мен тексеруге дейінгі толық өңдеу циклі жүзеге асырылды. Шешімдердің тиімділігі келесі метрикалар арқылы бағаланды: Accuracy, F1-score, Cohen’s Kappa, MCC, оқыту уақыты және сыныптар арасындағы қателіктердің маңыздылығын ескеретін арнайы Penalty Score метрикасы.

Әзірленген алгоритмдер жіктеудің жоғары дәлдігін және модельдердің тұрақтылығын көрсетті, бұл әдістің геофизикалық деректерді автоматтандырылған түрде талдау міндеттерінде іс жүзінде қолдануға болатындығын дәлелдейді. Жоба Python бағдарламалау тілі мен Scikit-learn, XGBoost, LightGBM, CatBoost, Keras және TensorFlow сияқты машиналық оқыту кітапханаларын қолдана отырып жүзеге асырылды.

АННОТАЦИЯ

Дипломный проект посвящён разработке системы машинного обучения для автоматической классификации литологических типов пород на основе геофизических данных каротажа скважин. Актуальность темы обусловлена необходимостью повышения точности и скорости интерпретации больших объёмов геологических данных, что особенно важно для задач геологоразведки и бурения.

В рамках работы проведён сравнительный анализ алгоритмов машинного обучения: Decision Tree, Random Forest, XGBoost, LightGBM, CatBoost и нейронных сетей. Реализован полный цикл обработки данных от предобработки до обучения и валидации моделей. Эффективность решений оценивалась с использованием метрик Accuracy, F1-score, Cohen's Kappa, MCC, времени обучения и специализированной метрики Penalty Score, учитывающую серьёзность ошибок между классами.

Разработанная алгоритмы продемонстрировала высокую точность классификации и устойчивость моделей, что подтверждает практическую применимость метода в задачах автоматизированного анализа геофизических данных. Проект выполнен с использованием языка программирования Python и библиотек машинного обучения, таких как Scikit-learn, XGBoost, LightGBM, CatBoost, Keras и TensorFlow.

ABSTRACT

The diploma project is dedicated to the development of a machine learning system for the automatic classification of lithological rock types based on geophysical well log data. The relevance of the topic is due to the need to improve the accuracy and speed of interpreting large volumes of geological data, which is especially important for exploration and drilling tasks.

Within the scope of the project, a comparative analysis of machine learning algorithms was conducted: Decision Tree, Random Forest, XGBoost, LightGBM, CatBoost, and neural networks. A complete data processing pipeline was implemented from preprocessing to model training and validation. The effectiveness of the solutions was evaluated using metrics such as Accuracy, F1-score, Cohen's Kappa, MCC, training time, and the specialized Penalty Score, which accounts for the severity of misclassifications between classes.

The developed algorithms demonstrated high classification accuracy and model robustness, confirming the practical applicability of the method for automated analysis of geophysical data. The project was implemented using the Python programming language and machine learning libraries such as Scikit-learn, XGBoost, LightGBM, CatBoost, Keras, and TensorFlow.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современная нефтегазовая отрасль Республики Казахстан сталкивается с необходимостью более точной и оперативной интерпретации геофизических данных, получаемых в процессе исследований скважин. Одним из ключевых этапов геологоразведочных работ является классификация литологических типов пород по данным геофизических исследований скважин (ГИС), от точности которой напрямую зависят построение геологических моделей, планирование буровых операций и эффективность добычи. Однако традиционные методы интерпретации, основанные на линейных зависимостях и экспертной оценке, нередко оказываются недостаточно точными, особенно в условиях сложных геологических разрезов.

Актуальность и практическая значимость разработки алгоритмов классификации литологических данных с использованием методов машинного обучения обусловлены необходимостью повышения уровня автоматизации, снижения зависимости от человеческого фактора и увеличения точности при обработке больших объёмов каротажной информации. Алгоритмы машинного обучения позволяют выявлять скрытые закономерности, учитывать нелинейные зависимости между признаками и эффективно работать с разнородными и неполными данными, что делает их перспективным инструментом в задачах интерпретации ГИС.

Цифровизация нефтегазовой отрасли имеет стратегическое значение и поддерживается на государственном уровне. В Послании народу Казахстана от 1 сентября 2023 года Президент Республики Казахстан Касым-Жомарт Токаев подчеркнул необходимость модернизации и цифровой трансформации ключевых отраслей экономики, включая топливно-энергетический комплекс. Внедрение интеллектуальных систем анализа данных, включая решения на основе методов машинного обучения, рассматривается как важнейшее направление устойчивого развития и повышения эффективности использования природных ресурсов [1].

Целью настоящей дипломной работы является сравнительный анализ различных алгоритмов машинного обучения для классификации литологических типов пород по данным ГИС, с целью выявления наиболее эффективных подходов.

Объектом исследования являются каротажные данные, полученные в результате геофизических исследований скважин, отражающие физико-геологические характеристики пород.

Предметом исследования являются алгоритмы машинного обучения, применяемые для автоматической классификации литологических типов пород на основе данных ГИС.

В соответствии с поставленной целью в работе решаются следующие задачи:

- провести анализ существующих методов классификации геофизических данных с использованием машинного обучения;
- выполнить подготовку и предобработку каротажных данных;
- реализовать и обучить модели на основе алгоритмов: Decision Tree, Random Forest, XGBoost, LightGBM, CatBoost, MLP и Keras Neural Network;
- оценить эффективность моделей с использованием стандартных и специализированных метрик;
- провести сравнительный анализ моделей и определить наиболее эффективные подходы.

1 Исследовательско-технологическая часть

1.1 Геофизические исследования скважин (ГИС)

Геофизические исследования скважин (ГИС) представляют собой совокупность методов, направленных на изучение геологического строения и физических свойств горных пород путём измерений внутри ствола скважины. Они играют важную роль в разведке и разработке месторождений полезных ископаемых. ГИС позволяют регистрировать изменения физических параметров пород с высокой вертикальной разрешающей способностью, что обеспечивает ценную информацию для построения геологических и гидродинамических моделей [2, 3].

На рисунке 1.1 представлена последовательность этапов проведения геофизических исследований в стволе скважины. Процесс ГИС начинается с предварительной подготовки: очистки скважины от бурового раствора, установки обсадной колонны, калибровки инструментов и обеспечения условий для надёжной регистрации данных. В ходе измерений применяются приборы, фиксирующие физико-технические свойства горных пород, такие как удельное электрическое сопротивление, плотность, радиоактивность и акустические характеристики. Полученные данные записываются в виде каротажных диаграмм, которые служат основой для последующего анализа и интерпретации [2].

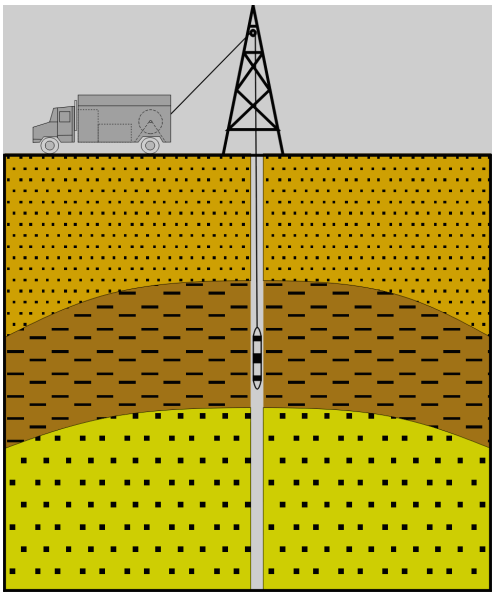


Рисунок 1.1 - Процесс исследования скважины

На рисунке 1.2 представлено основное оборудование, применяемое при проведении геофизических исследований. Стандартный состав аппаратурного комплекса включает каротажные станции, кабельные лебёдки, телеметрические

системы, зондовые устройства и блоки первичной обработки сигналов. Современное оборудование позволяет регистрировать множественные параметры одновременно, что существенно повышает информативность и надёжность получаемых результатов [3].

Рисунок - 1.2 - Оборудования для исследования

Пример каротажных диаграмм и результатов их литологической интерпретации приведён на рисунке 1.3. Интерпретация геофизических данных основывается на анализе геофизических кривых, отражающих изменения физических свойств пород по глубине скважины. Это позволяет выделять литологические границы, определять продуктивные пласты и уточнять структуру геологического строения [2, 3].

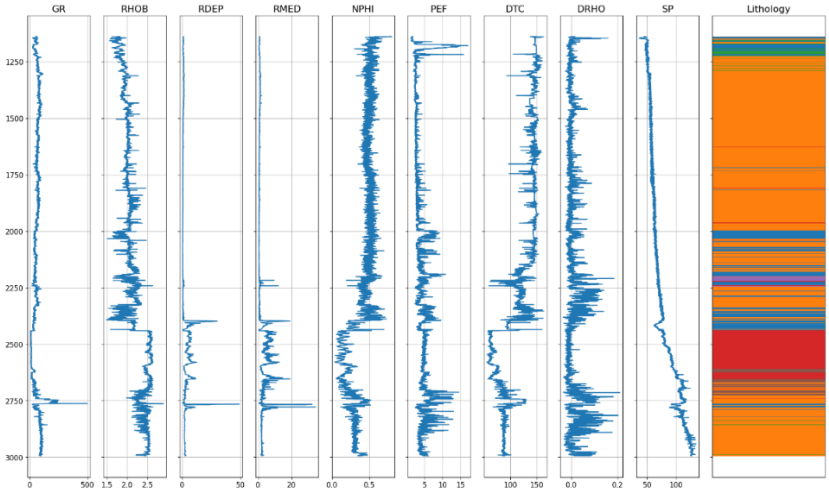


Рисунок 1.3 - Каротажная диаграмма и литологическая интерпретация

Методы ГИС классифицируются в зависимости от используемых физических принципов на следующие группы:

- Электрические методы измерение сопротивления, вызванной поляризации и других электрофизических характеристик. Применяются для оценки насыщенности и глинистости пород.
- Акустические методы анализ скорости распространения упругих волн. Позволяют оценить плотность, пористость и трещиноватость.
- Ядерные методы гамма-каротаж, нейтронные и гамма-гамма измерения. Используются для изучения радиоактивности и минералогического состава пород.
- Термические методы регистрация температурных градиентов и теплопроводности пород.
- Механические методы оценка прочностных и деформационных характеристик горных пород.
- Электромагнитные методы определение проводимости и характеристик насыщенности пластов.

Эффективность интерпретации существенно возрастает при комплексном использовании различных методов. Это позволяет получить многопараметрическую информацию о геологическом разрезе и значительно снизить уровень неопределённости при построении геолого-геофизических моделей [2, 3].

1.2 Обзор методов классификации литологических данных

Классификация литологических типов пород по данным геофизических исследований скважин (ГИС) является одной из ключевых задач в нефтегазовой геологии. Каротажные данные, содержащие сведения о физических свойствах пород (гамма-активность, пористость, акустическая скорость и др.), позволяют оценивать продуктивность пластов и строить достоверные геологические модели [2, 3].

Традиционные методы интерпретации, основанные на эмпирических зависимостях между параметрами и литологией, оказываются недостаточно точными при наличии больших объёмов данных и высокой геологической неоднородности. В таких условиях всё более востребованными становятся методы машинного обучения (ML), которые позволяют автоматизировать процесс классификации, повысить точность анализа и минимизировать влияние субъективного фактора [4].

Примером успешного применения алгоритмов машинного обучения для классификации пород является исследование, посвящённое породам викуловской свиты. В нём использовались алгоритмы CatBoost, Random Forest и многослойный персептрон (MLP), обеспечившие точность более 85% [5].

Ансамблевые алгоритмы, такие как Random Forest (RFC), XGBoost (XGB) и LightGBM (LGBM), также продемонстрировали высокие значения Accuracy (от 84.4% до 86.8%) и F1-метрик. Это подтверждает их устойчивость к шумам и

пригодность для обработки реальных геофизических данных [6]. В таблице 1.1 показано сравнения моделей.

Таблица 1.1 - Сравнение моделей классификации по метрикам

Classifier	Accuracy	F1 Macro	F1 Micro	Duration (s)
LGBM	0.868	0.813	0.868	5.757
RFC	0.844	0.766	0.844	94.193
XGB	0.859	0.803	0.859	161.549
kNN	0.666	0.588	0.666	134.631
DT	0.721	0.645	0.721	18.003
MLP	0.809	0.759	0.809	54.788
NB	0.479	0.454	0.479	0.98
Linear SVM	0.75	0.664	0.75	3322.33
RBF SVM	0.799	0.73	0.799	1156.59

Современные исследования демонстрируют разнообразие алгоритмов машинного обучения, применяемых для задач литологической классификации. В одной из работ предложен открытый набор данных и бенчмарк на основе 3D-геологической модели блока F3 в Северном море [7]. Визуализация структуры приведена на рисунке 1.4.

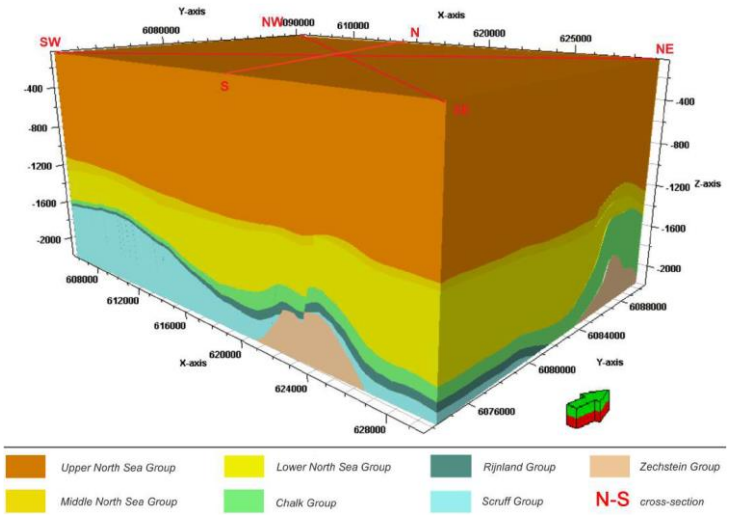


Рисунок 1.4 - 3D-модель геологического строения блока F3

В модели выделено семь литостратиграфических групп (включая отложения кайнозоя, мезозоя и перми), а также сложные геологические структуры соляные диапиры, разломы и пр. Это делает блок F3 удобным полигоном для тестирования алгоритмов классификации.

Глубинные нейросетевые архитектуры, в частности сверточные нейронные сети (CNN), также находят применение в интерпретации ГИС. В одном из исследований CNN достигала точности до 87% на тестовых данных [8].

Сравнительные результаты различных моделей приведены на рисунке 1.5.

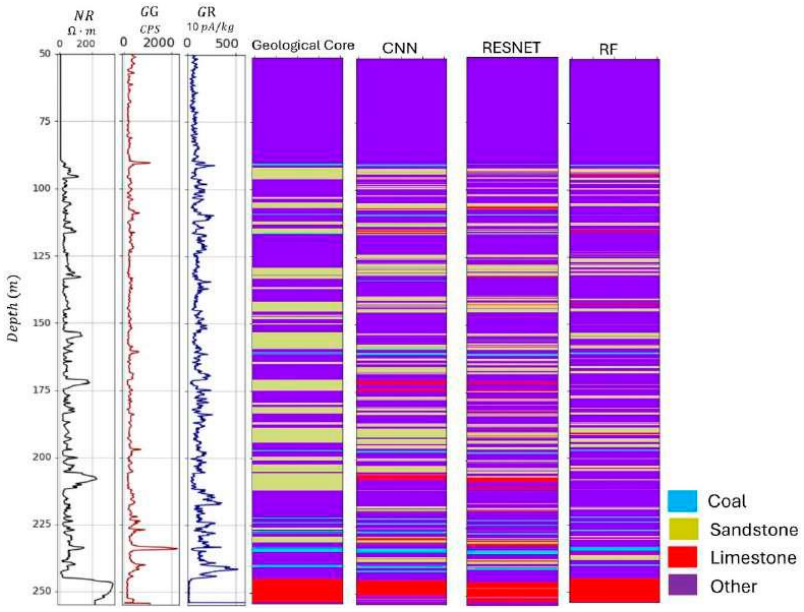


Рисунок 1.5 - Визуализация результатов классификации литологий моделями CNN, RESNET и Random Forest по данным каротажа

В другом исследовании, посвящённом классификации пород Баикокуаньской свиты, наилучшие результаты продемонстрировал алгоритм XGBoost: точность достигала 90% даже при наличии пропусков и шумов [9].

