Федеральное государственное автономное

образовательное учреждение высшего образования

«Пермский государственный национальный исследовательский университет» (ПГНИУ)

Региональный институт непрерывного образования (РИНО ПГНИУ)

Цифровая кафедра

Выпускная аттестационная (квалификационная) работа

по курсу профессиональной переподготовки «Анализ данных и машинное обучение»

**Построение модели для анализа факторов, влияющих на качество воды**

Разработчики проекта:

Богомаз Виктор Артемович,

Ксенофонтов Антон Витальевич,

Матвеев Егор Дмитриевич

Пермь, 2024

**Оглавление**

[ПАСПОРТ ПРОЕКТА 3](#_heading=h.30j0zll)

[СОДЕРЖАНИЕ ПРОЕКТА 4](#_heading=h.1fob9te)

[Анализ проблемы исследования 4](#_heading=h.3znysh7)

[Исходные данные](#_heading=h.2et92p0) 5

[Реализация проекта](#_heading=h.tyjcwt) 6

[Этап 1. Подготовка данных к анализу](#_heading=h.3dy6vkm) 6

[Этап 2. Предварительный анализ данных](#_heading=h.1t3h5sf) 9

[Этап 3. Корреляционный анализ данных](#_heading=h.4d34og8) 17

[Этап 4. Моделирование и прогнозирование](#_heading=h.2s8eyo1) 19

[ЗАКЛЮЧЕНИЕ 3](#_heading=h.3rdcrjn)2

[СПИСОК ЛИТЕРАТУРЫ](#_heading=h.3rdcrjn) 34

ПАСПОРТ ПРОЕКТА

**Название проекта:**Построение модели для анализа факторов, влияющих на качество воды

**Сведения об авторах:**Богомаз Виктор Артемович, Ксенофонтов Антон Витальевич, Матвеев Егор Дмитриевич

**Цель:**проанализировать данные по показателям качества воды и построить модель бинарной классификации.

**Задачи:**

1. Выполнить анализ проблемы, обосновать её актуальность.
2. Осуществить загрузку данных и подготовку их к анализу количественными методами, включая устранение пропущенных значений.
3. Выполнить предварительный анализ данных, в том числе выявление и обработку выбросов, корреляционный анализ.
4. Осуществить моделирование зависимости целевого признака от факторных различными методами, в том числе подобрать наилучшую модель, оценить её качество и выполнить прогнозирование.
5. Выполнить интерпретацию полученных результатов и сделать выводы о достижении цели.

**Краткое описание проекта:**

Требуется проанализировать данные по показателям качества воды и построить модель бинарной классификации. Подобрать метрику качества модели и объяснить её выбор. Выявить характеристики, наиболее существенные для модели. Сделать выводы.

**Конкретные ожидаемые результаты:**

Построенная модель бинарной классификации и рекомендации по её использованию.

**СОДЕРЖАНИЕ ПРОЕКТА**

**Анализ проблемы исследования**

Качество воды играет ключевую роль в обеспечении здоровья населения и устойчивого развития общества. Потребление некачественной воды может стать причиной различных заболеваний, включая инфекционные и хронические. Кроме того, обеспечение доступа к чистой воде является одной из целей устойчивого развития, определённых ООН, что подчёркивает глобальную значимость данной проблемы.

Современные подходы к мониторингу качества воды требуют обработки больших объёмов данных, содержащих физико-химические показатели воды, такие как кислотно-щелочной баланс (pH), твёрдость воды, содержание органического углерода и другие. Однако наличие большого количества данных не гарантирует их эффективного использования, так как для выявления закономерностей и взаимосвязей требуется применение методов машинного обучения и анализа данных.

На сегодняшний день одной из сложностей является разработка моделей, которые могут не только классифицировать пригодность воды для питья, но и выявлять наиболее значимые факторы, влияющие на её качество. Это особенно важно для целей улучшения процессов водоочистки и профилактики загрязнения водных ресурсов.

В рамках данного исследования поставлена задача построения модели бинарной классификации, способной предсказывать пригодность воды для питья на основе её характеристик.

Актуальность темы обусловлена глобальным характером проблемы качества воды, её влиянием на здоровье населения и экологию, а также необходимостью внедрения автоматизированных решений для анализа и прогнозирования состояния водных ресурсов.

**Исходные данные**

В настоящей работе анализируются факторы, влияющие на качество воды, с акцентом на его значимость для здоровья населения и устойчивого развития.

Список колонок анализируемого набора данных:

1. **PH** - важный параметр при оценке кислотно-щелочного баланса воды;
2. **Hardness** - твердость воды;
3. **Solids -** твердые вещества;
4. **Chloramines -** хлорамины: хлор и хлорамин
5. **Sulfate** - сульфат;
6. **Conductivity** - проводимость;
7. **Organic\_carbon** - органический\_углерод;
8. **Trihalomethanes** - тригалометаны;
9. **Turbidity** - мутность;
10. **Potability** – пригодность (**целевая переменная).**

Необходимо проанализировать факторы, влияющие на качество воды и определить её пригодность к питью.

Выдвинем гипотезу исследования: качество воды в определённой местности определяется множеством факторов и эти зависимости могут быть эффективно описаны и предсказаны с помощью методов машинного обучения на основе параметров воды.

**Реализация проекта**

**Этап 1. Подготовка данных к анализу**

Загрузим данные в датафрейм и подключим необходимые библиотеки:

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import StratifiedKFold

from sklearn.metrics import classification\_report

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import ConfusionMatrixDisplay

from imblearn.over\_sampling import SMOTE

from imblearn.over\_sampling import RandomOverSampler

from imblearn.over\_sampling import ADASYN

from imblearn.combine import SMOTETomek

from imblearn.combine import SMOTEENN

from sklearn.model\_selection import GridSearchCV

from sklearn import model\_selection, preprocessing, ensemble, metrics

import warnings

warnings.filterwarnings('ignore')

Загрузим данные через GoogleDrive:

# Подключаемся к диску

from google.colab import drive

drive.mount('/content/drive')

# Читаем датасет

dataset = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/data/water\_potability.csv')

# Выводим первые строки датасета

dataset.head()

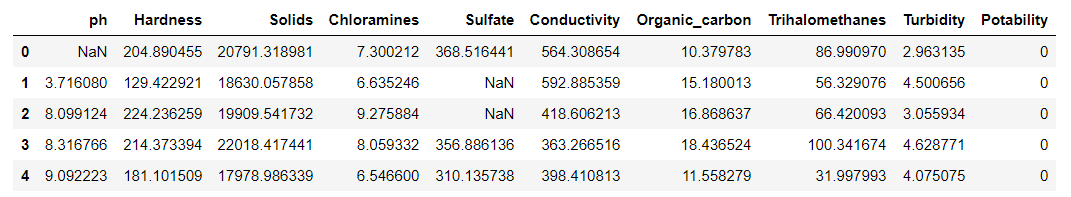


Рисунок 1. Исходный датафрейм

Проверим данные на пропуски.

