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

по курсу «Технологии машинного обучения и нейросети для решения прикладных задач»

**ИССЛЕДОВАНИЕ НА ПРЕДМЕТ ВЫЯВЛЕНИЯ ЗНАНИЙ НЕЙРОСЕТЕВОЙ СИСТЕМЫ ПРОГНОЗИРОВАНИЯ КАЧЕСТВА ПРЕФОРМЫ-СТЕРЖНЯ НА ОСНОВЕ ДАННЫХ О ТОЛЩИНЕ ЛЕГИРОВАННОГО СЛОЯ, ПРЕДНАЗНАЧЕННОЙ ДЛЯ ИЗГОТОВЛЕНИЯ ОПТОВОЛОКНА**

Разработчики проекта:

Лешехва Екатерина,

Трефилов Владимир

Пермь, 2023

ОГЛАВЛЕНИЕ

[ПАСПОРТ ПРОЕКТА 3](#_Toc185847573)

[СОДЕРЖАНИЕ ПРОЕКТА 4](#_Toc185847574)

[Анализ проблемы исследования 4](#_Toc185847575)

[Реализация проекта 9](#_Toc185847576)

[Этап 1. Подготовка данных к анализу 9](#_Toc185847577)

[Этап 2. Корреляционный анализ 11](#_Toc185847578)

[Этап 3. Моделирование и прогнозирование 15](#_Toc185847579)

[Построение модели прогнозирования для станка 1 15](#_Toc185847580)

[Построение модели прогнозирования для станка 2 16](#_Toc185847581)

[Заключение 19](#_Toc185847582)

[Список использованных источников и литературы 20](#_Toc185847583)

[Приложения 22](#_Toc185847584)

# ПАСПОРТ ПРОЕКТА

**Название проекта:**исследование на предмет выявления знаний нейросетевой системы прогнозирования качества преформы-стержня на основе данных о толщине легированного слоя, предназначенной для изготовления оптоволокна.

**Сведения об авторах:**Лешехва Екатерина, Трефилов Владимир

**Цель:**создать нейросетевую систему прогнозирования качества преформы-стержня на основе анализа данных о толщине легированного слоя, с целью повышения продуктивности производственного процесса изготовления оптоволокна.

**Задачи:**

1. Выполнить анализ проблемы, обосновать ее актуальность.
2. Осуществить загрузку данных и подготовку их к анализу.
3. Исследовать собранные данные для выявления закономерностей и зависимостей между факторами и гладкостью.
4. Выбрать и создать релевантные признаки для модели классификации, которые наиболее сильно влияют на гладкость преформы-стержня.
5. Разработать и обучить модель классификации, предварительно выбрав наилучшую.
6. Выполнить интерпретацию полученных результатов и сделать выводы о достижении цели.

**Краткое описание проекта:**

Требуется проанализировать данные о процессе производства преформы-стержня и определить, существует ли зависимость качества преформы-стержня, а именно его гладкости от имеющихся в наборе данных факторных переменных. Дать интерпретацию полученным результатам. Сделать выводы.

**Конкретные ожидаемые результаты:**

Построенная модель классификации и рекомендации по ее использованию.

# СОДЕРЖАНИЕ ПРОЕКТА

## Анализ проблемы исследования

Оптоволоконные линии связи сегодня играют критическую роль в глобальной инфраструктуре передачи данных. Их способность передавать информацию на огромные расстояния с высокой скоростью и низкими потерями делает их незаменимыми для интернета, телекоммуникаций, а также в различных областях, требующих высокоскоростной передачи данных, таких как медицина, научные исследования и промышленность. Высокое качество оптоволокна напрямую влияет на эффективность и надежность этих систем. Любое ухудшение качества может привести к существенным потерям данных, снижению скорости передачи и увеличению стоимости обслуживания.

Производство оптоволокна – сложный технологический процесс, включающий в себя несколько этапов. Ключевой из них – модификация стеклянного сердечника волокна путем легирования, то есть добавления примесей, которые изменяют его оптические свойства. Этот процесс осуществляется на специализированных станках, таких как машины модификации химического осаждения из паровой фазы (MCVD). Эти станки позволяют с высокой точностью контролировать параметры процесса, но даже незначительные отклонения могут привести к существенным изменениям характеристик готового волокна.