Инновационные подходы включают гибридные архитектуры, такие как GrowNet с использованием Deep-Insight. В одной из таких моделей входные признаки преобразуются в двумерные изображения, повышающие интерпретируемость и точность при ограниченном объёме обучающей выборки [10]. Пример визуализации приведён на рисунке 1.6.

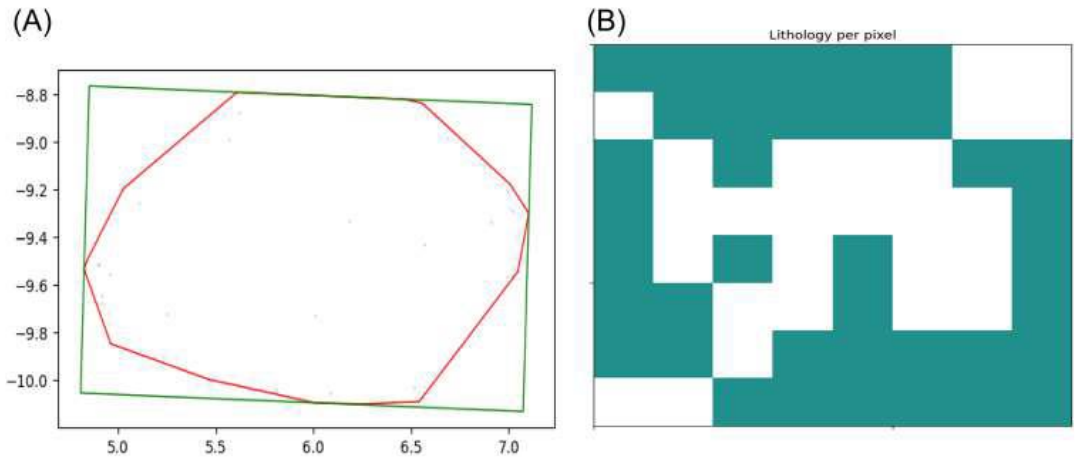


Рисунок 1.6 - Преобразование признаков в двумерную карту признаков

Многие авторы подчёркивают важность этапа предварительной обработки

данных. Нормализация, восстановление пропусков, отбор признаков и балансировка классов существенно повышают эффективность моделей. Кроме того, успешность алгоритмов зависит от адаптации под региональные геологические особенности и интерпретируемости результатов [4, 11].

Таким образом, методы машинного обучения от классических до глубинных и гибридных демонстрируют высокую результативность в автоматизированной классификации литологических типов. Их внедрение способствует повышению точности интерпретации ГИС, ускорению обработки данных и усилению поддержки принятия решений в геологоразведке.

2 Проектно-экспериментальная часть

2.1 Анализ набора данных ГИС

FORCE (Forum for Reservoir Characterization and Exploration) это инициативная платформа, действующая под Norwegian Offshore Directorate и объединяющая более 40 норвежских и международных нефтегазовых компаний и исследовательских организаций [12]. Цель инициативы стимулировать обмен знаниями, разработку новых технологий и внедрение цифровых решений в разведку и разработку углеводородных ресурсов. В 2020 году была организована открытая индустриальная инициатива соревнование FORCE 2020 Machine Learning Contest Lithology Prediction, целью которого стало продвижение методов машинного обучения в области геонаук. Соревнование привлекло множество участников со всего мира, а его результаты, методики и лучшие решения были опубликованы в открытом доступе [13, 14].

В рамках настоящей работы для решения задачи классификации литологических типов пород использовался открытый набор данных из упомянутого соревнования. Данный набор данных представляет собой результат геофизических исследований скважин [13]. На рисунке 2.1 показано расположение скважин, включённых в набор данных FORCE 2020.

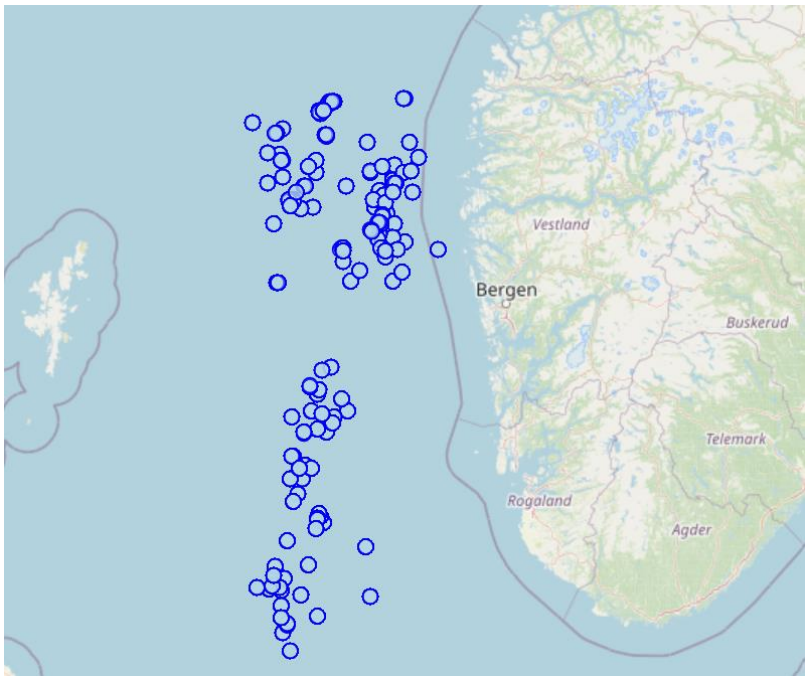


Рисунок 2.1 - Координаты скважин

Все скважины находятся в акватории Северного моря у западного побережья Норвегии. Каждая точка на карте соответствует координатам одной из скважин, по которым были собраны геофизические данные. Распределение

охватывает несколько зон шельфа, что обеспечивает разнообразие геологических условий в выборке.

Общий объем обучающей выборки составляет 1 170 511 строк, каждая из которых соответствует определённой глубине. Данные собраны по 98 скважинам, каждая из которых ассоциирована с определённой геологической формацией и группой.

В дополнение к обучающему набору в рамках конкурса были предоставлены два тестовых набора данных: leaderboard_test, включающий 136 786 записей, и hidden_test, содержащий 122 397 записей и 10 уникальных скважин. Первый используется для промежуточной валидации моделей, второй для финального тестирования, обеспечивая независимую оценку обобщающей способности алгоритмов. Эти данные позволяют проверять модели на новых, ранее не встречавшихся скважинах, оценивать устойчивость и адаптивность к геологической изменчивости, а также эффективность моделей с применением штрафной матрицы ошибок. Такой формат соответствует современным стандартам машинного обучения и обеспечивает объективность сравнения подходов.

В таблице 2.1 приведены 29 признаков обучающего набора данных, отражающих физико-геологические характеристики, пространственные координаты и категориальные метки.

Таблица 2.1 – Основные признаки набора данных

№	Признак	Описание
1	WELL	Идентификатор скважины
2	DEPTH_MD	Глубина измерения (в метрах)
3	X_LOC	Координата X местоположения
4	Y_LOC	Координата Y местоположения
5	Z_LOC	Координата Z (высота над уровнем моря)
6	GROUP	Геологическая группа
7	FORMATION	Геологическая формация
8	CALI	Диаметр скважины
9	RSHA	Глубинная резистивность (альтернативный канал)
10	RMED	Средняя резистивность
11	RDEP	Глубинная резистивность
12	RHOB	Объёмная плотность породы
13	GR	Гамма-каротаж (естественная радиоактивность)
14	SGR	Спектральный гамма-каротаж
15	NPHI	Нейтронная пористость

Продолжение таблицы 2.1

№	Признак	Описание
16	PEF	Фотоэлектрический коэффициент
17	DTC	Скорость продольных акустических волн (Compressional slowness)
18	SP	Спонтанная поляризация
19	BS	Диаметр бурового инструмента (bit size)
20	ROP	Скорость проходки бурения
21	DTS	Скорость поперечных акустических волн (Shear slowness)
22	DCAL	Диаметр скважины по другому каналу
23	DRHO	Разность плотности (градиент)
24	MUDWEIGHT	Плотность бурового раствора
25	RMIC	Микрорезистивность
26	ROPA	Скорость проходки (альтернатива ROP)
27	RXO	Резистивность промытой зоны
28	FORCE_2020_ LITHOFACIES_LITHOLOGY	Целевая переменная код литологического класса
29	FORCE_2020_ LITHOFACIES_CONFIDENCE	Уверенность в правильности метки литологии

Целевая переменная числовой признак FORCE_2020_LITHOFACIES_LITHOLOGY, каждому значению которого соответствует конкретный тип породы. В таблице 2.2 приведены 12 литологических классов, различающихся по физико-геологическим свойствам и распространённости.

Таблица 2.2 - Литологические классы

Код литологии	Литология (англ.)	Перевод
30000	Sandstone	Песчаник
65030	Sandstone/Shale	Песчаник/Сланец
65000	Shale	Сланец
80000	Marl	Мергель
74000	Dolomite	Доломит
70000	Limestone	Известняк
70032	Chalk	Мел
88000	Halite	Галит
86000	Anhydrite	Ангидрит

Продолжение таблицы 2.2

Код литологии	Литология (англ.)	Перевод
99000	Tuff	Туф
90000	Coal	Уголь
93000	Basement	Фундамент

Классы охватывают как осадочные породы, так и специфические типы, такие как галит и кристаллический фундамент. Присутствие смешанных классов, например, Sandstone/Shale, указывает на сложные литологические переходы и повышает сложность задачи классификации.

Набор данных является мультиклассовым и характеризуется выраженным дисбалансом. На рисунке 2.2 показано распределение литологических классов: более 61% записей относятся к классу Shale, далее следуют Sandstone 14.43% и Sandstone/Shale 12.85%.

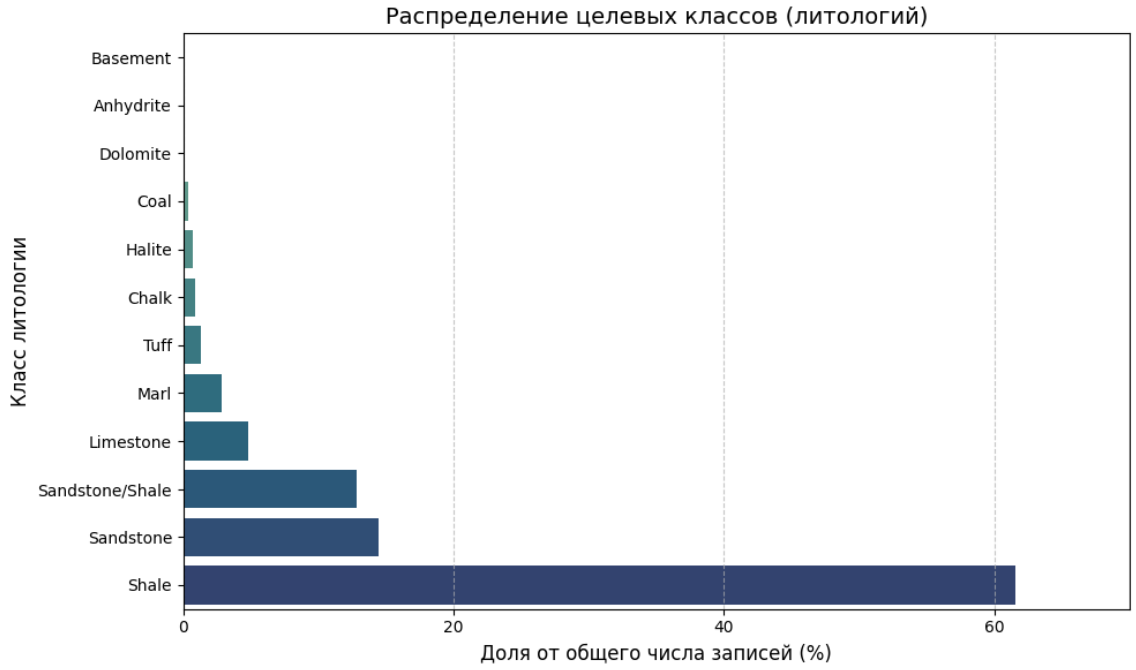


Рисунок 2.2 - Распределение целевых переменных

Остальные классы представлены в значительно меньшем объёме: Limestone менее 5%, Marl около 2.85%, Halite, Coal, Dolomite и Chalk менее 1%. Наименее представлены Basement и Anhydrite. Такой дисбаланс может привести к смещённости моделей в сторону доминирующих классов. Это требует применения специальных подходов балансировки классов, взвешенных функций потерь, а также метрик, чувствительных к редким классам.

На рисунке 2.3 представлена доля пропущенных значений по каждому признаку. Наиболее полными являются GR, RDEP и RMED с долей пропусков менее 5%, в то время как GR вообще не содержит пропусков, что делает его

надёжным и информативным признаком.

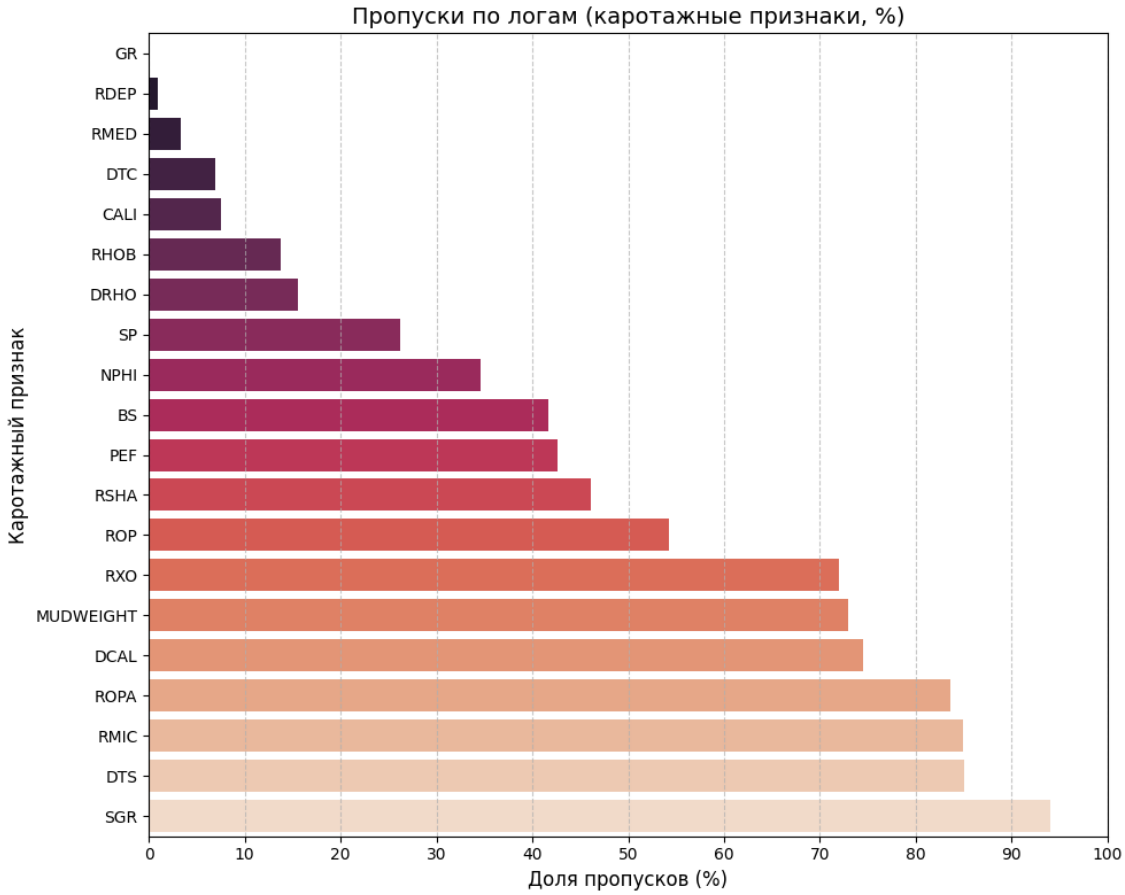


Рисунок 2.3 - Пропуски каротажных признаков

Наибольшее количество пропусков наблюдается у SGR, DTS, RMIC и ROPA. Высокая пропущенность требует либо удаления таких признаков, либо восстановления их значений по другим каналам, либо применения моделей, устойчивых к неполным данным. Также высокую долю пропусков имеют DCAL, MUDWEIGHT, RXO, RSHA и PEF.