# Проверим, сколько пропущенных значений в датасете

np.count\_nonzero(dataset.isnull().sum(axis= 1).values)



Убедимся, что все количественные столбцы имеют числовой тип. Если это не так, выполним преобразование типа столбца к числовому.

dataset.info()

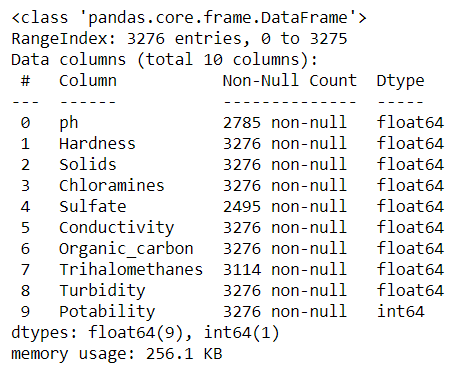


Рисунок 2. Типы данных колонок

Видно, что все колонки обладают числовым типом.

Все пропуски заполним импьютацией (при помощи модели RandomForestRegressor):

from sklearn.ensemble import RandomForestRegressor

# Для 'ph'

ph\_data = dataset[dataset['ph'].notna()]

X = ph\_data.drop(columns=['ph'])

y = ph\_data['ph']

model = RandomForestRegressor()

model.fit(X, y)

missing\_ph = dataset[dataset['ph'].isna()]

dataset.loc[dataset['ph'].isna(), 'ph'] = model.predict(missing\_ph.drop(columns=['ph']))

# Для 'Sulfate'

Sulfate\_data = dataset[dataset['Sulfate'].notna()]

X = Sulfate\_data.drop(columns=['Sulfate'])

y = Sulfate\_data['Sulfate']

model = RandomForestRegressor()

model.fit(X, y)

missing\_Sulfate = dataset[dataset['Sulfate'].isna()]

dataset.loc[dataset['Sulfate'].isna(), 'Sulfate'] = model.predict(missing\_Sulfate.drop(columns=['Sulfate']))

# Для 'Trihalomethanes'

Trihalomethanes\_data = dataset[dataset['Trihalomethanes'].notna()]

X = Trihalomethanes\_data.drop(columns=['Trihalomethanes'])

y = Trihalomethanes\_data['Trihalomethanes']

model = RandomForestRegressor()

model.fit(X, y)

missing\_Trihalomethanes = dataset[dataset['Trihalomethanes'].isna()]

dataset.loc[dataset['Trihalomethanes'].isna(), 'Trihalomethanes'] = model.predict(missing\_Trihalomethanes.drop(columns=['Trihalomethanes']))

# Датасет после заполнения пропусков

dataset

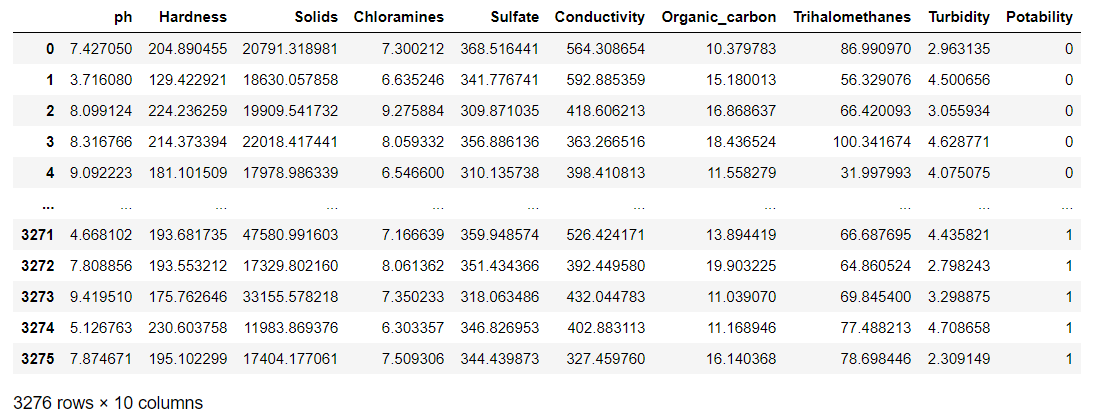


Рисунок 3. Датасет после заполнения пропусков

Итак, результат первого этапа – это готовый к анализу набор данных в виде датафрейма.

**Этап 2. Предварительный анализ данных**

Вычислим описательные статистики по колонкам (среднее, моду, медиану, стандартное отклонение, квартили).

***Среднее арифметическое*** *равно сумме значений всех вариант выборки, делённой на объем выборки:*

.

Здесь *п* − объем выборки, а *xi* − варианты выборки.

***Модой*** называется значение признака, встречающееся в выборке наиболее часто. Условимся использовать для обозначения моды символы *Mo*.

В случае несгруппированных данных для нахождения медианы необходимо ранжировать выборку, т. е. расположить данные в порядке их возрастания или убывания. Медианой будет являться значение признака, находящееся в середине ранжированного ряда. Медиана находится по формуле

Выборочная дисперсия находится по формуле *.*

Используется также другая формула для вычисления дисперсии: , где *.*

Дисперсия имеет размерность квадрата размерности случайной величины, что затрудняет её интерпретацию и делает не очень наглядной. Для более наглядного описания рассеяния удобнее пользоваться характеристикой, размерность которой совпадает с размерностью исследуемого признака. С этой целью вводится понятие ***стандартного отклонения*** (или ***среднего квадратичного отклонения***).

***Стандартным отклонением*** называется положительный квадратный корень из дисперсии:

.

Стандартное отклонение имеет те же единицы измерения, что и результаты измерения исследуемого признака, и, таким образом, оно характеризует степень отклонения признака от среднего арифметического. Иными словами, оно показывает, как расположена основная часть вариант относительно среднего арифметического.

dataset.describe()

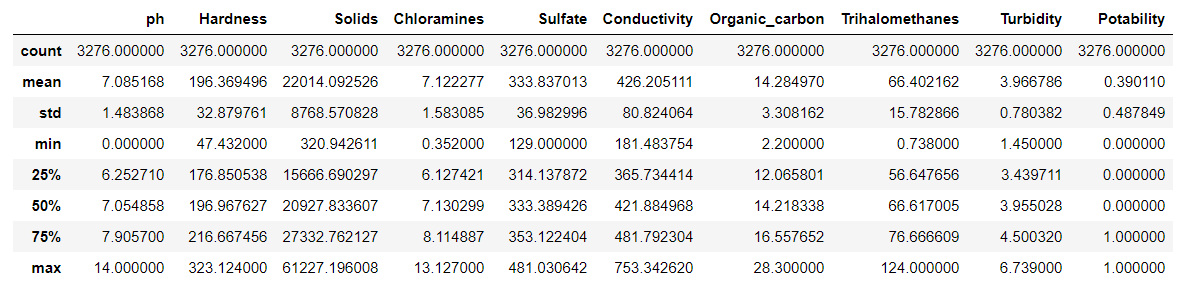


Рисунок 4. Описательные статистики по колонкам

***Аномальными наблюдениями*** (*выбросами*, англ. *Outliers, Extreme values*) называют такие значения уровня временного ряда, которые значительно отличаются от остальных. При выявлении подобных «выбросов» возникают серьёзные вопросы: являются ли отклоняющиеся данные действительно ошибками (например, регистрации) или это реальные значения и как получить адекватные оценки для параметров изучаемой совокупности.

Проверим данные на наличие выбросов, для этого можно использовать диаграмму «ящик с усами» (boxplot). Если выбросов мало, то следует их сгладить.

График ***«ящик с усами»,*** или ***«ящичковая диаграмма»***, или ***диаграмма размаха*** − график, используемый в описательной статистике и компактно изображающий одномерное [распределение вероятностей](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9). Такой вид диаграммы в удобной форме показывает медиану, нижний и верхний квартили, минимальное и максимальное значения выборки и [выбросы](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81_(%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0)).

Несколько таких ящиков можно нарисовать рядом друг с другом, чтобы визуально сравнивать одно распределение с другим, их можно рисовать горизонтально либо вертикально. Расстояния между различными частями ящика позволяют определить степень распространения (дисперсии) и асимметрии в данных и выявить выбросы.

Границами ящика служат первый и третий [квартили](https://ru.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D1%80%D1%82%D0%B8%D0%BB%D1%8C) (25-й и 75-й [процентили](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%BD%D1%82%D0%B8%D0%BB%D1%8C) соответственно), линия в середине ящика — медиана (50-й процентиль). Концы усов — края статистически значимой выборки (без [выбросов](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81_(%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0))) могут определяться несколькими способами. В общем виде эта формула имеет вид

*.*

*X*н — нижняя граница уса, *X*в — верхняя граница уса, *Q*1 — первый квартиль ,*Q*3 — третий квартиль, *k* — коэффициент, наиболее часто употребляемое значение которого равно 1,5. Данные, выходящие за границы усов ([выбросы](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81_(%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0))), отображаются на графике в виде точек, маленьких кружков или звёздочек. Иногда на графике отмечают среднее арифметическое и его [доверительный интервал](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%B2%D0%B5%D1%80%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%B2%D0%B0%D0%BB_%D0%B4%D0%BB%D1%8F_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B6%D0%B8%D0%B4%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B9_%D0%B2%D1%8B%D0%B1%D0%BE%D1%80%D0%BA%D0%B8) («зарубка» на ящике). На рис. изображён график «ящик с усами».

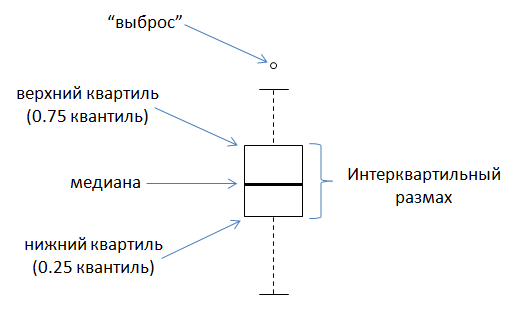


Рисунок 5. «Ящик с усами»

Посмотрим на диаграммы boxplot всех числовых колонок датафрейма:

dataset.boxplot();

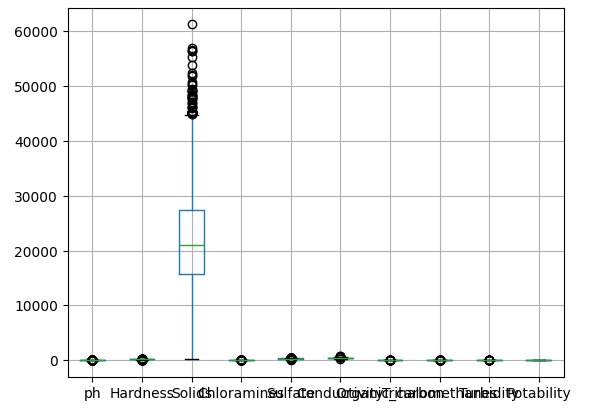


Рисунок 6. Диаграммы boxplot всех числовых колонок

Напишем функцию для удаления выбросов:

# Функция для удаления выбросов по IQR для признаков

def remove\_outliers\_iqr(df, feature):

data = df[feature].values

q25, q75 = np.percentile(data, 25), np.percentile(data, 75)

iqr = q75 - q25

cut\_off = iqr \* 1.5

lower, upper = q25 - cut\_off, q75 + cut\_off

return df.drop(df[(df[feature] > upper) | (df[feature] < lower)].index)

new\_df\_without\_outliers = dataset

features\_to\_check = list(dataset.corr()['Potability'][(dataset.corr()['Potability'] != 1)].index) # Берём признаки (кроме целевой переменной)

for feature in features\_to\_check:

# Вызов функции для признаков

new\_df\_without\_outliers = remove\_outliers\_iqr(new\_df\_without\_outliers, feature)

# Выведем датасет после удаления выбросов

new\_df\_without\_outliers

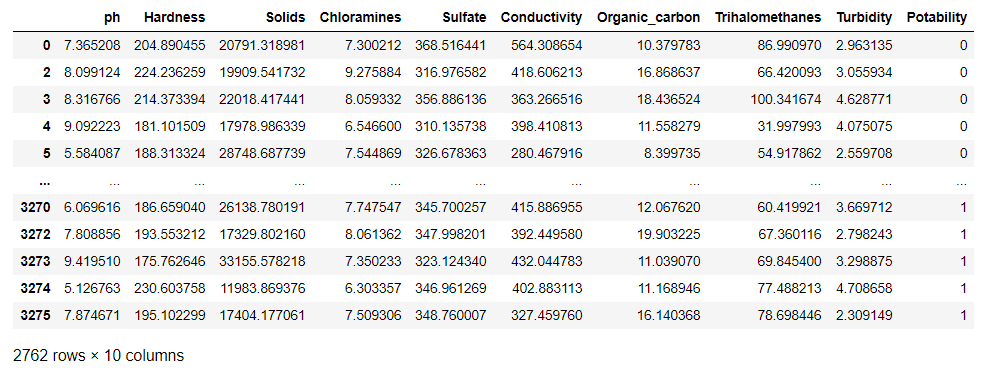


Рисунок 7. Визуализация датасета после удаления объектов

Видно, что из датасета удалилось 507 выбросов.