На рисунке 2.4 представлена тепловая карта корреляционной матрицы между числовыми признаками. Яркие цвета соответствуют сильной положительной или отрицательной корреляции, а светлые слабой связи.

Наиболее выраженные связи:

- RHOB и DTC: отрицательная корреляция -0.83 отражает связь между плотностью породы и скоростью продольных волн.
- NPHI и DTC: положительная корреляция 0.79.
- CALI и DTC: умеренная положительная корреляция 0.59.
- CALI и BS: высокая корреляция 0.90.
- BS и DTC: положительная связь 0.54.

Некоторые признаки, например RMIC, ROPA, RXO, демонстрируют слабую корреляцию, вероятно, из-за высокой доли пропущенных значений или специфики измерения.

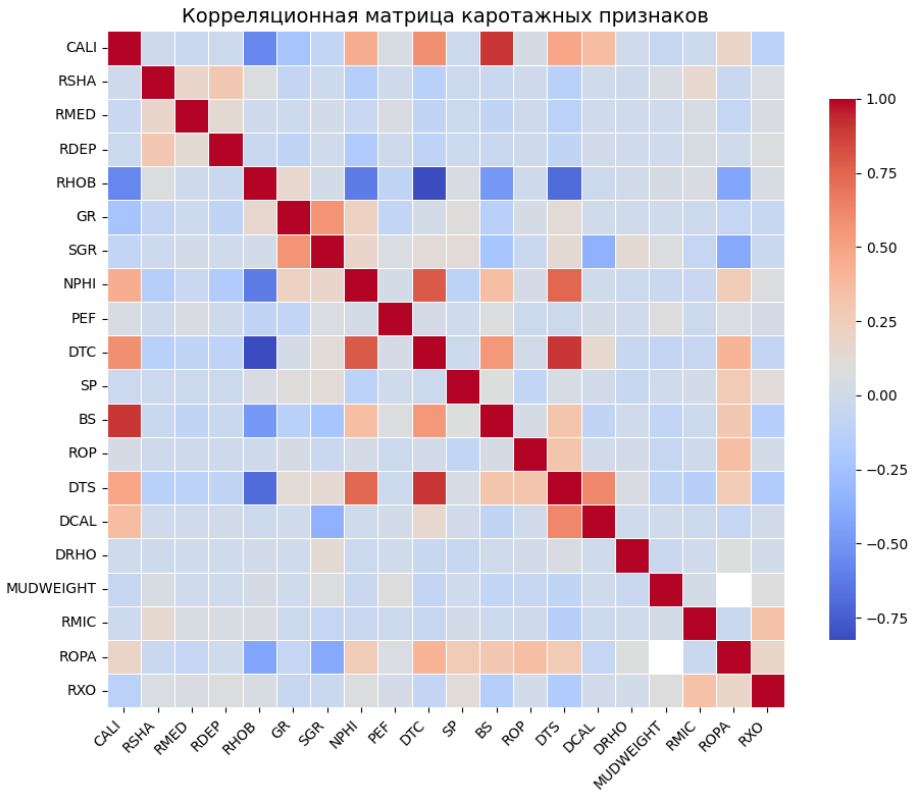


Рисунок 2.4 - Корреляционная матрица признаков

На рисунке 2.5 представлены диаграммы с выбросами, определёнными по межквартильному размаху.

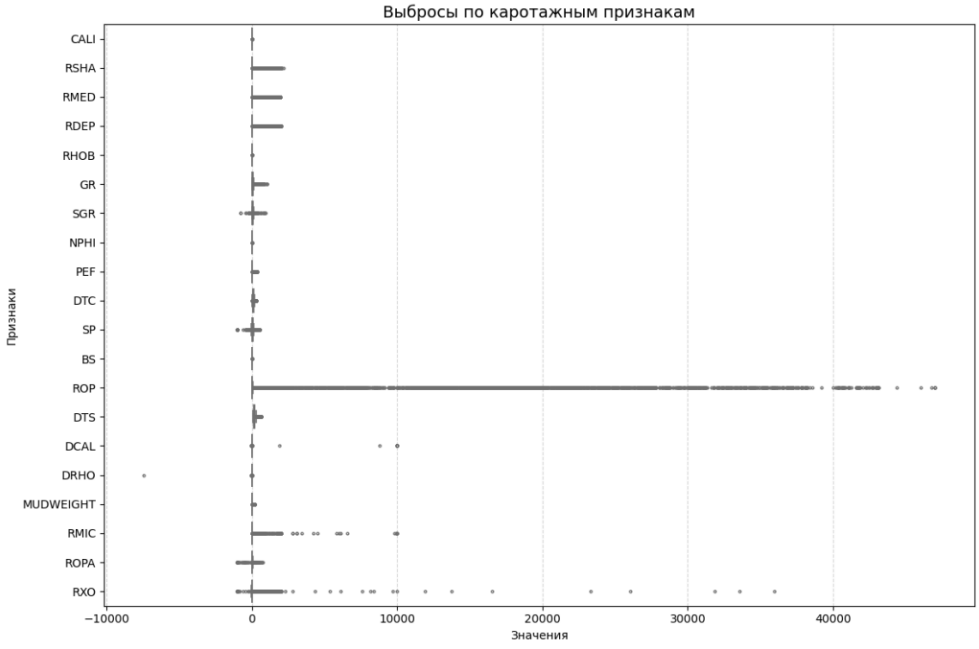


Рисунок 2.5 - Выбросы по каротажным признакам

Точки за пределами усов диаграмм отражают экстремальные значения. Признаки CALI, RHOB, GR, SGR, NPHI, DTC, BS и MUDWEIGHT демонстрируют стабильное распределение и малую долю выбросов. Они могут быть использованы без строгой фильтрации. Умеренное количество выбросов зафиксировано у ROP, DTS, PEF, ROPA и DCAL для них может потребоваться нормализация или логарифмическое преобразование. Выраженная зашумлённость наблюдается у RSHA, RMED, RDEP, SP, DRHO, RMIC и особенно RXO. Это требует тщательной диагностики и фильтрации перед обучением моделей.

Таким образом, набор данных FORCE 2020 представляет собой многоаспектный и реалистичный пример геофизических данных, подходящий для разработки и тестирования алгоритмов машинного обучения в задачах литологической классификации. Его широкое применение в научных исследованиях подтверждает его ценность как в образовательных, так и в прикладных целях.

2.2 Предобработка данных ГИС

Предобработка геофизических данных скважин (ГИС) является важным этапом перед обучением моделей машинного обучения. На этом этапе происходит загрузка и преобразование исходных данных, включая обучающий и тестовые наборы train.csv, leaderboard_test_features.csv, hidden_test.csv. После загрузки осуществляется преобразование кодов литологий в числовые индексы, что обеспечивает совместимость с алгоритмами классификации. Также формируются словари соответствий между кодами и индексами, сохраняются читаемые метки литологий.

Для построения моделей используются наиболее информативные каротажные признаки, отражающие физические свойства пород: гамма-каротаж (GR), плотность (RHOB), резистивности (RDEP и RMED), нейтронная пористость (NPHI), фотоэлектрический эффект (PEF), скорость акустических волн (DTC), градиент плотности (DRHO) и спонтанная поляризация (SP).

Пропущенные значения обрабатываются с помощью функции interpolate_fill, которая сначала выполняет линейную интерполяцию значений по глубине внутри каждой скважины, восстанавливая непрерывность каротажных кривых. Если после интерполяции остаются пустые значения, они заполняются глобальной медианой признака. Такой подход позволяет сохранить структуру данных без искажения распределения признаков.

Для устранения выбросов применяется метод обрезки по перцентилям. Значения ниже 0.5% и выше 99.5% заменяются на соответствующие пороговые уровни. Это снижает влияние аномальных точек, при этом основное распределение признаков сохраняется.

Так как различные признаки могут иметь разные масштабы, применяется стандартизация методом StandardScaler. Все признаки преобразуются к нулевому среднему и единичному стандартному отклонению, что необходимо для корректной работы многих моделей, особенно нейросетевых.

Целевая переменная FORCE_2020_LITHOFACIES_LITHOLOGY преобразуется в числовые индексы, что обеспечивает совместимость с алгоритмами классификации. Это особенно важно для моделей, таких как Keras Neural Network, где требуется формат one-hot или целочисленные индексы.

Для борьбы с дисбалансом классов применяется автоматический расчёт весов классов на основе их частот. Полученные веса передаются в модели, поддерживающие параметр class_weight, что позволяет усилить вклад редких классов и повысить устойчивость модели к перекосу в сторону доминирующих литологий.

Все описанные процедуры повторяются для тестовых наборов leaderboard_test и hidden_test, что обеспечивает идентичную предобработку данных на всех этапах от обучения до валидации и финального инференса.

2.3 Описание моделей для классификации

Для решения задачи классификации литологических типов пород по данным геофизических исследований скважин были использованы как классические алгоритмы машинного обучения, так и нейросетевые архитектуры. Выбор моделей обусловлен их популярностью в научной и прикладной практике, а также разнообразием подходов к обучению, интерпретации и обработке данных. Это позволяет провести всестороннее сравнение алгоритмов в контексте геофизической задачи.

На рисунке 2.6 представлены основные типы моделей на основе деревьев решений: одиночное дерево решений, градиентный бустинг и случайный лес.

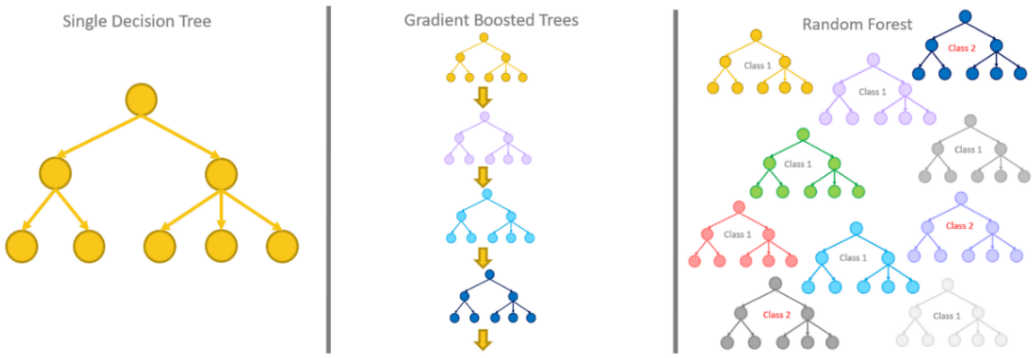


Рисунок 2.6 - Сравнение моделей на основе деревьев решений: одиночное дерево, градиентный бустинг и случайный лес

Decision Tree (дерево решений) - простая и интерпретируемая модель, основанная на последовательных разбиениях пространства признаков. Несмотря на склонность к переобучению, полезна на этапах начального анализа и в задачах, требующих объяснимости решений.

Random Forest (случайный лес) - ансамблевый метод, объединяющий множество деревьев, обученных на различных подвыборках данных. Усреднение предсказаний снижает переобучение и повышает устойчивость. Метод масштабируем, стабилен и хорошо работает с несбалансированными выборками при использовании взвешивания классов.

XGBoost (Extreme Gradient Boosting) - мощный бустинговый алгоритм с регуляризацией, обработкой пропущенных значений и поддержкой параллельных вычислений. Обеспечивает высокую точность за счёт последовательного минимизирования ошибки.

LightGBM (Light Gradient Boosting Machine) - высокопроизводительный бустинг, оптимизированный по скорости и памяти. Применяет leaf-wise стратегию роста деревьев, что позволяет строить более глубокие и точные модели на больших объёмах данных.

CatBoost - алгоритм от компании Яндекс, ориентированный на работу с категориальными признаками. Обеспечивает высокую точность без сложной настройки, устойчив к переобучению и автоматически учитывает дисбаланс классов.

Нейросетевые архитектуры представлены на рисунке 2.7, где приведена структура полносвязной сети, иллюстрирующая архитектуру многослойного перцептрона.

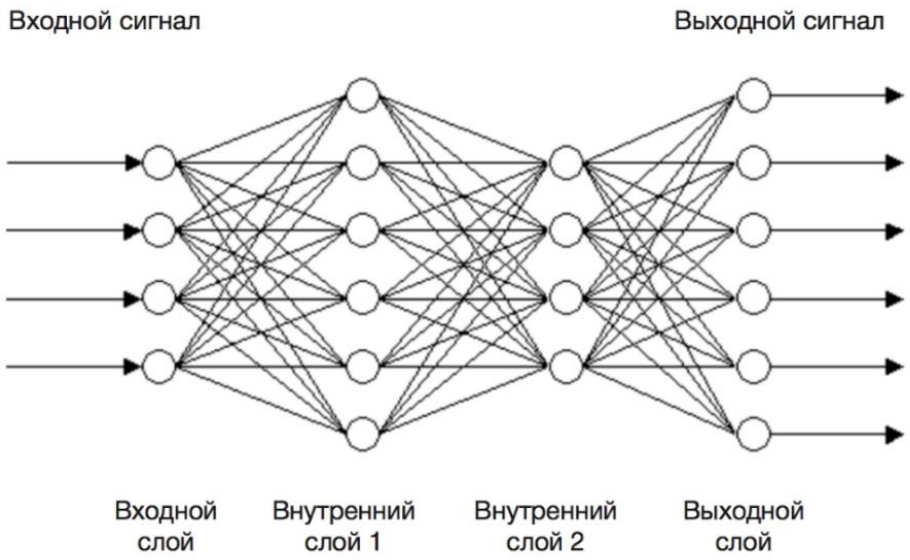


Рисунок 2.7 - Архитектура полносвязной нейронной сети

MLP Classifier (многослойный перцептрон) - классическая полносвязная нейросеть, состоящая из нескольких скрытых слоёв с функциями активации.

Обеспечивает распознавание сложных нелинейных зависимостей, однако чувствительна к масштабу данных и требует тщательной настройки гиперпараметров.

Keras Neural Network - глубокая нейронная сеть, реализованная с использованием библиотеки Keras на базе TensorFlow. Применяет последовательную архитектуру, использует активации ReLU в скрытых слоях и softmax на выходе. Обучение выполняется с помощью оптимизатора Adam и функции потерь categorical_crossentropy. Подходит для многоклассовых задач и демонстрирует высокую точность при наличии большого объёма данных.

Каждая модель имеет свои преимущества и ограничения. Классические модели обеспечивают высокую интерпретируемость и стабильность при умеренных вычислительных затратах. Бустинговые методы демонстрируют отличные показатели точности и гибкости, особенно при наличии большого количества признаков. Нейросетевые подходы обеспечивают высокую обобщающую способность, но требуют значительных вычислительных ресурсов и аккуратной настройки архитектуры.

Таким образом, использование широкого набора моделей позволяет провести комплексный сравнительный анализ и выбрать оптимальные решения с учётом специфики геофизических данных и ограничений вычислительной среды.

2.4 Используемые библиотеки

В процессе реализации алгоритмов машинного обучения для классификации литологических типов пород по данным геофизических исследований скважин был использован язык программирования Python, благодаря своей гибкости, богатому набору библиотек и широкому применению в задачах анализа данных и построения интеллектуальных систем.