Посмотрим на диаграммы boxplot всех числовых колонок датафрейма после удаления выбросов:

new\_df\_without\_outliers.boxplot()

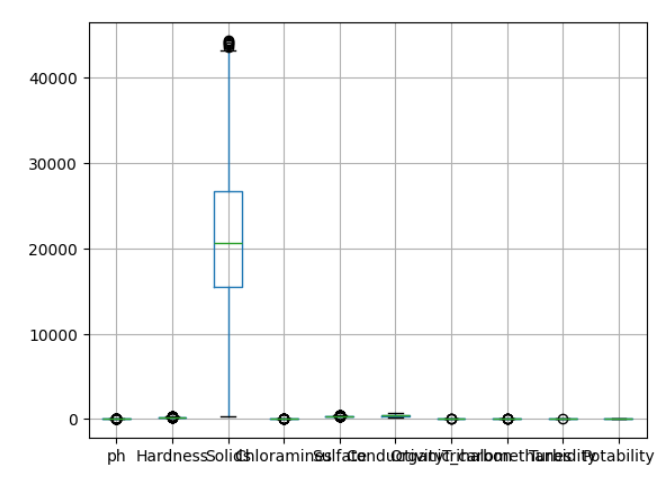


Рисунок 8. Диаграммы boxplot всех числовых колонок после удаления выбросов

Посмотрим на баланс классов при помощи гистограммы:

y\_name = "Potability" # Целевая переменная

# Считаем количество значений и сортируем их

ax = dataset[y\_name].value\_counts().sort\_values().plot(kind="barh")

totals= [] # Высота столбцов

for i in ax.patches:

totals.append(i.get\_width())

total = sum(totals) # Суммарная высота

# Выводим названия классов и процентное соотношение

for i in ax.patches:

ax.text(i.get\_width()+.3, i.get\_y()+.20,

str(round((i.get\_width()/total)\*100, 2))+'%',

fontsize=10, color='black')

ax.grid(axis="x") # Выводи мсетку по x

plt.suptitle(y\_name, fontsize=20) # Название графика

plt.show()

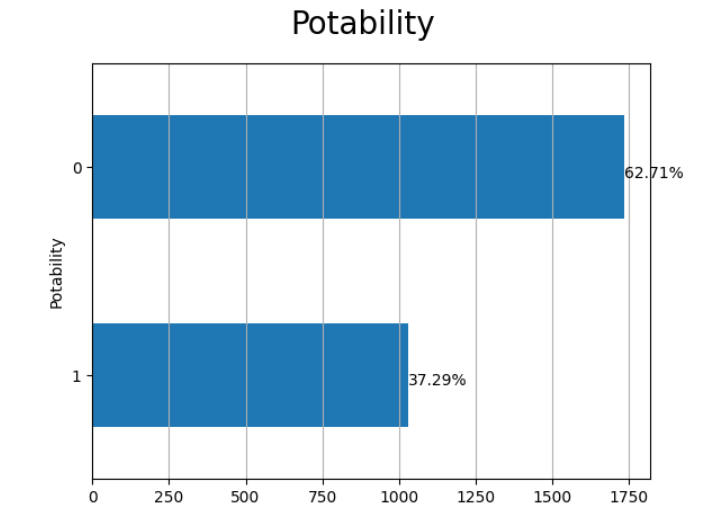


Рисунок 9. Визуализация распределения классов

Видно, что классы не сбалансированы. Поэтому проведём балансировку.

Мы применили различные методы балансировки: RandomOverSampler, SMOTE, ADASYN, SMOTETomek и SMOTEENN и выбрали лучший - **RandomOverSampler**.

X = dataset.drop('Potability', axis=1)

y = dataset['Potability']

# Разобьём данные на обучающую и тестовую выборки, используя соотношение 8 к 2

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=159)

ros = RandomOverSampler(random\_state=159)

X\_resampled, y\_resampled = ros.fit\_resample(X\_train, y\_train)

Данные в столбцах отличаются по размерности, поэтому нужно их нормализовать.

# Создаем объект StandardScaler

scaler = StandardScaler()

# Применяем нормализацию ко всем столбцам

X\_train\_norm = pd.DataFrame(scaler.fit\_transform(X\_resampled), columns=dataset.drop(y\_name, axis = 1).columns)

X\_test\_norm = pd.DataFrame(scaler.transform(X\_test), columns=dataset.drop(y\_name, axis = 1).columns)

y\_train\_norm = y\_resampled

y\_test\_norm = y\_test

X\_train\_norm.head()

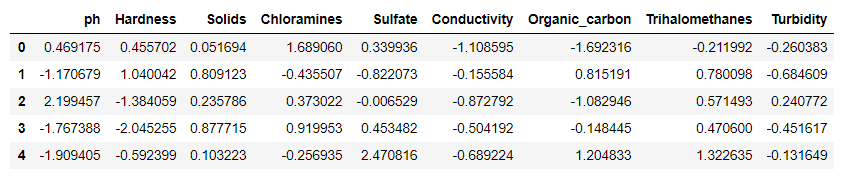


Рисунок 10. Датасет после нормализации

**Этап 3. Корреляционный анализ данных**

***Корреляционный анализ*** – это совокупность методов оценивания степени тесноты статистической связи между анализируемыми переменными.

Выполним корреляционный анализ данных с помощью матрицы корреляции. В случае нормальности всех данных следует использовать коэффициент корреляции Пирсона, в противном случае – ранговые коэффициенты корреляции Спирмена.

**Парный коэффициент** корреляции характеризует взаимосвязь двух переменных на фоне действия остальных показателей и является самым распространенным показателем тесноты связи при статистическом анализе данных.

Парный коэффициент корреляции между количественными случайными переменными и носит название *выборочного коэффициента корреляции* Пирсона (*sample correlation coefficient*) (или просто коэффициента корреляции) и находится по формуле

где и  — *выборочные дисперсии (sample variances*) переменных  и , а — *выборочная ковариация* или выборочный ковариационный момент, и соответствующие *средние (means)* определяются по формулам



Коэффициент корреляции обладает следующими свойствами:

1. Принимает значения от –1 до +1.
2. Если , то связь между переменными  и  считается сильной. Если , то связь слабая.
3. Если , то корреляционное поле наблюдений представляет собой совокупность точек, которые можно расположить на одной прямой. Знак «+» свидетельствует о прямой линейной зависимости между переменными и , а знак «—» − об обратной линейной зависимости.
4. При  линейная корреляционная связь отсутствует.