В рамках проекта применялись следующие основные библиотеки:

- Pandas - библиотека для работы с табличными данными. Применялась для загрузки, фильтрации, агрегации и объединения набора данных, а также для анализа распределения признаков и меток классов.
- NumPy - базовая библиотека для численных вычислений и работы с массивами. Использовалась при выполнении векторных и матричных операций, преобразованиях признаков и подготовке входных данных для моделей.
- Scikit-learn - универсальный инструмент для машинного обучения. С её помощью реализованы модели Decision Tree, Random Forest и MLP Classifier, масштабирование признаков, вычисление классических метрик, формирование confusion matrix, а также автоматический расчёт весов классов для борьбы с дисбалансом.
- XGBoost - эффективная библиотека градиентного бустинга, обеспечивающая высокую точность при работе с пропущенными и шумными

данными. Применялась для обучения модели XGBoost с настройкой регуляризации и параметров скорости обучения.

- LightGBM - высокопроизводительная библиотека градиентного бустинга, разработанная компанией Microsoft. Использовалась для построения модели LightGBM, отличающейся высокой скоростью работы и способностью обрабатывать большие объёмы данных.

- CatBoost - библиотека компании Яндекс, предназначенная для обработки категориальных данных. Применялась для обучения модели с автоматическим учётом дисбаланса классов и минимальной необходимостью настройки параметров.

- TensorFlow и Keras - библиотеки для глубокого обучения. Использовались для реализации модели Keras Neural Network: многослойной нейросети с функциями активации ReLU и softmax, оптимизатором Adam и функцией потерь categorical_crossentropy. Архитектура включала скрытые слои и механизм регуляризации Dropout.

- Matplotlib и Seaborn - библиотеки визуализации. Применялись для построения диаграмм распределений, корреляционных матриц, графиков выбросов, confusion matrix и сравнительных графиков эффективности моделей.

- SHAP (SHapley Additive exPlanations) - библиотека для интерпретации результатов моделей. Использовалась для анализа важности признаков в нейросетевых моделях и построения объяснений на основе SHAP-значений.

Использование данных библиотек обеспечило воспроизводимость экспериментов, высокую скорость разработки и возможность масштабирования проекта. Такой инструментарий позволил реализовать полный аналитический цикл от подготовки данных и обучения моделей до интерпретации и визуализации результатов.

2.5 Реализация моделей машинного обучения

Для решения задачи классификации литологических типов по данным геофизических исследований скважин были использованы как классические алгоритмы машинного обучения, так и нейросетевые архитектуры. Среди них: Decision Tree, Random Forest, XGBoost, LightGBM, CatBoost, MLP Classifier и Keras Neural Network. Выбор моделей обусловлен разнообразием их подходов к обучению, интерпретируемостью, устойчивостью к шуму и способностью эффективно работать в условиях выраженного дисбаланса между литологическими классами.

Охвачены методы от простых деревьев решений до современных ансамблевых и глубоких нейросетевых алгоритмов, что позволило провести всестороннее сравнение их эффективности в контексте многоклассовой геофизической задачи. Все модели были реализованы с использованием популярных библиотек Python: scikit-learn, XGBoost, LightGBM, CatBoost и Keras

на базе TensorFlow.

Каждая модель инициализировалась вручную подобранными гиперпараметрами. Для деревьев решений и ансамблей задавались параметры глубины, количество деревьев и стратегия разбиения. В бустинговых моделях настраивались коэффициенты регуляризации, скорость обучения и параметры подвыборки признаков и объектов. Для моделей, чувствительных к дисбалансу классов, применялись параметры class_weight или auto_class_weights.

Для учёта выраженного дисбаланса был реализован механизм автоматического расчёта весов классов с использованием функции compute_class_weight. Это позволяло повысить значимость редких литологических типов при обучении. Рассчитанные веса передавались в соответствующие алгоритмы. Decision Tree, Random Forest, MLP Classifier и XGBoost принимали словарь явно, а LightGBM и CatBoost использовали встроенные механизмы балансировки.

Для нейросетевых моделей обязательно выполнялась стандартизация признаков с помощью StandardScaler, что приводило значения к нулевому среднему и единичному стандартному отклонению. Это улучшало сходимость градиентных методов и обеспечивало устойчивость обучения.

Обучение всех моделей было реализовано в едином программном цикле, что обеспечивало идентичную предобработку и структуру входных данных. Для каждой модели выполнялись:

- обучение;
- предсказание на валидационной выборке;
- вычисление набора метрик: Accuracy, F1-мера, Precision, Recall, коэффициенты Маттьюса и Каппа;
- расчёт специализированной метрики Penalty Score, отражающей тяжесть ошибок между литологиями;
- построение матриц ошибок в абсолютной и нормализованной форме;
- фиксация времени обучения.

Модель Keras Neural Network обучалась отдельно с использованием TensorFlow. Её архитектура включала три скрытых слоя с функциями активации ReLU и Dropout для регуляризации. Выходной слой применял softmax-активацию. В качестве функции потерь использовалась categorical_crossentropy, а оптимизация производилась с помощью метода Adam. Целевая переменная предварительно преобразовывалась в формат one-hot, что является стандартной практикой в многоклассовых задачах классификации.

Результаты всех моделей сравнивались по единым метрикам, что обеспечивало объективную и реалистичную оценку качества в условиях многоклассовой и несбалансированной выборки.

Таким образом, реализованный подход охватывал весь жизненный цикл моделей машинного обучения от подготовки данных и настройки гиперпараметров до обучения, валидации, визуализации результатов и их интерпретации. Это обеспечило всестороннюю оценку эффективности алгоритмов и позволило выделить наиболее перспективные решения для

автоматизированной классификации литологических типов пород по данным ГИС.

2.6 Метрики оценки и анализ результатов классификации

Для оценки эффективности моделей машинного обучения, реализованных в рамках задачи классификации литологических типов пород по геофизическим данным скважин, использовался комплекс метрик, отражающих как общее качество предсказаний, так и особенности многоклассовой и несбалансированной классификации. Основными метриками стали Accuracy, F1-score, Penalty Score, а также время обучения моделей.

Accuracy отражает долю правильно классифицированных объектов и служит базовой метрикой производительности. Однако при выраженном дисбалансе классов она может быть неинформативной, поскольку модель может демонстрировать высокую точность, преимущественно за счёт правильно предсказанных доминирующих классов.

F1-score обеспечивает более объективную оценку, поскольку учитывает как точность, так и полноту классификации по каждому классу. Усреднение F1-меры по всем классам позволяет оценить качество модели с учётом редких литологических типов.

Penalty Score специализированная метрика, отражающая тяжесть ошибок между различными литологиями. Она рассчитывается на основе штрафной матрицы (см. рисунок 2.8), в которой каждой паре классов сопоставляется вес, характеризующий степень их геологического сходства или различия.

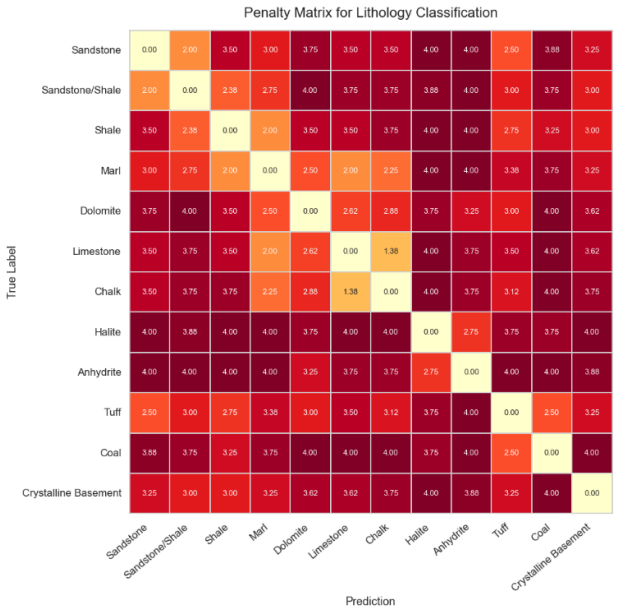


Рисунок 2.8 - Штрафная матрица для классификации литологических пород

Итоговая оценка Penalty Score определяется как среднее значение штрафов между предсказанными и истинными метками, взятое с обратным знаком: чем ближе значение к нулю, тем выше качество классификации.

Дополнительно учитывалось время обучения, как важный показатель вычислительной эффективности, особенно в условиях потенциального промышленного внедрения.

Также проводилась оценка значимости признаков для различных моделей. В деревообразных алгоритмах, например, Random Forest, использовалась встроенная метрика feature_importances_. На рисунке 2.9 видно, что наибольший вклад в классификацию вносят параметры RHOB, NPHI, GR, DTC и SP.

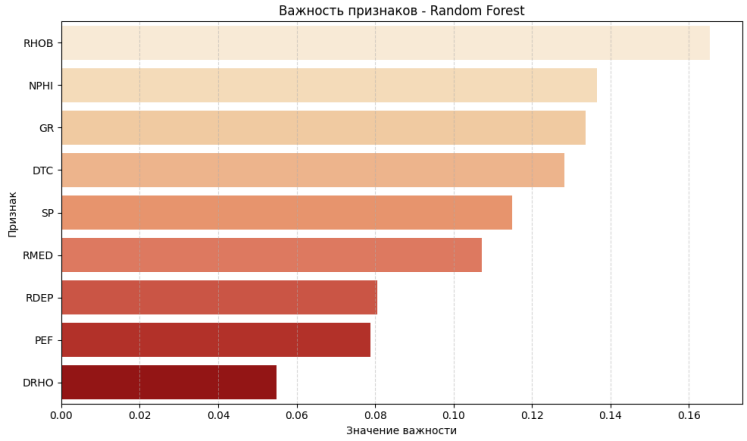


Рисунок 2.9 - Значимость признаков Random Forest

Для нейросетевых моделей, таких как Keras Neural Network и MLP Classifier, была использована библиотека SHAP, позволяющая интерпретировать вклад каждого признака в предсказание. На рисунке 2.10 видно, что для модели Keras наибольшее влияние оказывают GR, RHOB, NPHI и PEF, тогда как RDEP и RMED имеют наименьший вклад. Это подтверждает высокую информативность признаков, связанных с плотностью и составом пород.

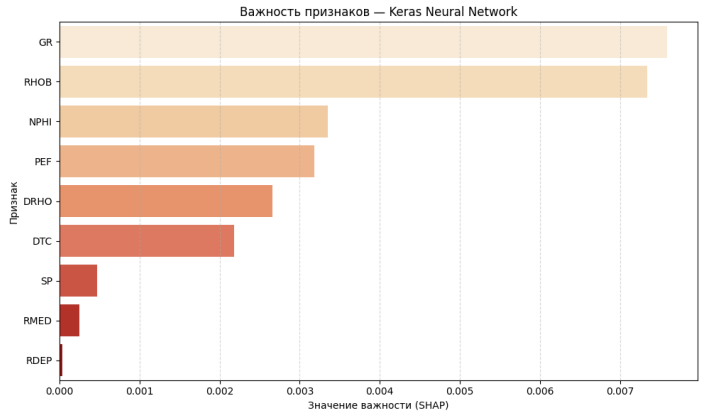


Рисунок 2.10 - Значимость признаков Keras Neural Network

Результаты тестирования моделей на валидационном наборе leaderboard_test представлены в таблице 2.3. Лучшие значения Accuracy 0.7344 и F1-score 0.6979 показала модель Random Forest, продемонстрировав также один из лучших Penalty Score -0.7274, что указывает на низкое количество серьёзных ошибок. Второе место заняла модель XGBoost, однако её Penalty Score был ниже -0.8425, что говорит о более "дорогих" ошибках. Keras Neural Network показала высокую обобщающую способность, но время её обучения превысило 290 секунд. CatBoost обеспечила устойчивую работу, но была самой медленной. MLP Classifier показал средние результаты при времени обучения более 5000 секунд. Наименее успешными оказались LightGBM и Decision Tree.

Таблица 2.3 - Результаты на leaderboard_test

Модель	Accuracy	F1-score	Penalty Score	Runtime (s)
Random Forest	0.7344	0.6979	-0.7274	102.84
XGBoost	0.704	0.6868	-0.8425	385.96
Keras Neural Network	0.6878	0.6788	-0.8324	293.03
CatBoost	0.6379	0.6637	-0.9663	1022.45
MLP Classifier	0.5967	0.6092	-1.1417	5047.72
Decision Tree	0.5871	0.5997	-1.1582	32.99
LightGBM	0.5474	0.6011	-1.2787	82.02

На скрытой тестовой выборке hidden_test распределение результатов изменилось (см. таблицу 2.4). Keras Neural Network показала лучшие значения по всем ключевым метрикам: Accuracy 0.7296, F1 0.7160, Penalty Score -0.7113. Это говорит о высокой устойчивости к переобучению и хорошей способности к обобщению.

Таблица 2.4 - Результаты на hidden_test

Model	Accuracy	F1-score	Penalty Score
Keras Neural Network	0.7296	0.716	-0.7113
Random Forest	0.683	0.6174	-0.9077
CatBoost	0.6806	0.6523	-0.9149
XGBoost	0.677	0.6221	-0.9347
LightGBM	0.6172	0.6185	-1.0559
MLP Classifier	0.5939	0.5793	-1.1436
Decision Tree	0.5907	0.5845	-1.2072

Визуальное сравнение моделей по основным метрикам Accuracy, F1-score, Penalty Score и времени обучения представлено на рисунках Приложения В. Эти графики позволяют быстро определить, какие алгоритмы демонстрируют наилучшее соотношение между качеством классификации и вычислительной

эффективностью.

Также в приложении содержатся матрицы ошибок для ключевых моделей и графики значимости признаков, демонстрирующие вклад каждого геофизического параметра в итоговое предсказание.

2.7 Оценка эффективности

Для объективной оценки эффективности реализованных моделей машинного обучения применялся комплексный подход, включающий как классические метрики, так и специализированный показатель Penalty Score, учитывающий степень ошибок между различными литологиями. Кроме того, оценивались вычислительные характеристики моделей, в частности время обучения, что критично при их практическом внедрении.

На валидационной выборке наилучшие результаты продемонстрировала модель Random Forest: Accuracy 0.7344, F1-score 0.6979 и один из лучших показателей Penalty Score -0.7274. Модель отличалась высокой интерпретируемостью и сбалансированным соотношением между качеством классификации и вычислительными затратами. Сравнимые результаты показали модели XGBoost и Keras Neural Network, однако у XGBoost оказался менее благоприятный Penalty Score, указывающий на более серьёзные ошибки между геологически различными породами.

На скрытой выборке, имитирующей реальные условия эксплуатации, Keras Neural Network показала наилучшие значения по всем ключевым метрикам: Accuracy 0.7296, F1-score 0.7160 и минимальный по модулю Penalty Score -0.7113. Это свидетельствует о высокой устойчивости к переобучению и способности к обобщению на новые данные. Модель Random Forest сохранила приемлемые показатели, но уступила Keras по всем метрикам, особенно по F1 и Penalty Score.

Алгоритмы CatBoost и XGBoost показали стабильные и предсказуемые результаты на обеих выборках, продемонстрировав надёжность, но уступили лидерам по точности и F1-score. LightGBM, MLP Classifier и Decision Tree оказались наименее успешными, особенно по Penalty Score, что указывает на частые критические ошибки между геологически несходными классами.

Анализ времени обучения показал, что самыми ресурсоёмкими моделями являются MLP и CatBoost более 1000–5000 секунд, в то время как Decision Tree и LightGBM обучаются значительно быстрее, но не обеспечивают удовлетворительного качества. Эти различия особенно важны при выборе моделей для промышленного применения, где ресурсы ограничены.

Визуализация результатов (см. Приложение В) подтверждает, что Keras Neural Network и Random Forest являются наиболее сбалансированными решениями с точки зрения точности, устойчивости и производительности. При этом нейросетевая модель превосходит в генерализации, а ансамблевая в

интерпретируемости и скорости.

Таким образом, проведённый анализ подтверждает высокую эффективность применения современных алгоритмов машинного обучения для автоматизированной интерпретации геофизических данных. Реализованный подход обеспечивает надёжную классификацию литологических типов пород и может быть рекомендован для использования в системах поддержки принятия решений в геологоразведке.

ЗАКЛЮЧЕНИЕ

В рамках дипломного проекта был разработан и реализован алгоритм автоматической классификации литологических типов пород по данным геофизических исследований скважин (ГИС) с использованием методов машинного обучения. В процессе работы был проведён комплексный анализ предметной области, обзор современных подходов к обработке каротажных данных, а также изучены, реализованы и сравнены различные алгоритмы классификации.