Метод ранговой корреляции Спирмена позволяет определить тесноту (силу) и направление корреляционной связи между двумя признаками (как количественными, так и качественными). Коэффициент ранговой корреляции имеет границы изменения от –1 до +1. Полное совпадение рангов означает максимально тесную прямую связь, полная противоположность рангов – максимально тесную обратную связь. Формула расчета ***коэффициента корреляции рангов Ч. Спирмена:***

где  – ранг  в выборке .

Матрицу корреляции отобразим с помощью диаграммы «тепловая карта» (heatmap).

# Матрица корреляции

plt.figure(figsize=(15, 10)).add\_subplot(1,1,1)

sns.heatmap(dataset.corr(), annot = True, cmap='coolwarm')

plt.show()

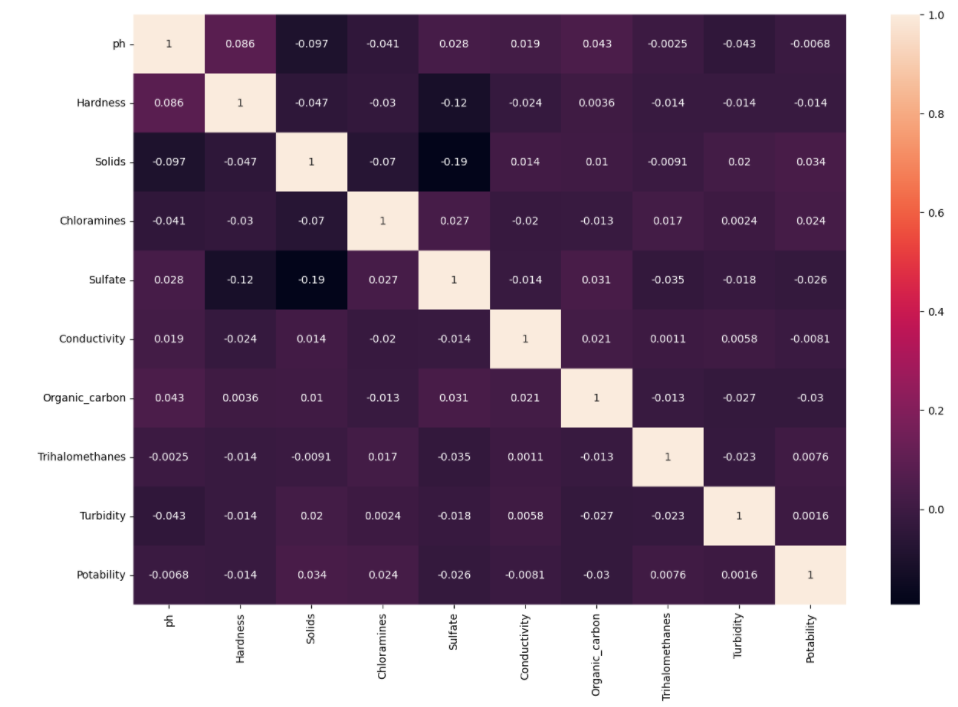


Рисунок 11. Тепловая карта матрицы корреляции

**Этап 4. Моделирование и прогнозирование**

Данные на обучающую и тестовую выборки были разбиты выше.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=159)

Оценим значимость признаков:

# Оценим важность каждого столбца

X = X\_train\_norm.values

y = y\_train\_norm.values

feature\_names = X\_train\_norm.columns.tolist() # Имена столбцов в датасете.

# Важность параметров

model = ensemble.RandomForestClassifier(n\_estimators=100,

criterion="entropy", random\_state=159)

model.fit(X,y)

importances = model.feature\_importances\_ # Это оценка, присваиваемая функциям модели машинного обучения,которая определяет, насколько «важной» является функция для прогноза модели.

# Создаём датафрейм

dtf\_importances = pd.DataFrame({"IMPORTANCE":importances,

"VARIABLE":feature\_names}).sort\_values("IMPORTANCE",

ascending=False)

dtf\_importances['cumsum'] = dtf\_importances['IMPORTANCE'].cumsum(axis=0)

dtf\_importances = dtf\_importances.set\_index("VARIABLE")

# Графики

fig, ax = plt.subplots(nrows=1, ncols=2, sharex=False, sharey=False,figsize=(12,6),dpi=100) # Распаковываем кортеж в переменные fig и ax.

fig.suptitle("Features Importance", fontsize=20)

ax[0].title.set\_text('Переменные') # Подпишем график

dtf\_importances[["IMPORTANCE"]].sort\_values(by="IMPORTANCE").plot(

kind="barh", legend=False, ax=ax[0]).grid(axis="x")

ax[0].set(ylabel="")

ax[1].title.set\_text('Накопление') # Подпишем график

dtf\_importances[["cumsum"]].plot(kind="line", linewidth=4,

legend=False, ax=ax[1])

ax[1].set(xlabel="", xticks=np.arange(len(dtf\_importances)),

xticklabels=dtf\_importances.index)

plt.xticks(rotation=70)

plt.grid(axis='both') # Настроим сетку.

plt.show()

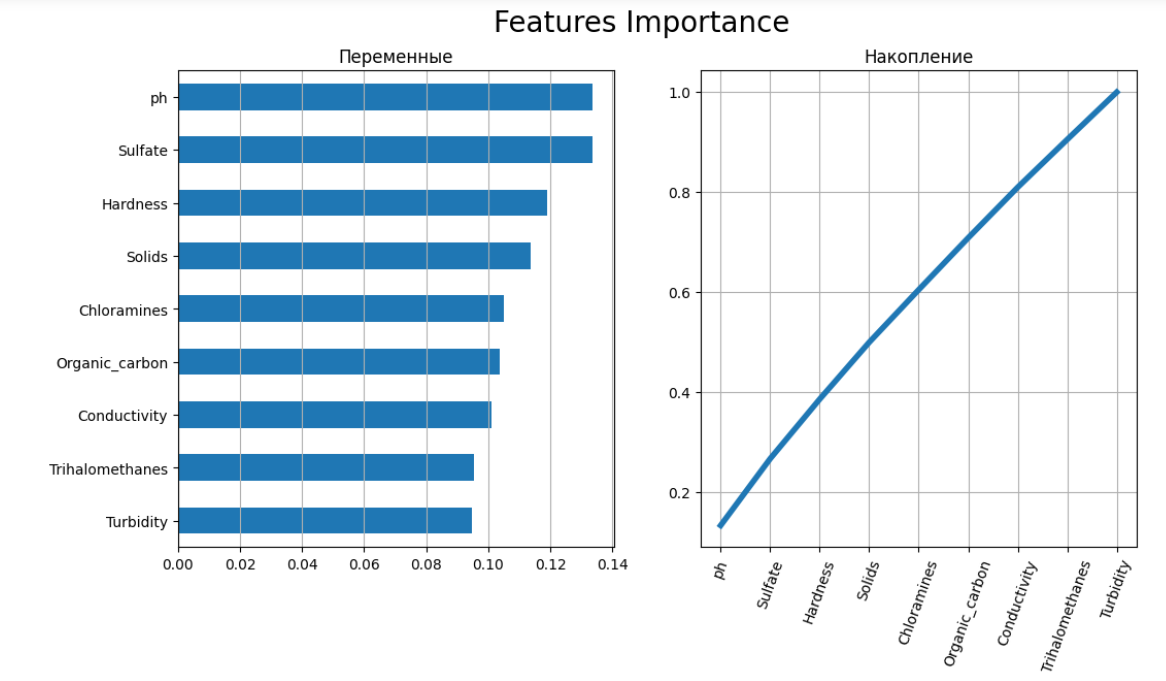


Рисунок 12. Значимость признаков в виде гистограммы и кривой накопления

По гистограмме видно, что все признаки вносят примерно одинаковый вклад, поэтому нет смысла удалять какой-либо признак.

На обучающей выборке построим несколько моделей бинарной классификации: логистическая регрессия(LR), линейный дискриминантный анализ(LDA), KNN, CART, наивная байесовская классификация(NB) и метод опорных векторов(SVC). Выведем усреднённые метрики для каждой модели:

# Создаем лист для тех моделей, которые будем изучать

models = []

models.append(('LR', LogisticRegression(solver='lbfgs', max\_iter=1000)))

models.append(('LDA', LinearDiscriminantAnalysis()))

models.append(('KNN', KNeighborsClassifier()))

models.append(('CART', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVC', SVC(gamma='auto')))

# оцениваем их метрики

results = []

model\_names = []

for name, model in models:

kfold = StratifiedKFold(n\_splits=10, random\_state=159, shuffle=True)

cv\_results = cross\_val\_score(model, X\_train\_norm, y\_train\_norm, cv=kfold, scoring='accuracy')

results.append(cv\_results)

model\_names.append(name)

print('%s: %f (%f)' % (name, cv\_results.mean(), cv\_results.std()))

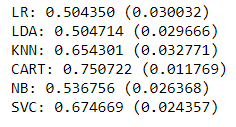


Рисунок 12. Усреднённые метрики для каждой модели

Лучшие метрики показали модели CART и SVC. Построим сначала модель CART. Выберем гиперпараметры.

tree = DecisionTreeClassifier(random\_state=159)

# Задаем диапазоны гиперпараметров для подбора

param\_grid = {

'criterion': ['gini', 'entropy'],

'max\_depth': [None, 5, 10, 15, 20],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

'max\_features': [None, 'sqrt', 'log2']

}

grid\_model = GridSearchCV(estimator=tree, param\_grid=param\_grid, scoring='accuracy', cv = 5)

grid\_model.fit(X\_train\_norm, y\_train\_norm)

grid\_model.best\_params\_

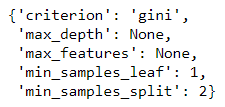


Рисунок 13. Гиперпараметры для модели CART

Выводим кривые обучения:

from sklearn.model\_selection import learning\_curve

train\_sizes, train\_scores, test\_scores = learning\_curve(

grid\_model.best\_estimator\_,

X\_train\_norm,

y\_train\_norm,

cv=kfold,

scoring='accuracy',

random\_state=159

)

plt.plot(train\_sizes, train\_scores.mean(axis=1), label='Train Accuracy')

plt.plot(train\_sizes, test\_scores.mean(axis=1), label='Validation Accuracy')

plt.legend()

plt.show()

y\_pred = grid\_model.best\_estimator\_.predict(X\_test\_norm)

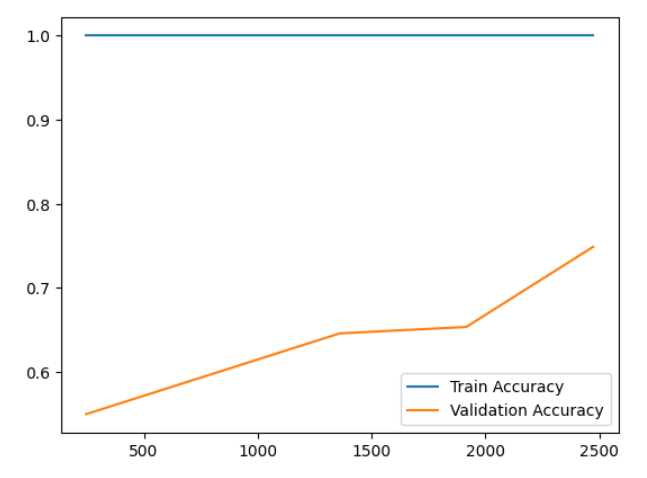


Рисунок 14. Кривые обучения

По графику видно, что accuracy обучающей выборки не меняется и равна 1, значит наша модель переобучилась. Это связано с тем, что максимальная глубина дерева при построении модели None (строит максимально возможную глубину).

Построим модель, ограничив глубину дерева вручную:

# Настройка модели с параметрами, минимизирующими переобучение

model = DecisionTreeClassifier(

criterion='entropy',

max\_depth=10, # Ограничение глубины дерева

min\_samples\_split=10, # Минимальное количество образцов для разбиения узла

min\_samples\_leaf=5, # Минимальное количество образцов в листе

random\_state=159 # Для воспроизводимости

)

Выводим кривые обучения:

# Обучение модели

model.fit(X\_train\_norm, y\_train\_norm)

# Оценка модели на тестовой выборке

y\_pred = model.predict(X\_test\_norm)

train\_sizes, train\_scores, test\_scores = learning\_curve(

#DecisionTreeClassifier(max\_depth=8, random\_state=159),

model,

X\_train\_norm,

y\_train\_norm,

cv=kfold,

scoring='accuracy',

random\_state=159

)

plt.plot(train\_sizes, train\_scores.mean(axis=1), label='Train Accuracy')

plt.plot(train\_sizes, test\_scores.mean(axis=1), label='Validation Accuracy')

plt.legend()

plt.show()

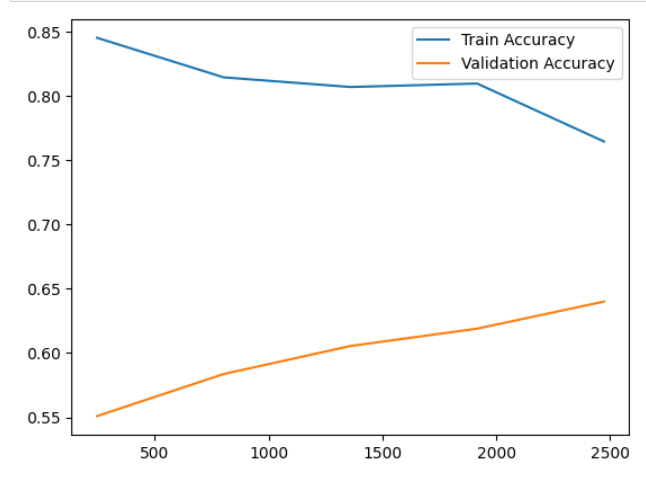


Рисунок 15. Кривые обучения

Видно, что кривые сходятся, значит модель не переобучена.