В ходе выполнения проекта были решены следующие задачи:

- проведён анализ существующих методов классификации литологических данных и определены наиболее перспективные подходы на основе машинного обучения;
- изучены научные публикации, современные исследования и открытые источники данных;
- выполнена предварительная обработка каротажных данных из набора FORCE 2020, включая интерполяцию, устранение выбросов, масштабирование и балансировку классов;
- реализованы и обучены модели машинного обучения: Decision Tree, Random Forest, XGBoost, LightGBM, CatBoost, MLP и Keras Neural Network;
- проведена всесторонняя оценка моделей с использованием метрик Accuracy, F1-score, Penalty Score, времени обучения и других;
- выполнен сравнительный анализ на открытой и скрытой выборках, обеспечивший объективную оценку качества и обобщающей способности моделей.

Разработанный алгоритм показал высокую точность и устойчивость к переобучению. Наилучшие результаты продемонстрировала модель Keras Neural Network, что подтверждает эффективность нейросетевых архитектур в задачах интерпретации геофизических данных.

Таким образом, все задачи дипломной работы выполнены, поставленная цель достигнута, а разработанный подход может быть рекомендован к использованию в практических системах анализа каротажной информации и поддержки принятия решений в геологоразведке.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1 Послание Главы государства Касым-Жомарта Токаева народу Казахстана «Экономический курс Справедливого Казахстана». 1 сентября 2023 г. https://www.akorda.kz/ru/poslanie-glavy-gosudarstva-kasym-zhomarta-tokaeva-narodu-kazahstana-ekonomicheskiy-kurs-spravedlivogo-kazahstana-18588

2 Калинникова М.В., Головин Б.А., Головин К.Б. Геофизические исследования скважин. Учебное пособие. М.: ГеоКнига, 2011. URL: https://www.geokniga.org/bookfiles/geokniga-geofizicheskie-issledovaniya-skvazhin_1.pdf

3 Лекция 7: Геофизические исследования скважин. Казанский федеральный университет. URL: https://kpfu.ru/staff_files/F_1211465147/Lekciya_5_GIS_geofizicheskie_issledovaniya_skvazhin.pdf

4 Mukhamediev R.I., Popova Y., Kuchin Y., Zaitseva E., Kalimoldayev A., Symagulov A., Levashenko V., Abdoldina F., Gopejenko V., Yakunin K., et al. Review of Artificial Intelligence and Machine Learning Technologies: Classification, Restrictions, Opportunities and Challenges // Mathematics. 2022. Vol. 10, No. 15. P. 2552. DOI: https://doi.org/10.3390/math10152552

5 Сахнюк В.И., Новиков Е.В., Шарифуллин А.М., Белохин В.С., Антонов А.П. Применение методов машинного обучения в обработке данных геофизических исследований скважин отложений викуловской свиты // Георесурсы. 2022. Т. 24, № 2. С. 230-238. DOI: https://doi.org/10.18599/grs.2022.2.21

6 Mukhamediev R., Kuchin Y., Yunicheva N., Kalpeyeva Z., Muhamedijeva E., Gopejenko V., Rystygulov P. Classification of Logging Data Using Machine Learning Algorithms // Applied Sciences. 2024. Vol. 14, No. 17. P. 7779. DOI: https://doi.org/10.3390/app14177779

7 Alaudah Y., Michalowicz P., Alfarraj M., AlRegib G. A Machine Learning Benchmark for Facies Classification // arXiv preprint. 2019. arXiv:1901.07659. URL: https://arxiv.org/abs/1901.07659

8 Shi Y., Liao J., Gan L., Tang R. Lithofacies Prediction from Well Log Data Based on Deep Learning: A Case Study from Southern Sichuan, China // Applied Sciences. 2024. Vol. 14, No. 18. P. 8195. DOI: https://doi.org/10.3390/app14188195

9 Zhang J., He Y., Zhang Y., Li W., Zhang J. Well-Logging-Based Lithology Classification Using Machine Learning Methods for High-Quality Reservoir Identification: A Case Study of Baikouquan Formation in Mahu Area of Junggar Basin, NW China // Energies. 2022. Vol. 15, No. 10. P. 3675. DOI: https://doi.org/10.3390/en15103675

10 Mousavi S.H.R., Hosseini-Nasab S.M. A Novel Approach to Classify Lithology of Reservoir Formations Using GrowNet and Deep-Insight with Physic-Based Feature Augmentation // Energy Science & Engineering. 2024. Vol. 12, No. 10. P. 4453-4477. DOI: https://doi.org/10.1002/ese3.1895

11 Kostorz W. A Practical Method for Well Log Data Classification // Computational Geosciences. 2021. Vol. 25, No. 1. P. 345-355. DOI: https://doi.org/10.1007/s10596-020-10011-4

12 About The Norwegian Offshore Directorate https://www.sodir.no/en/force/about-us/

13 FORCE 2020 Machine Learning Competition https://github.com/bolgebrygg/Force-2020-Machine-Learning-competition

14 Results of the FORCE 2020 lithology machine learning competition The Norwegian Offshore Directorate https://www.sodir.no/en/force/Previous-events/2020/results-of-the-FORCE-2020-lithology-competition/

Приложение А

Техническое задание

А.1 Техническое задание на разработку алгоритма классификации литологических данных по данным ГИС

Настоящее техническое задание распространяется на разработку и исследование алгоритмов машинного обучения, предназначенных для автоматизированной классификации литологических типов пород на основе данных геофизических исследований скважин (ГИС).

А1.1 Основание для разработки

Проект разрабатывается на основании приказа проректора университета № 1804-до от 25.11.24

А.1.2 Назначение

Алгоритм предназначен для сравнительного анализа автоматической классификации литологических типов горных пород на основе каротажных данных с помощью моделей машинного обучения. Основная задача заключается в выявления эффективной модели интерпретации ГИС данных при решении задач геологоразведки с применением методов машинного обучения.

А.1.3 Требования к функциональным характеристикам

Алгоритм должен обеспечивать:

- загрузку и предварительную обработку данных: устранение пропусков, устранение выбросов, масштабирование признаков;
- реализацию моделей машинного обучения: Decision Tree, Random Forest, XGBoost, LightGBM, CatBoost, MLP, Keras Neural Network;
- оценку качества моделей по метрикам: Accuracy, Precision, Recall, F1-score, Cohen’s Kappa, MCC, Penalty Score, время обучения;
- визуализацию результатов: confusion matrix, сравнение моделей, важности признаков.

А.1.4 Требования к надежности

Алгоритм должен обеспечивать воспроизводимость результатов при фиксированных параметрах, устойчивость к пропущенным значениям и корректную работу на больших объемах данных без ошибок.

А.1.5 Требованию к составу и параметрам технических средств

Работа должна выполняться в среде Jupyter Notebook с использованием языка программирования Python. Минимальные требования к оборудованию: процессор Intel Core i7, не менее 4 ГБ оперативной памяти.

Приложение Б

Листинг (код) программы

```
# Импорты
import pandas as pd
import numpy as np
import time
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, cohen_kappa_score, matthews_corrcoef, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from xgboost import XGBClassifier
import lightgbm as lgb
from catboost import CatBoostClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import StandardScaler
import shap
import warnings
warnings.filterwarnings("ignore")
# Обработка train.csv
print("Загрузка train.csv")
df = pd.read_csv('train.csv', sep=';')
print(f"Форма train.csv: {df.shape}")
# Маппинг литологий
lithology_keys = {
    30000: 'Sandstone',
    65030: 'Sandstone/Shale',
    65000: 'Shale',
    80000: 'Marl',
    74000: 'Dolomite',
    70000: 'Limestone',
    70032: 'Chalk',
    88000: 'Halite',
    86000: 'Anhydrite',
    99000: 'Tuff',
```

```
        90000: 'Coal',
        93000: 'Basement'
    }
    lithology_code_to_idx = {code: idx for idx, code in enumerate(lithology_keys.keys())}
    lithology_idx_to_code = {idx: code for code, idx in lithology_code_to_idx.items()}
    lithology_labels = list(lithology_keys.values())
    # Признаки
    features = [
        'GR',       # Гамма-каротаж
        'RHOB',     # Плотность породы
        'RDEP',     # Глубинное сопротивление
        'RMED',     # Среднее сопротивление
        'NPHI',     # Нейтронная пористость
        'PEF',      # Фотоэлектрический эффект
        'DTC',      # Скорость звука
        'DRHO',     # Разница плотностей
        'SP'        # Спонтанная поляризация
    ]
    # Заполнение пропусков интерполяцией и медианой
    def interpolate_fill(df, features, well_col='WELL', depth_col='DEPTH_MD'):
        print("Заполнение пропусков")
        for col in features:
            nan_before = df[col].isna().sum()
            # Интерполяция
            df[col] = df.groupby(well_col)[col].transform(
                lambda x: x.interpolate(method='linear', limit_direction='both')
            )
            #медиана
            df[col] = df[col].fillna(df[col].median())
            nan_after = df[col].isna().sum()
            print(f"- {col}: было {nan_before}, осталось {nan_after}")
        return df
    df = interpolate_fill(df, features)
    # Обработка выбросов (clipping по перцентилям)
    def clip_outliers(df, features, lower_percentile=0.005, upper_percentile=0.995):
        print("Обработка выбросов")
        for col in features:
            low, high = df[col].quantile([lower_percentile, upper_percentile])
            df[col] = df[col].clip(lower=low, upper=high)
            print(f"- {col}: обрезано ниже {low:.2f}, выше {high:.2f}")
        return df
    df = clip_outliers(df, features)
```

```
        # Кодирование целевой переменной
        df['target'] = df['FORCE_2020_LITHOFACIES_LITHOLOGY'].map(lithology_code_to_idx)
        # Балансировка классов
        print("Вычисление веса классов для балансировки")
        class_weights = compute_class_weight(class_weight='balanced',
                                             classes=np.unique(df['target']),
                                             y=df['target'])
        class_weight_dict = {i: w for i, w in enumerate(class_weights)}
        print("Вес классов:", class_weight_dict)
        # Подготовка данных
        X = df[features]
        y = df['target']
        # Масштабирование
        scaler = StandardScaler()
        X_train = X
        y_train = y
        X_train_scaled = scaler.fit_transform(X_train)

        # Обработка leaderboard_test
        print("Загрузка leaderboard_test")
        lb_features = pd.read_csv("leaderboard_test_features.csv", sep=";")
        lb_target = pd.read_csv("leaderboard_test_target.csv", sep=";")
        print(f"Форма leaderboard_test_features.csv: {lb_features.shape}")
        lb_features = interpolate_fill(lb_features, features)
        lb_features = clip_outliers(lb_features, features)
        X_test = lb_features[features]
        X_test_scaled = scaler.transform(X_test)
        y_test = lb_target['FORCE_2020_LITHOFACIES_LITHOLOGY'].map(lithology_code_to_idx)
        # Penalty matrix
        print("Загрузка penalty_matrix.npy")
        penalty_matrix = np.load('penalty_matrix.npy')
        # Функция для штрафа
        def penalty_score(y_true, y_pred, penalty_matrix):
            y_true = np.array(y_true).astype(int)
            y_pred = np.array(y_pred).astype(int)
            penalties = np.array([penalty_matrix[i, j] for i, j in zip(y_true, y_pred)])
            score = -np.mean(penalties)
            return score
        # Функция для матрицы
        def plot_confusion_matrices(y_true, y_pred, model_name):
            cm = confusion_matrix(y_true, y_pred)
```

```
        cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        fig, axes = plt.subplots(1, 2, figsize=(18, 7))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                    xticklabels=lithology_labels, yticklabels=lithology_labels,
ax=axes[0])
        axes[0].set_title(f'{model_name} - Confusion Matrix (Counts)')
        axes[0].set_xlabel('Predicted')
        axes[0].set_ylabel('True')
        sns.heatmap(cm_normalized, annot=True, fmt='.2f', cmap='YlGnBu',
                    xticklabels=lithology_labels, yticklabels=lithology_labels,
ax=axes[1])
        axes[1].set_title(f'{model_name} - Confusion Matrix (Normalized)')
        axes[1].set_xlabel('Predicted')
        axes[1].set_ylabel('True')
        plt.tight_layout()
        plt.show()
    # Функция для расчёта расширенных метрик
    def calculate_all_metrics(y_true, y_pred, penalty_matrix):
        metrics = {
            'Accuracy': accuracy_score(y_true, y_pred),
            'F1-score (weighted)': f1_score(y_true, y_pred, average='weighted'),
            'Precision (weighted)': precision_score(y_true, y_pred, average='weighted',
zero_division=0),
            'Recall (weighted)': recall_score(y_true, y_pred, average='weighted',
zero_division=0),
            'Cohen Kappa': cohen_kappa_score(y_true, y_pred),
            'MCC': matthews_corrcoef(y_true, y_pred),
            'Penalty Score': penalty_score(y_true, y_pred, penalty_matrix)
        }
        return metrics
    # Инициализация моделей
    models = [
        ('Decision Tree', DecisionTreeClassifier(class_weight=class_weight_dict,
random_state=42)),
        ('Random Forest', RandomForestClassifier(n_estimators=100,
class_weight=class_weight_dict, random_state=42, n_jobs=-1)),
        ('XGBoost', XGBClassifier(
            n_estimators=500,
            max_depth=10,
            learning_rate=0.05,
            subsample=0.8,
            colsample_bytree=0.8,
            objective='multi:softprob',
            num_class=len(lithology_labels),
```

```
        use_label_encoder=False,
        eval_metric='mlogloss',
        random_state=42
    )),
    ('LightGBM', lgb.LGBMClassifier(
        n_estimators=500,
        learning_rate=0.05,
        max_depth=12,
        min_child_samples=50,
        min_split_gain=0.05,
        subsample=0.8,
        colsample_bytree=0.8,
        class_weight='balanced',
        verbosity=-1,
        random_state=42
    )),
    ('CatBoost', CatBoostClassifier(
        iterations=1000,
        learning_rate=0.05,
        depth=8,
        l2_leaf_reg=5,
        bagging_temperature=0.5,
        random_strength=0.5,
        auto_class_weights='SqrtBalanced',
        loss_function='MultiClass',
        eval_metric='TotalF1',
        early_stopping_rounds=50,
        verbose=0,
        random_state=42
    )),
    ('MLP Classifier', MLPClassifier(
        hidden_layer_sizes=(512, 256, 128),
        activation='relu',
        solver='adam',
        alpha=1e-4,
        batch_size=512,
        learning_rate='adaptive',
        early_stopping=True,
        max_iter=500,
        verbose=0,
        random_state=42
    ))
]
# Keras Model
```

```
        def create_keras_model(input_dim, num_classes):
            model = Sequential([
                Dense(256, input_dim=input_dim, activation='relu'),
                Dropout(0.3),
                Dense(128, activation='relu'),
                Dropout(0.3),
                Dense(64, activation='relu'),
                Dense(num_classes, activation='softmax')
            ])
            model.compile(loss='categorical_crossentropy',                optimizer='adam',
metrics=['accuracy'])
            return model
        # Цикл обучения и оценки моделей
        results = []
        print("Обучение модели")
        for name, model in models:
            print(f"Обучение модель: {name}")
            start_time = time.time()
            if name in ['MLP Classifier']:
                model.fit(X_train_scaled, y_train)
                mlp_model = model
                y_pred = model.predict(X_test_scaled)
            else:
                model.fit(X_train, y_train)
                y_pred = model.predict(X_test)
            end_time = time.time()
            runtime = end_time - start_time
            metrics = calculate_all_metrics(y_test.values, y_pred, penalty_matrix)
            for key, value in metrics.items():
                print(f"- {key}: {value:.4f}")
            print(f"- Время обучения: {runtime:.2f} секунд")
            plot_confusion_matrices(y_test, y_pred, model_name=name)
            results.append({
                'Model': name,
                'Accuracy': metrics['Accuracy'],
                'F1-score': metrics['F1-score (weighted)'],
                'Penalty Score': metrics['Penalty Score'],
                'Runtime (s)': runtime
            })
        # Keras Neural Network
        print("Обучение модели: Keras Neural Network")
        start_time = time.time()

        # One-hot encoding целевых меток
```

```
        y_train_cat = to_categorical(y_train, num_classes=len(lithology_labels))
        y_test_cat = to_categorical(y_test, num_classes=len(lithology_labels))

        # Callbacks
        early_stop = EarlyStopping(patience=5, restore_best_weights=True)
        reduce_lr = ReduceLROnPlateau(patience=3, factor=0.5)

        # Создание модели
        keras_model = create_keras_model(input_dim=X_train_scaled.shape[1],
num_classes=len(lithology_labels))

        # Обучение модели
        keras_model.fit(
            X_train_scaled, y_train_cat,
            validation_data=(X_test_scaled, y_test_cat),
            epochs=50,
            batch_size=256,
            callbacks=[early_stop, reduce_lr],
            verbose=1
        )

        end_time = time.time()

        # Предсказание
        y_pred_keras = np.argmax(keras_model.predict(X_test_scaled), axis=1)

        # Метрики
        runtime = end_time - start_time
        metrics = calculate_all_metrics(y_test.values, y_pred_keras, penalty_matrix)

        # Вывод метрик
        for key, value in metrics.items():
            print(f"- {key}: {value:.4f}")
        print(f"- Время обучения: {runtime:.2f} секунд")

        # Матрица ошибок
        plot_confusion_matrices(y_test, y_pred_keras, model_name="Keras NN")

        # Добавление результатов
        results.append({
            'Model': 'Keras Neural Network',
            'Accuracy': metrics['Accuracy'],
            'F1-score': metrics['F1-score (weighted)'],
            'Penalty Score': metrics['Penalty Score'],
```

```
    'Runtime (s)': runtime
})
# Преобразование результатов в DataFrame
results_df = pd.DataFrame(results)
results_df = results_df.round(4)

# Сортировка по Accuracy
results_df.sort_values(by='Accuracy', ascending=False, inplace=True)

# Печать итоговой таблицы
print("Итоговые результаты на leaderboard_test.csv")
print(results_df)

# Визуализация Accuracy
plt.figure(figsize=(12, 6))
sns.barplot(x='Accuracy', y='Model', data=results_df, palette='Blues_d')
plt.title('Accuracy моделей на валидации')
plt.xlabel('Accuracy')
plt.ylabel('Модель')
plt.xlim(0, 1)
plt.grid(True, axis='x', linestyle='--', alpha=0.5)
plt.show()

# Визуализация F1-score
plt.figure(figsize=(12, 6))
sns.barplot(x='F1-score', y='Model', data=results_df, palette='Greens_d')
plt.title('F1-score моделей на валидации')
plt.xlabel('F1-score (weighted)')
plt.ylabel('Модель')
plt.xlim(0, 1)
plt.grid(True, axis='x', linestyle='--', alpha=0.5)
plt.show()

# Визуализация Penalty Score
plt.figure(figsize=(12, 6))
sns.barplot(x='Penalty Score', y='Model', data=results_df, palette='Reds_d')
plt.title('Penalty Score моделей на валидации')
plt.xlabel('Penalty Score')
plt.ylabel('Модель')
plt.grid(True, axis='x', linestyle='--', alpha=0.5)
plt.show()

# Визуализация времени обучения
plt.figure(figsize=(12, 6))
```

```
    sns.barplot(x='Runtime (s)', y='Model', data=results_df, palette='Purples_d')
    plt.title('Время обучения моделей')
    plt.xlabel('Время (секунды)')
    plt.ylabel('Модель')
    plt.grid(True, axis='x', linestyle='--', alpha=0.5)
    plt.show()
    # Инференс hidden_test.csv
    # Обработка hidden_test.csv
    print("Загрузка hidden_test.csv")
    hidden_test = pd.read_csv('hidden_test.csv', sep=';')
    print(f"Форма hidden_test.csv: {hidden_test.shape}")
    hidden_test = interpolate_fill(hidden_test, features)
    hidden_test = clip_outliers(hidden_test, features)
    X_hidden = hidden_test[features]
    y_hidden_true = hidden_test['FORCE_2020_LITHOFACIES_LITHOLOGY'].map(lithology_code_to_idx)
    X_hidden_scaled = scaler.transform(X_hidden)
    # Сбор результатов инференса на hidden_test
    hidden_results = []
    print("Итоговые результаты на hidden_test.csv")
    # Инференс для всех моделей
    for name, model in models:
        print(f"Предсказывание для модели: {name}")
        if name == 'MLP Classifier':
            y_hidden_pred = model.predict(X_hidden_scaled)
        else:
            y_hidden_pred = model.predict(X_hidden)
        # Расчёт расширенных метрик
        metrics = calculate_all_metrics(y_hidden_true.values, y_hidden_pred, penalty_matrix)
        print(f"Метрики для {name} (hidden_test):")
        for key, value in metrics.items():
            print(f"- {key}: {value:.4f}")
        # Матрица ошибок
        plot_confusion_matrices(y_hidden_true, y_hidden_pred, model_name=f"{name} (hidden_test)")
        # Добавление результатов
        hidden_results.append({
            'Model': name,
            'Accuracy': metrics['Accuracy'],
            'F1-score': metrics['F1-score (weighted)'],
            'Penalty Score': metrics['Penalty Score']
        })
```

```
        # Инференс для Keras модели
        print(f"Предсказывание для модели: Keras Neural Network")
        y_hidden_pred_keras = np.argmax(keras_model.predict(X_hidden_scaled), axis=1)
        # Метрики Keras
        metrics = calculate_all_metrics(y_hidden_true.values, y_hidden_pred_keras, penalty_matrix)
        print(f"Метрики для Keras Neural Network (hidden_test):")
        for key, value in metrics.items():
            print(f"- {key}: {value:.4f}")
        # Матрица ошибок
        plot_confusion_matrices(y_hidden_true, y_hidden_pred_keras, model_name="Keras Neural Network (hidden_test)")
        # Добавление результатов
        hidden_results.append({
            'Model': 'Keras Neural Network',
            'Accuracy': metrics['Accuracy'],
            'F1-score': metrics['F1-score (weighted)'],
            'Penalty Score': metrics['Penalty Score']
        })


        # Преобразование и сортировка
        hidden_results_df = pd.DataFrame(hidden_results)
        hidden_results_df = hidden_results_df.round(4)
        hidden_results_df.sort_values(by='Accuracy', ascending=False, inplace=True)

        # Печать финальной таблицы
        print("Финальные метрики на hidden_test.csv")
        print(hidden_results_df)

        # Визуализация Accuracy
        plt.figure(figsize=(12, 6))
        sns.barplot(x='Accuracy', y='Model', data=hidden_results_df, palette='Blues_d')
        plt.title('Accuracy моделей на hidden_test')
        plt.xlabel('Accuracy')
        plt.ylabel('Модель')
        plt.xlim(0, 1)
        plt.grid(True, axis='x', linestyle='--', alpha=0.5)
        plt.show()

        # Визуализация F1-score
        plt.figure(figsize=(12, 6))
```

```
    sns.barplot(x='F1-score', y='Model', data=hidden_results_df, palette='Greens_d')
    plt.title('F1-score моделей на hidden_test')
    plt.xlabel('F1-score (weighted)')
    plt.ylabel('Модель')
    plt.xlim(0, 1)
    plt.grid(True, axis='x', linestyle='--', alpha=0.5)
    plt.show()

    # Визуализация Penalty Score
    plt.figure(figsize=(12, 6))
    sns.barplot(x='Penalty Score', y='Model', data=hidden_results_df, palette='Reds_d')
    plt.title('Penalty Score моделей на hidden_test')
    plt.xlabel('Penalty Score')
    plt.ylabel('Модель')
    plt.grid(True, axis='x', linestyle='--', alpha=0.5)
    plt.show()
    # Функция визуализации важности признаков
    def plot_feature_importances(model, model_name, feature_names):
        if hasattr(model, 'feature_importances_'):
            importances = model.feature_importances_
        elif hasattr(model, 'get_feature_importance'):
            importances = model.get_feature_importance()
        else:
            print(f"Модель {model_name} не поддерживает извлечение важности признаков.")
            return

        importance_df = pd.DataFrame({
            'Feature': feature_names,
            'Importance': importances
        }).sort_values(by='Importance', ascending=False)

        plt.figure(figsize=(10, 6))
        sns.barplot(x='Importance', y='Feature', data=importance_df, palette='OrRd')
        plt.title(f'Важность признаков - {model_name}')
        plt.xlabel('Значение важности')
        plt.ylabel('Признак')
        plt.grid(True, axis='x', linestyle='--', alpha=0.5)
        plt.tight_layout()
        plt.show()

    print("Визуализация важности признаков для моделей:")
```

```
for name, model in models:
    if name in ['Decision Tree', 'Random Forest', 'XGBoost', 'LightGBM', 'CatBoost']:
        print(f"- {name}")
        plot_feature_importances(model, name, features)



# Функция визуализации важности признаков с использованием SHAP
def plot_shap_feature_importance(model, X_background, X_sample, features, model_name="Model", is_sklearn=False):

    # Используем model.predict вместо model, если это sklearn-модель
    explainer_input = model.predict if is_sklearn else model

    # SHAP-объяснитель
    explainer = shap.Explainer(explainer_input, X_background)
    shap_values = explainer(X_sample)

    # Усреднение по классам
    if shap_values.values.ndim == 3:
        shap_vals_mean = np.abs(shap_values.values).mean(axis=2)
    else:
        shap_vals_mean = np.abs(shap_values.values)

    # Средняя важность признаков
    importances = shap_vals_mean.mean(axis=0)

    # Таблица важностей
    importance_df = pd.DataFrame({
        'Feature': features,
        'Importance': importances
    }).sort_values(by='Importance', ascending=False)

    # Визуализация
    plt.figure(figsize=(10, 6))
    sns.barplot(x='Importance', y='Feature', data=importance_df, palette='OrRd')
    plt.title(f'Важность признаков - {model_name}')
    plt.xlabel('Значение важности (SHAP)')
    plt.ylabel('Признак')
    plt.grid(axis='x', linestyle='--', alpha=0.5)
    plt.tight_layout()
    plt.show()
```

```
# Для Keras:
plot_shap_feature_importance(
    model=keras_model,
    X_background=X_train_scaled[:100],
    X_sample=X_test_scaled[:100],
    features=features,
    model_name="Keras Neural Network"
)

# Для MLPClassifier:
plot_shap_feature_importance(
    model=mlp_model,
    X_background=X_train_scaled[:100],
    X_sample=X_test_scaled[:100],
    features=features,
    model_name="MLP Classifier (sklearn)",
    is_sklearn=True
)
```