Выведем матрицу ошибок:

# Проверим, насколько наша модель хороша

model\_matrix = confusion\_matrix(y\_test\_norm, y\_pred, labels = [0,1])

ConfusionMatrixDisplay(confusion\_matrix = model\_matrix).plot()

plt.show()

matrix = classification\_report(y\_test\_norm, y\_pred)

print(matrix)

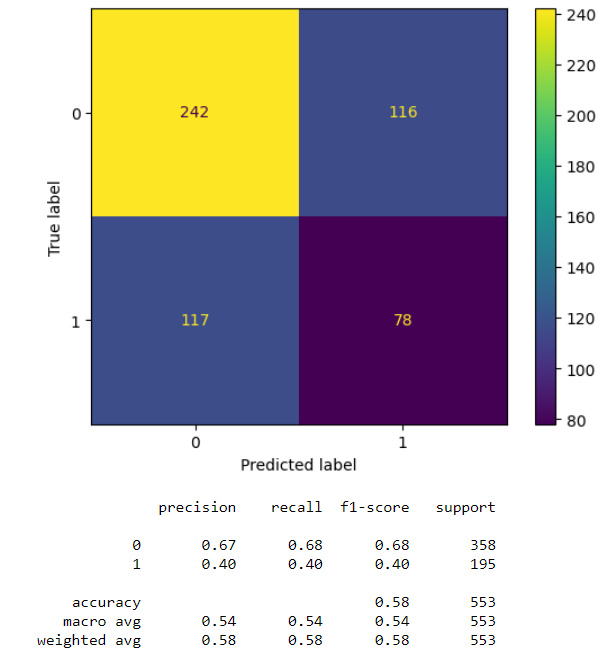


Рисунок 16. Матрица ошибок для модели CART

Accuracy = 0.58, что является достаточно средним показателем.

Построим теперь модель SVC:

# Выберем для модели SVC параметры kernel и C

svc = SVC()

grid\_model = GridSearchCV(svc, {'kernel': ['linear', 'poly', 'rbf'], 'C': [1, 2]}, cv = 5)

grid\_model.fit(X\_train\_norm, y\_train\_norm)

grid\_model.best\_params\_



Рисунок 17. Гиперпараметры для модели SVC

Обучим модель на лучших параметрах и на обучающем наборе данных:

svc = SVC(kernel=grid\_model.best\_params\_['kernel'], C=grid\_model.best\_params\_['C'], probability=True, random\_state=159)

svc.fit(X\_train\_norm, y\_train\_norm)

y\_pred = svc.predict(X\_test\_norm)

Построим матрицу ошибок:

# Проверим, насколько наша модель хороша

model\_matrix = confusion\_matrix(y\_test\_norm, y\_pred, labels = [1,0])

ConfusionMatrixDisplay(confusion\_matrix = model\_matrix).plot()

plt.show()

matrix = classification\_report(y\_test\_norm, y\_pred)

print(matrix)

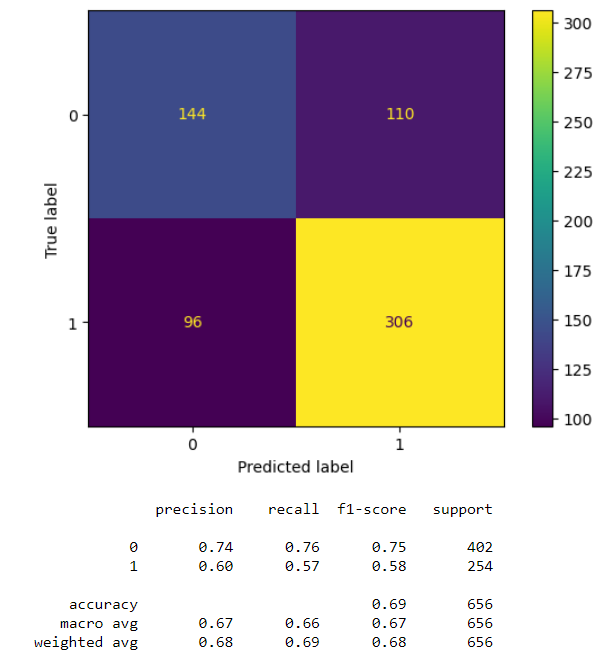


Рисунок 18. Матрица ошибок для модели SVC

Accuracy = 0.69, что является достаточно неплохим показателем.

Попробуем изменить значение Threshold:

# Построим графики значений отсечки для каждой метрики

# Значения метрик для различных пороговых значений вхождения в класс

fig, ax = plt.subplots(nrows=1, ncols=1,figsize=(10,5),dpi=100)

dic\_scores = {'accuracy':[], 'precision':[], 'recall':[], 'f1':[]}

XX\_train, XX\_test, yy\_train, yy\_test = model\_selection.train\_test\_split(X\_train\_norm, y\_train\_norm, test\_size=0.2, random\_state=159)

predicted\_prob = model.fit(XX\_train, yy\_train).predict\_proba(XX\_test)[:,1]

thresholds = []

for threshold in np.arange(0.1, 1, step=0.1):

predicted = (predicted\_prob > threshold)

thresholds.append(threshold)

dic\_scores["accuracy"].append(metrics.accuracy\_score(yy\_test, predicted))

dic\_scores["precision"].append(metrics.precision\_score(yy\_test, predicted, average='macro', zero\_division=1))

dic\_scores["recall"].append(metrics.recall\_score(yy\_test, predicted, average='macro'))

dic\_scores["f1"].append(metrics.f1\_score(yy\_test, predicted, average='macro'))

# Строим графики

dtf\_scores = pd.DataFrame(dic\_scores).set\_index(pd.Index(thresholds))

ax.set(xlabel='Threshold', ylabel="Score", title="Выбор порога")

dtf\_scores.plot(ax=ax)

plt.show()