Приложение В

Визуализация результатов

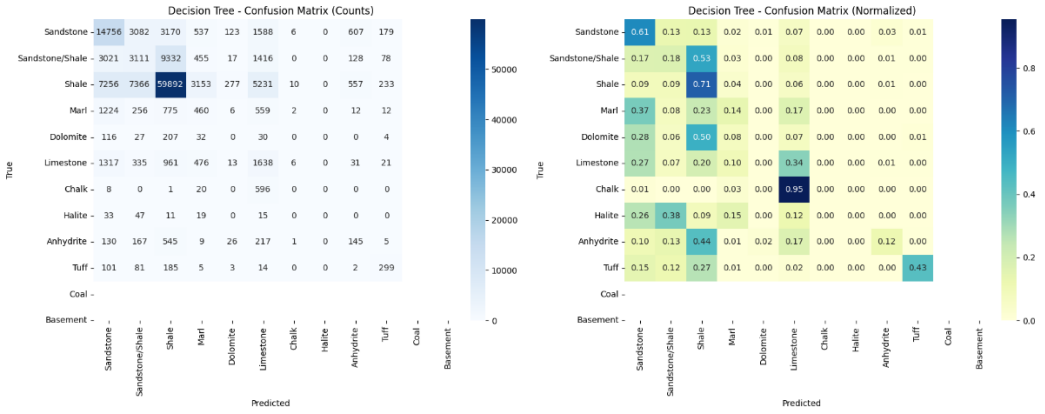


Рисунок В.1 - Матрица ошибок модели Decision Tree на валидации

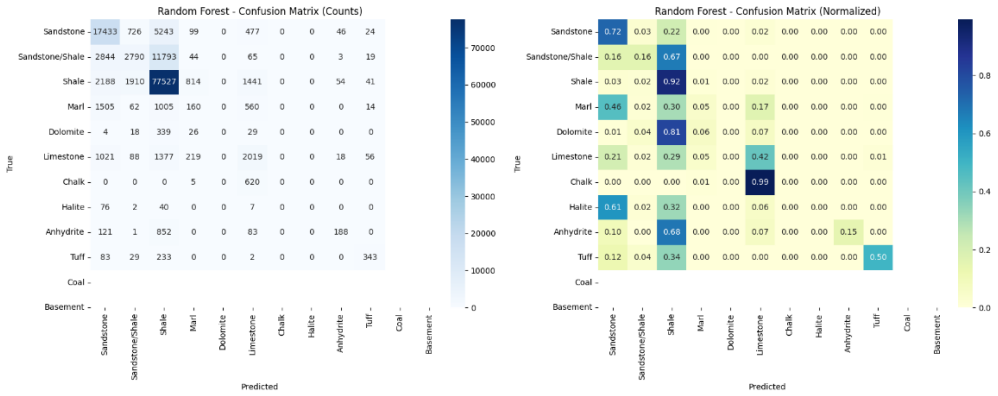


Рисунок В.2 - Матрица ошибок модели Random Forest на валидации

Рисунок В.3 - Матрица ошибок модели XGBoost на валидации

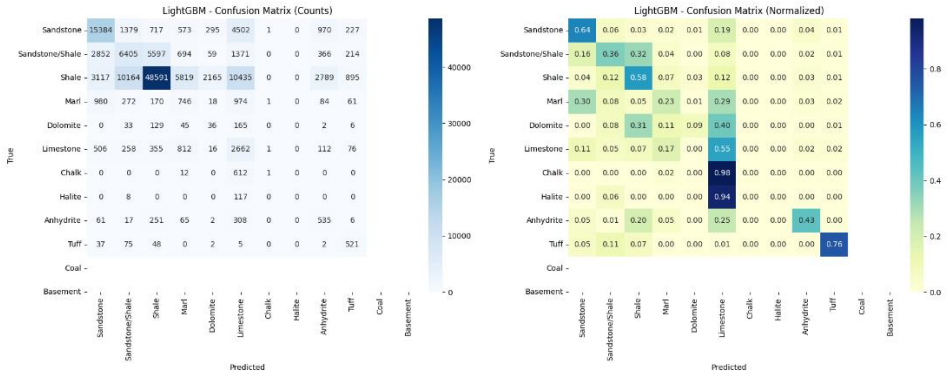


Рисунок В.4 - Матрица ошибок модели LightGBM на валидации

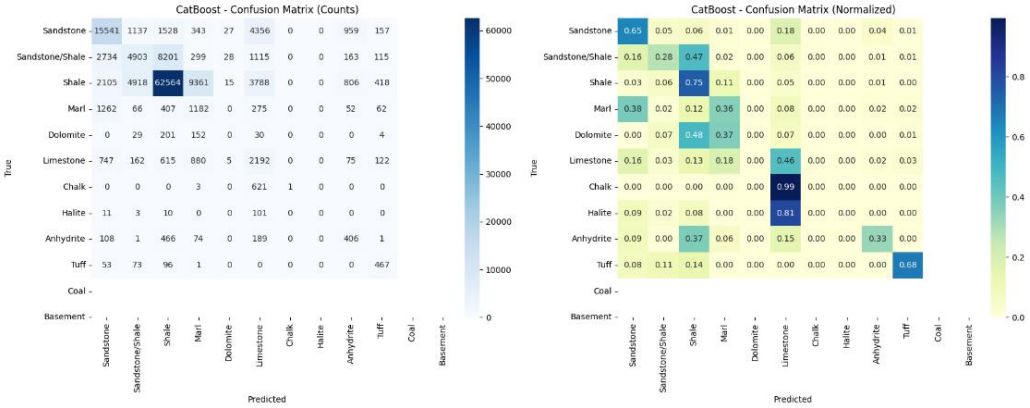


Рисунок В.5 - Матрица ошибок модели CatBoost на валидации

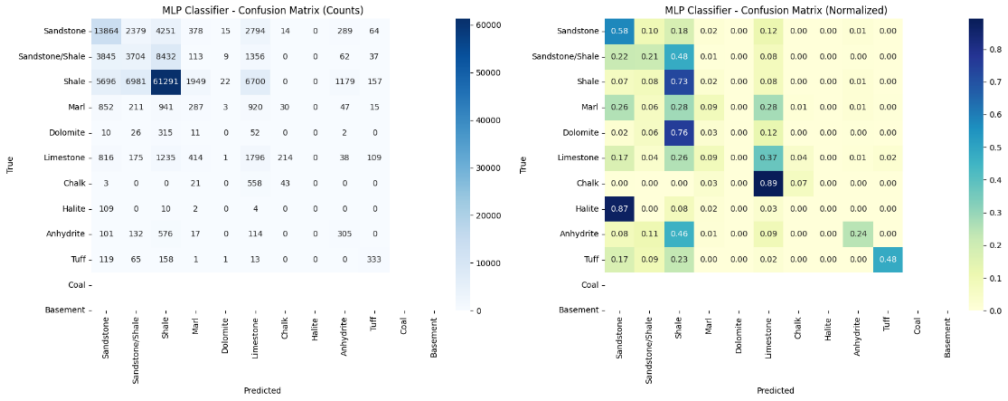


Рисунок В.6 - Матрица ошибок модели MLP Classifier на валидации

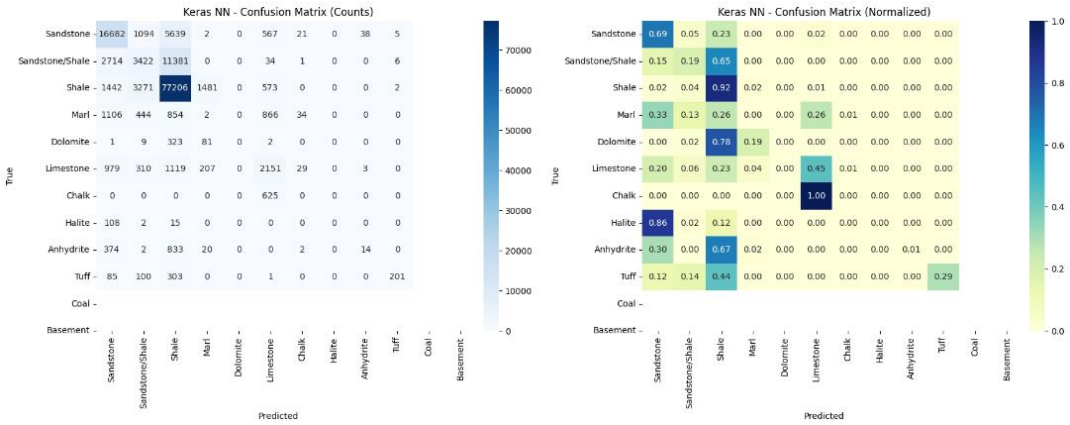


Рисунок В.7 - Матрица ошибок модели Keras Neural Network на валидации

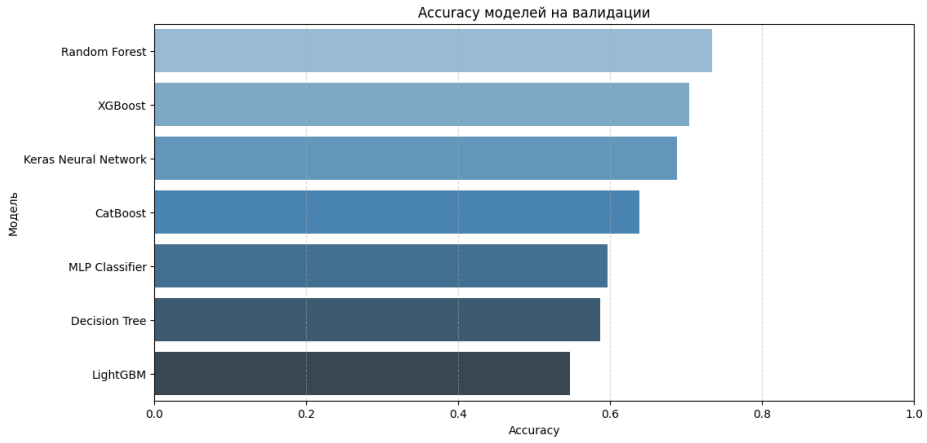


Рисунок В.8 - Accuracy моделей на валидации

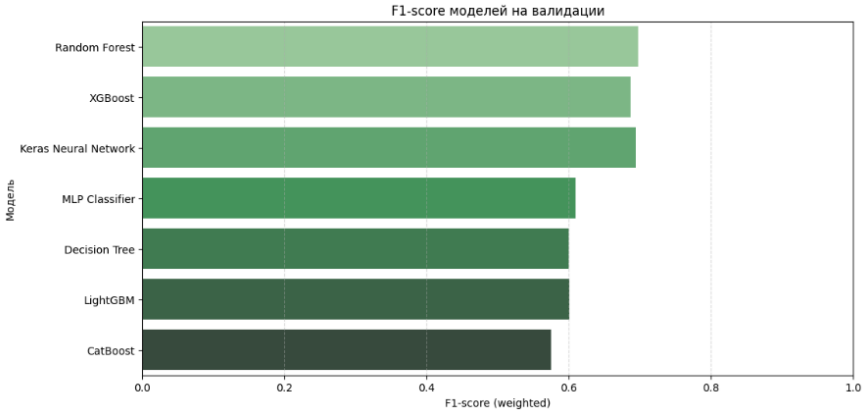


Рисунок В.9 - F1-score моделей на валидации

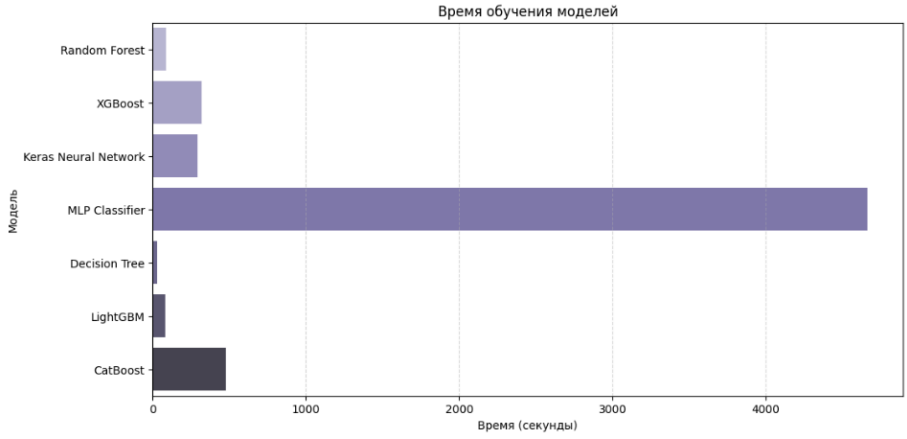


Рисунок В.10 - Время обучения моделей

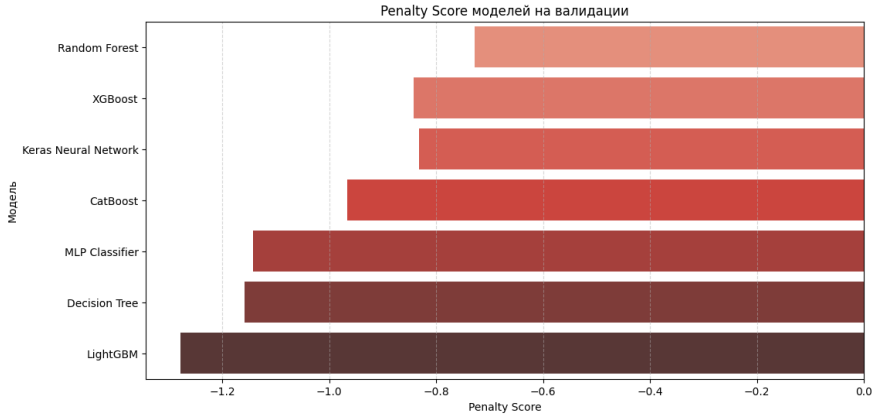


Рисунок В.11 - Penalty Score моделей на валидации

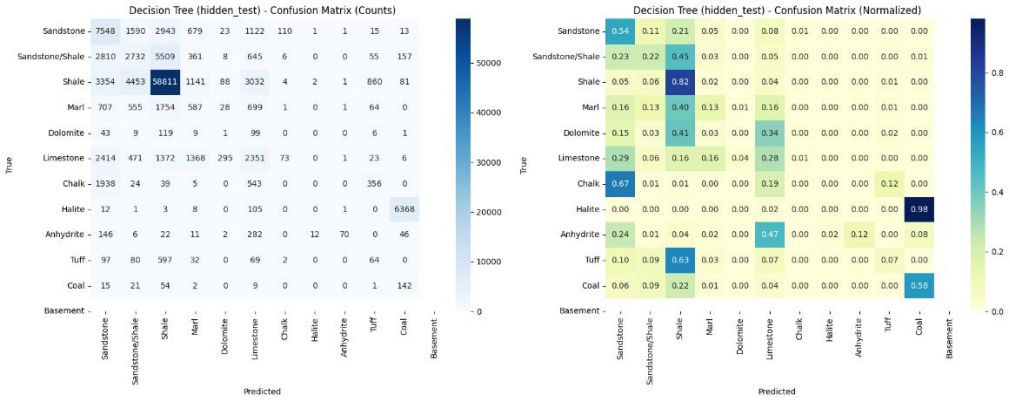


Рисунок В.12 - Матрица ошибок модели Decision Tree на hidden_test

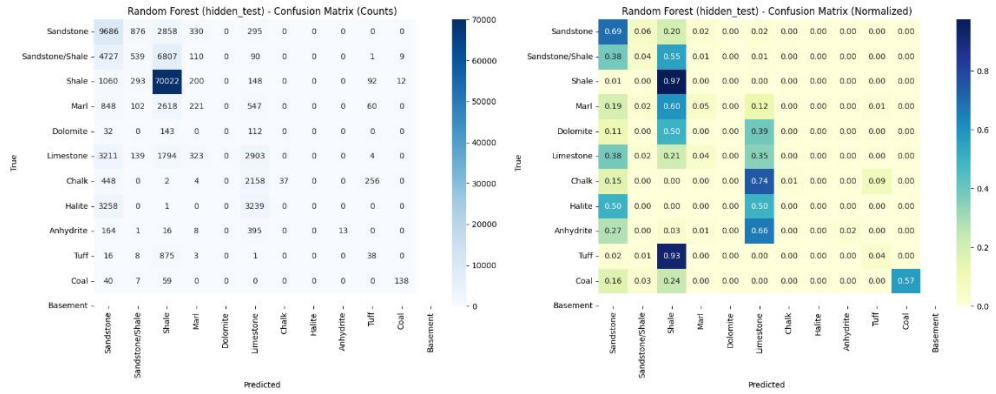


Рисунок B.12 - Матрица ошибок модели Random Forest на hidden_test

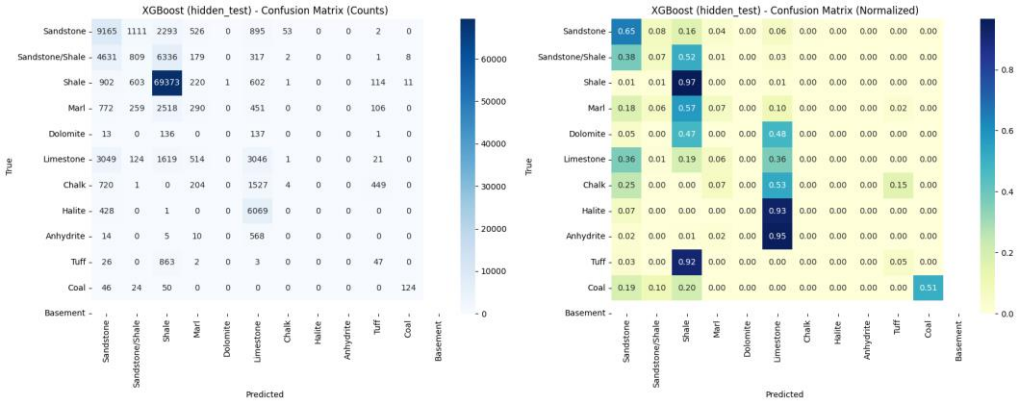