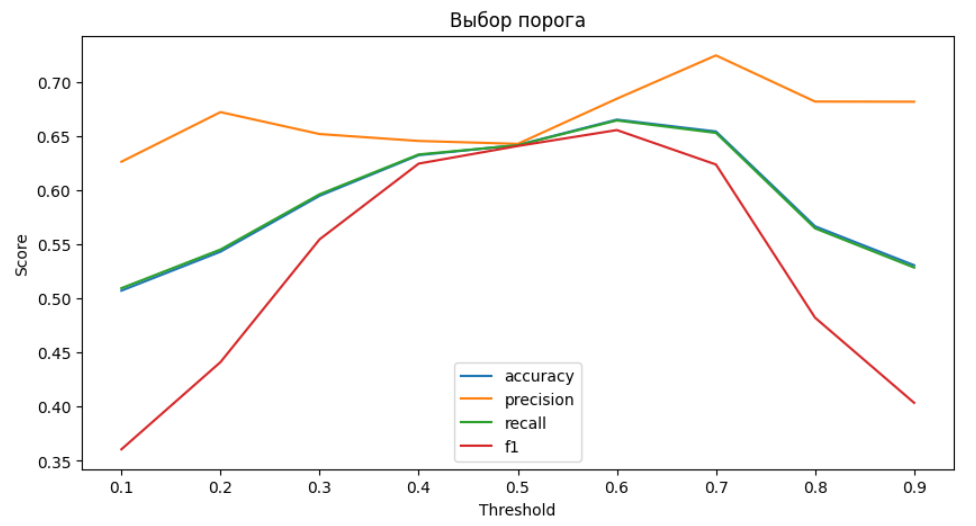


Рисунок 19. Значение Threshold для разных метрик

Лучшие метрики получились при threshold = 0.5. Теперь всё что меньше 0.5 это 0, а всё, что больше 0.5 это 1.

# Обучаем модель

model = SVC(kernel=grid\_model.best\_params\_['kernel'], C=grid\_model.best\_params\_['C'], probability=True, random\_state=159) # Используем probability=True для доступа к вероятностям

model.fit(X\_train\_norm, y\_train\_norm)

# Получаем вероятности принадлежности к классам

y\_pred = model.predict\_proba(X\_test\_norm)

# Задаем порог, например, 0.5 для положительного класса

threshold = 0.5

predictions = (y\_pred[:, 1] >= threshold).astype(int)

Выведем матрицу ошибок:

# Проверим, насколько наша модель хороша

model\_matrix = confusion\_matrix(y\_test\_norm, predictions, labels = [1,0])

ConfusionMatrixDisplay(confusion\_matrix = model\_matrix).plot()

plt.show()

matrix = classification\_report(y\_test\_norm, predictions)

print(matrix)

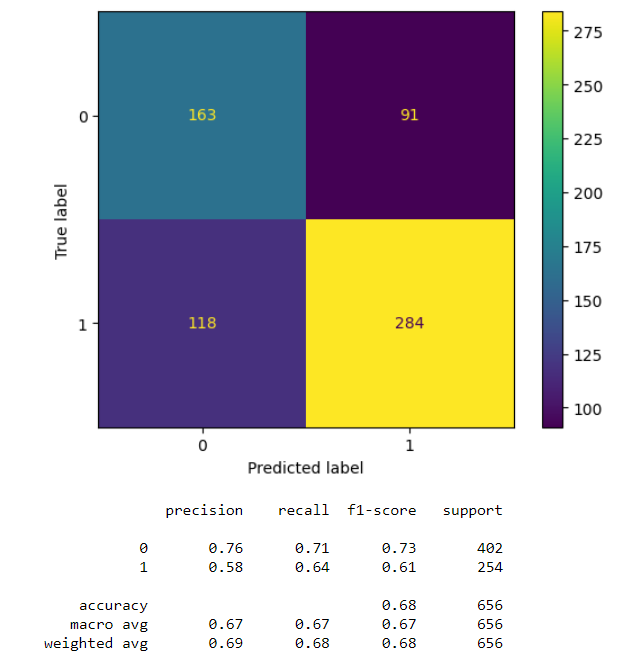


Рисунок 20. Матрица ошибок для модели SVC с threshold = 0.5

**Заключение**

На основе проведённого анализа факторов, влияющих на качество воды, можно сделать вывод, что поставленная цель работы была достигнута. Построенная модель бинарной классификации успешно определяет пригодность воды для питья на основе её физико-химических характеристик.

В процессе выполнения работы была подтверждена гипотеза о том, что качество воды зависит от таких параметров, как кислотно-щелочной баланс (pH), твёрдость воды (Hardness), содержание твердых веществ (Solids), хлораминов (Chloramines), сульфатов (Sulfate), проводимости (Conductivity), органического углерода (Organic Carbon), тригалометанов (Trihalomethanes) и мутности (Turbidity).

Для достижения цели были решены следующие задачи:

Выполнен анализ проблемы и обоснована её актуальность, связанная с важностью качества воды для здоровья населения и экологии.

Осуществлена загрузка данных, проведена подготовка, включая устранение пропусков значений с использованием модели RandomForestRegressor.

Проведен предварительный анализ данных, выявлены выбросы и выполнена их обработка, что улучшило качество модели.

Проведен корреляционный анализ, который показал слабую линейную связь между отдельными характеристиками и целевой переменной.

Построены и протестированы модели машинного обучения, включая DecisionTreeClassifier, SVC.

Подобраны наилучшие гиперпараметры для моделей, что позволило достичь высокой точности.

Проведен анализ зависимости метрик от порога классификации и оптимизировано значение порога для снижения ошибок предсказания классов.

Наилучшие результаты показала модель SVC, которая достигла точности (accuracy) в 69% при оптимальных гиперпараметрах. Для снижения переобучения при построении моделей использовались ограничения глубины деревьев решений, выборка сбалансированных данных с применением метода RandomOverSampler, а также нормализация признаков.

Таким образом, разработанная модель может быть использована для прогнозирования качества воды и оценки её пригодности для питья. Это решение может найти применение в задачах мониторинга состояния водных ресурсов, экологического контроля и принятия решений в области водоочистки.

**Список литературы**

1. Всемирная организация здравоохранения (ВОЗ). "Руководство по качеству питьевой воды". Женева, 2017.
2. Российская ассоциация водоснабжения и водоотведения. "Обеспечение качества воды в системах водоснабжения". Сборник научных статей, 2020.
3. Евстигнеев В.И., Евстигнеев Р.В. "Основы корреляционного анализа и регрессионного анализа". Москва: Юрайт, 2020.