Рисунок B.13 - Матрица ошибок модели XGBoost на hidden_test

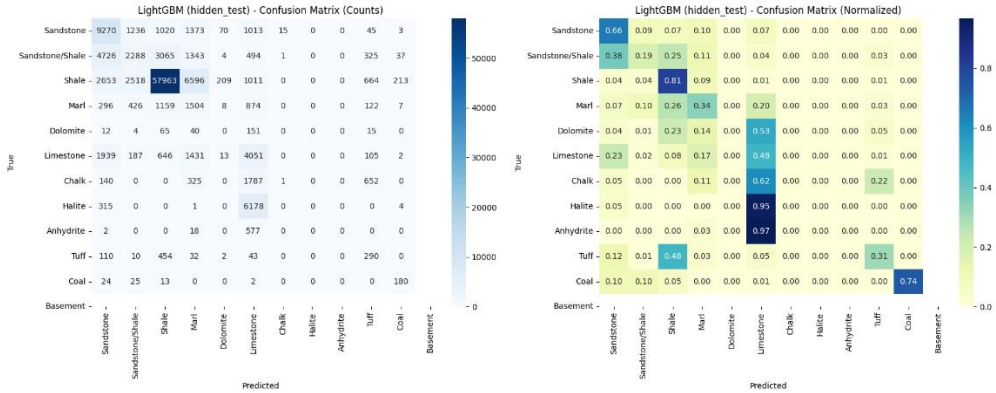


Рисунок B.14 - Матрица ошибок модели LightGBM на hidden_test

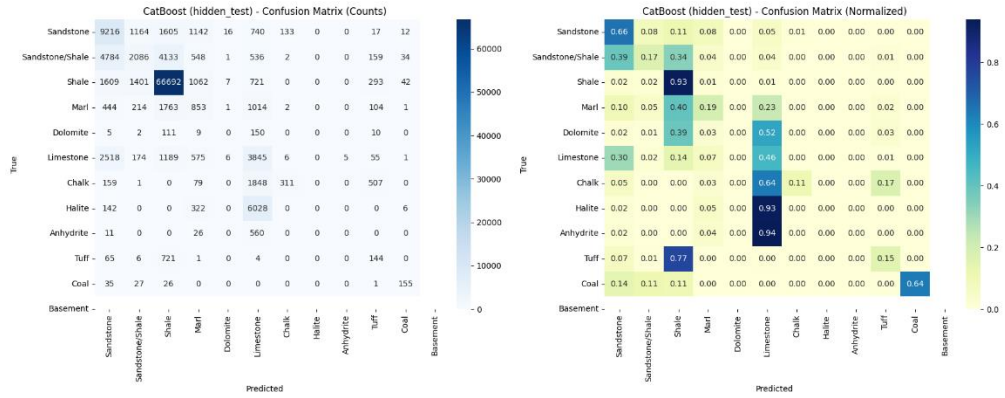


Рисунок B.15 - Матрица ошибок модели CatBoost на hidden_test

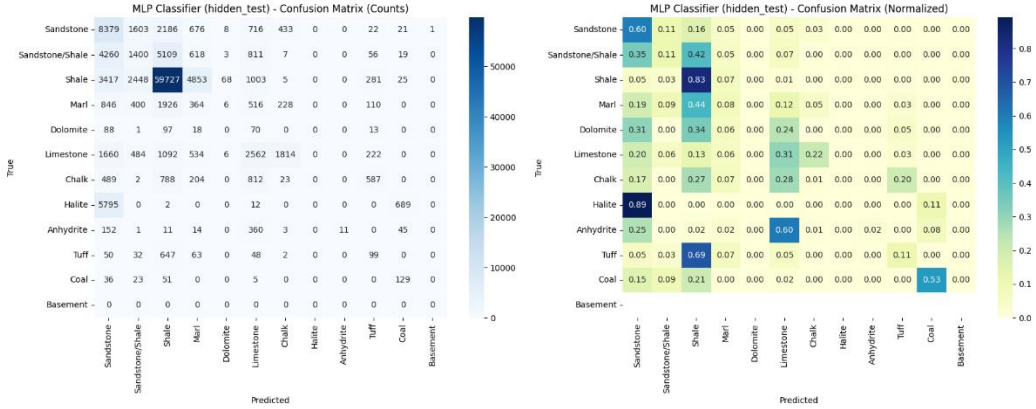


Рисунок B.16 - Матрица ошибок модели MLP Classifier на hidden_test

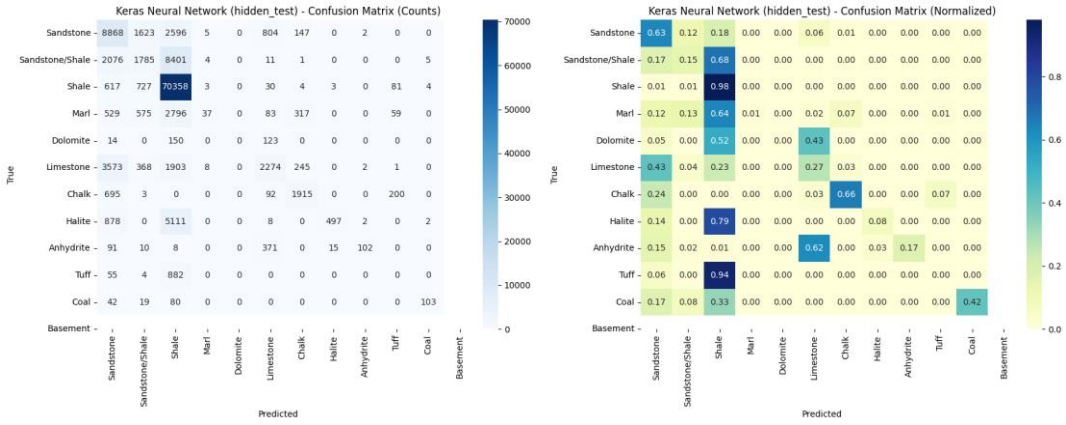


Рисунок B.17 - Матрица ошибок модели Keras Neural Network на hidden_test

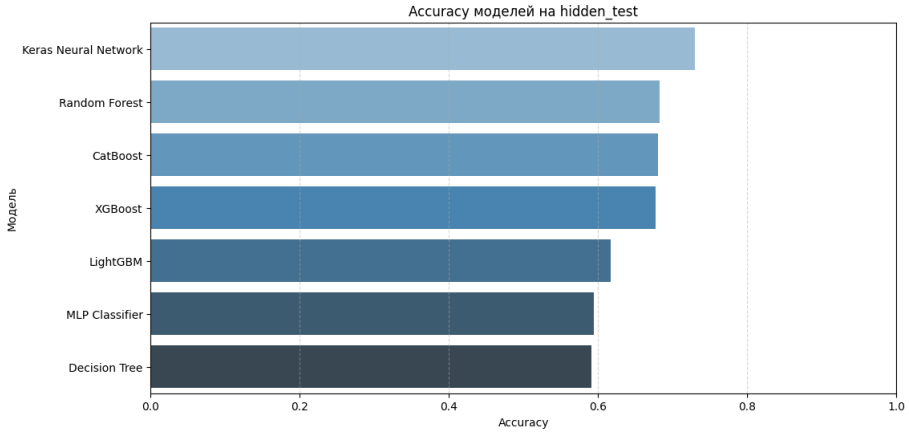


Рисунок В.18 - Accuracy моделей на hidden_test

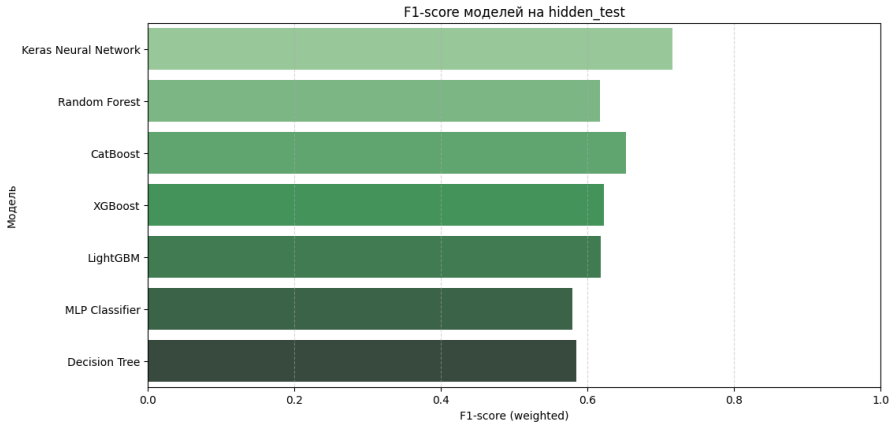


Рисунок В.19 - F1-score моделей на hidden_test

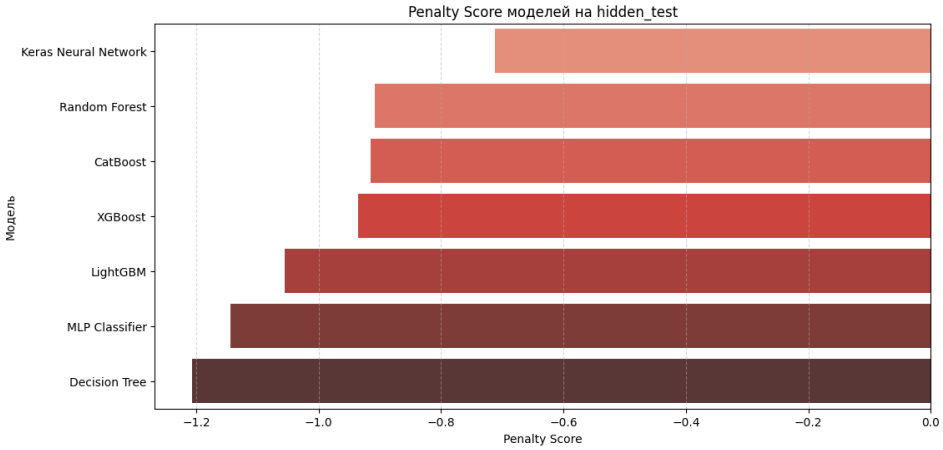
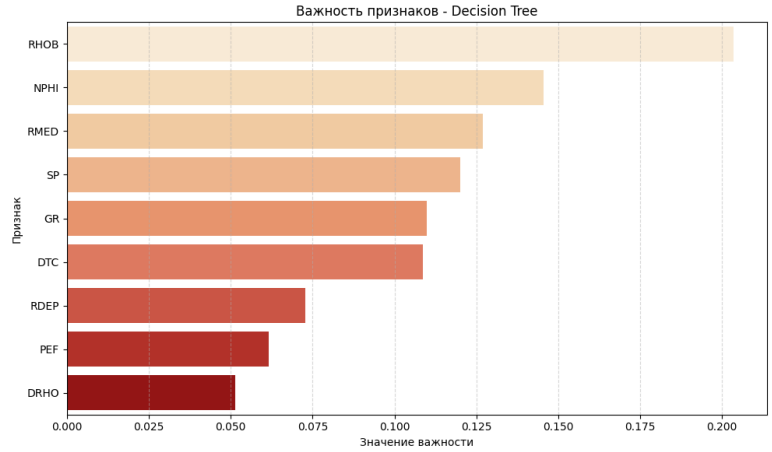
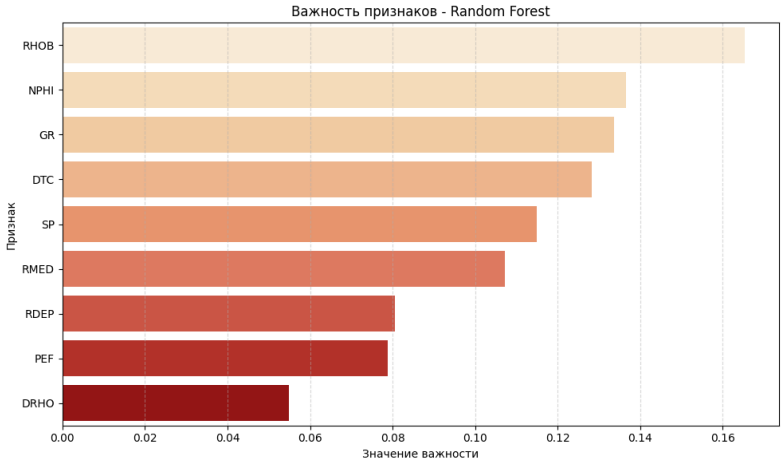


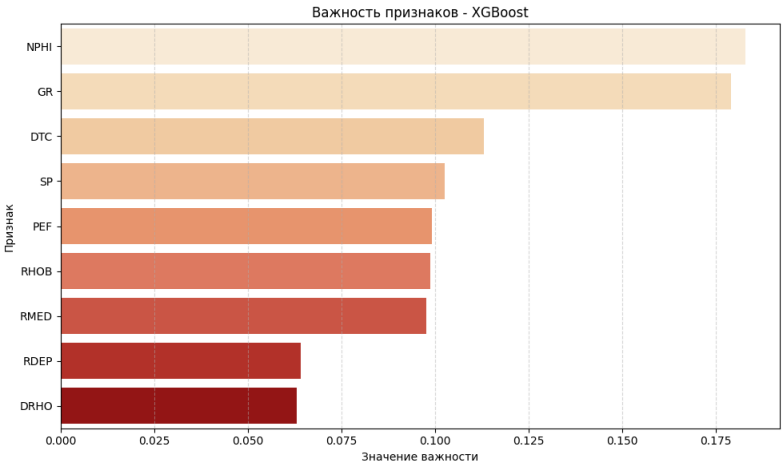
Рисунок В.20 - Penalty Score моделей на hidden_test.



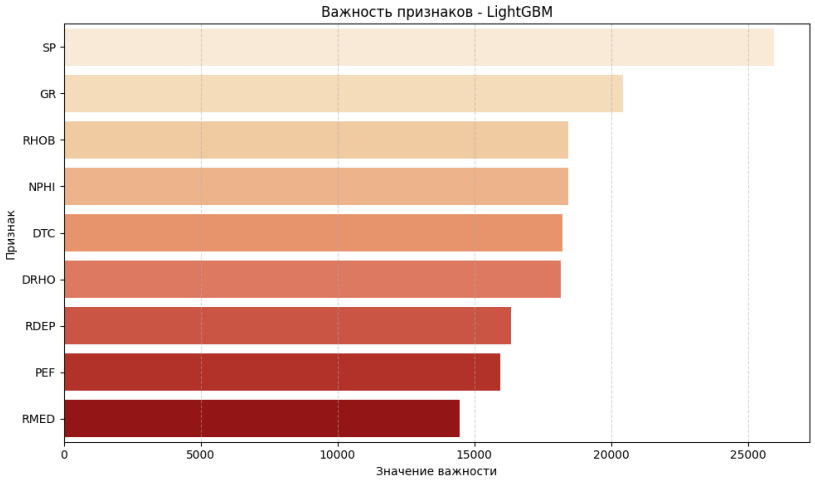
B.21 - Decision Tree: важность признаков



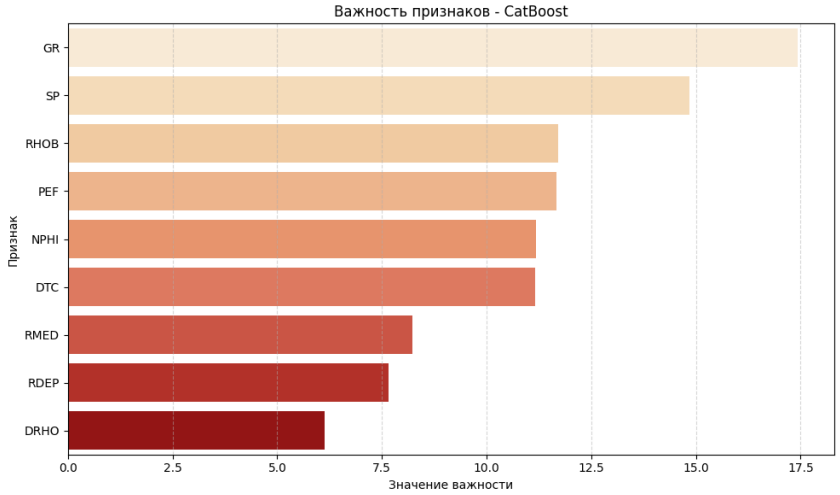
B.22 - Random Forest: важность признаков



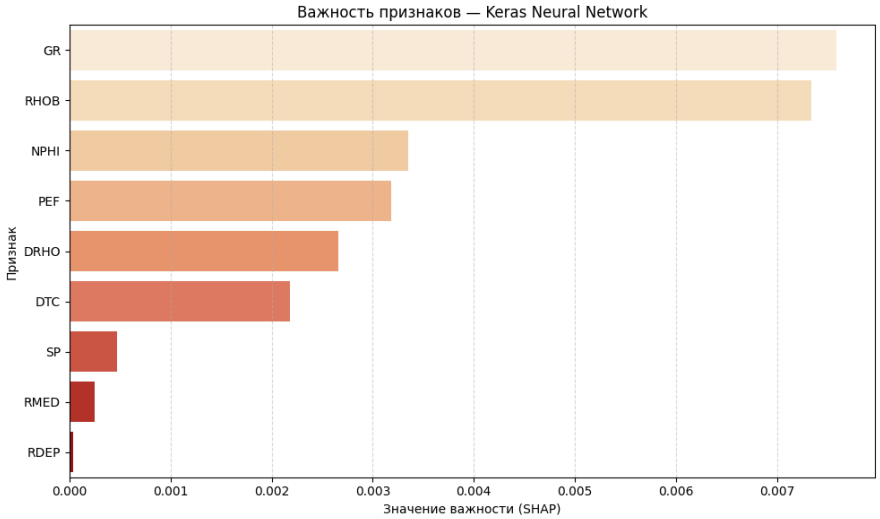
B.23 - XGBoost: важность признаков



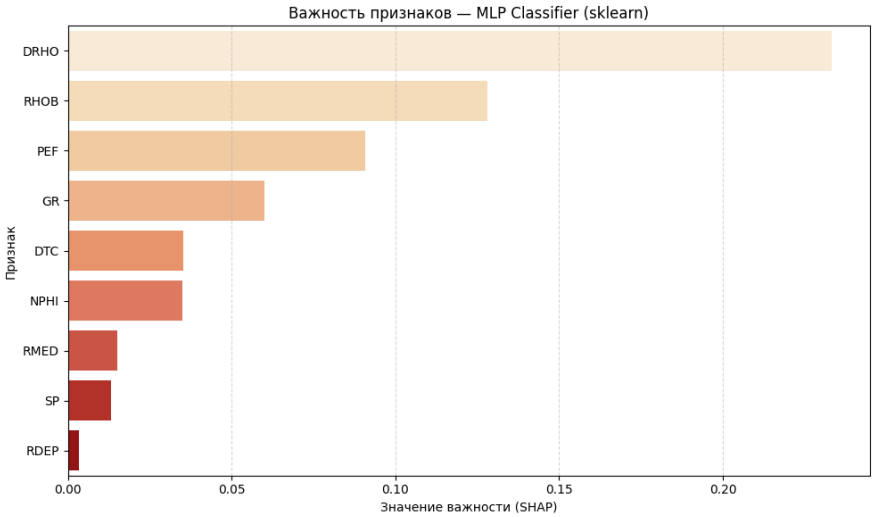
B.24 - LightGBM: важность признаков



B.25 - CatBoost: важность признаков



B.26 - Keras Neural Network: важность признаков (SHAP)



B.27 - MLP Classifier: важность признаков (SHAP)