Легированный слой в сердцевине оптоволокна – это область с измененным химическим составом, которая отвечает за передачу оптического сигнала. В зависимости от типа легирующих примесей (например, германия, бора, фосфора) изменяется показатель преломления света в этом слое. Это изменение показателя преломления и создает условия для полного внутреннего отражения светового сигнала внутри волокна, обеспечивая эффективную передачу информации. Качество легированного слоя – определяющий фактор эффективности всего оптоволокна.

Толщина легированного слоя – один из важнейших параметров, влияющих на качество оптоволокна. Слишком тонкий слой может приводить к значительным потерям оптического сигнала за счет несовершенства процесса полного внутреннего отражения и рассеяния света на границах раздела. Это снижает дальность передачи сигнала и ограничивает пропускную способность линии связи. С другой стороны, чрезмерно толстый слой может ухудшать другие параметры волокна, такие как дисперсия, которая искажает сигнал и снижает его качество. Оптимальная толщина слоя – результат компромисса между этими факторами.

Помимо толщины, важнейшим параметром является равномерность легированного слоя. Неравномерность слоя может вызвать нестабильность показателя преломления по длине волокна, что приводит к рассеянию света и снижению качества передачи сигнала. Наличие неровностей и неоднородностей в легированном слое также может негативно влиять на прочность и надежность оптоволокна, делая его более уязвимым к механическим повреждениям и внешним воздействиям.

Для обеспечения высокого качества легированного слоя необходимо контролировать множество параметров процесса производства. К ним относятся характеристики используемого оборудования, параметры технологического процесса (температура, давление, скорость потока газов), качество используемых исходных материалов, а также квалификация персонала. Даже небольшие изменения в этих параметрах могут оказывать существенное влияние на конечный результат.

В данном исследовании анализируется широкий спектр факторов, влияющих на качество легированного слоя оптоволокна, используя обширный набор данных, включающий в себя параметры процесса производства, характеристики оборудования и качество готовой продукции. Это позволяет выявить наиболее значимые факторы, которые определяют качество легированного слоя и разработать модель для прогнозирования качества продукции.

Цель исследования - разработка точной модели, которая бы предсказывала качество легированного слоя на основе входных параметров. Это позволит оптимизировать процесс производства, минимизировать брак и повысить эффективность производства оптоволокна. Такая модель может быть использована для автоматического контроля качества в режиме реального времени и для принятия обоснованных управленческих решений.

Полученные результаты исследования будут иметь значительный практический эффект, позволяя производителям оптоволокна улучшить качество своей продукции и снизить затраты. Это особенно актуально в условиях растущего спроса на высококачественное оптоволокно для быстро развивающейся индустрии связи и других сфер применения. Понимание взаимосвязи между параметрами процесса и качеством легированного слоя является ключом к созданию более эффективных и надежных технологий производства оптоволокна.

Задачи:

1. Выполнить анализ проблемы, обосновать ее актуальность.
2. Осуществить загрузку данных и подготовку их к анализу количественными методами, включая устранение пропущенных значений.
3. Исследовать собранные данные для выявления закономерностей и зависимостей между факторами и гладкостью.
4. Выбрать и создать релевантные признаки для модели классификации, которые наиболее сильно влияют на гладкость преформы-стержня.
5. Разработать и обучить модель классификации, предварительно выбрав наилучшую.
6. Выполнить интерпретацию полученных результатов и сделать выводы о достижении цели.

Исходные данные

В работе анализируется список факторов, участвующих в процессе производства оптоволоконных изделий на предприятии ПНППК.

Список колонок анализируемого набора данных:

1. **Type of machine** – Тип машины (станка)
2. **Length of the preform-rod after MCVD, mm** – Длина изделия
3. **Run length, mm** – Точка отреза изделия от станка по левому краю
4. **Position from the beginning of deposition (Ex.), mm** – Максимальная ширина для преформы
5. **Operator 1** – Первый сотрудник
6. **Operator 2** – Второй сотрудник
7. **Pipe diameter (max.), mm** – Максимальный диаметр трубы
8. **Pipe diameter (min.), mm** – Минимальный диаметр трубы
9. **Wall thickness, mm (max.)** – Максимальная толщина стенки
10. **Wall thickness, mm (min.)** – Минимальная толщина стенки
11. **Ovality, mm (max.)** – Максимальная овальность
12. **Ovality, mm (min.)** – Минимальная овальность
13. **Air flow rate 3B, m/s** – Расход воздуха (всасываемость вытяжки)
14. **Dew point O2, °C** – Точка росы кислорода
15. **Dew point N2, °C** – Точка росы азота
16. **Dew point He, °С** – Точка росы гелия
17. **Amount of BCl3 before the process** – Остаток трихлорида бора
18. **Amount of SiCl4 before the process** – Остаток тетрахлорида кремния
19. **Amount of POCl3 before the process** – Остаток оксихлорида фосфора
20. **Process pipe on the headstock side** – Технологическая труба на передней бабке (кол-во использований)
21. **Process pipe on the tailstock side** – Технологическая труба на задней бабке (кол-во использований)
22. **Pressure during leak testing** - Давление
23. **Number of passes Sedcevina BP** – Количество проходов горелкой при оснащении фосфидом бора
24. **Number of passes Sedcevina B** – Количество проходов горелкой при оснащении бором
25. **SICL in the recipe** - тетрахлорида кремния в рецепте
26. **POCL in the recipe** – оксихлорида фосфора в рецепте
27. **BCL in the recipe** – трихлорида бора в рецепте
28. **Position from tailstock** – Точка измерения диаметра слоя
29. **Core diameter (entire alloyed area), mm** – Диаметр слоя

**Гипотеза:** предполагается, что если создать нейросетевую модель прогнозирования параметра качества преформы-стрежня на основе данных о толщине легированного слоя, то повысится продуктивность производственного процесса изготовления оптоволокна.

## Реализация проекта

### Этап 1. Подготовка данных к анализу

Подключаем необходимые библиотеки.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

sns.set()

Загружаем данные исходного датасета в переменную data.

data = pd.read\_csv('/content/drive/MyDrive/MO Leshekhva/Fiberoptic/Легированный слой.csv') # считали исходный датасет

data.head()

data.describe() # статистические данные о датасете

data.info() # данные колонок

На рисунке 1 представлены статистические данные о датасете data.

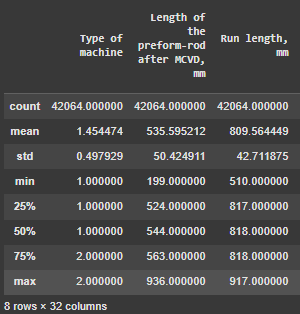


Рисунок 1 – Статистические данные data

Было обнаружено, что максимальная длина преформы-стрежня достигает 936 мм, а минимальная 199, что не предусматривает стандартная рецептура по изготовлению трубы. Было принято решение удалить изделие длиной более 650 мм и менее 350 мм, т.к. трубы нетипичной длины обычно не используются по причине высоких рисков неправильного хода процесса. Такие случаи являются исключениями и очень редки.

data = data.loc[(data['Length of the preform-rod after MCVD, mm'] >= 350) & (data['Length of the preform-rod after MCVD, mm'] <= 650)]

Каждая строка является результатом выполненного замера толщины легированного слоя, начиная с 20 сантиметра от начала стержня с шагом 20 см. Количество замеров различно. То есть в датасете хранятся дубликаты стержней, если убрать столбец «Core diameter (entire alloyed area), mm». Чтобы классифицировать стержни, необходимо выделить категории гладкости и создать новый датасет, «схлопнув» дубликаты воедино. Для определения категории необходимо вычислить разницу между максимальной и минимальной толщиной легированного слоя, исключая значения на первом и последнем замере. Определено 5 категорий:

* + - 1. Отлично. Класс 5, разница меньше 0.1 мм.
      2. Хорошо. Класс 4, разница больше либо равна 0.1 мм и меньше 0.15 мм.
      3. Среднее. Класс 3, разница больше либо равна 0.15 мм и меньше 0.2 мм.
      4. Плохо. Класс 2, разница больше либо равна 0.2 мм и меньше 0.3 мм.
      5. Не используется. Класс 1, разница больше либо равна 0,3 мм.

Функция av\_dev\_sm (data, data\_new) определяет категорию стержня и перезаписывает сведения о стержне в датасет data\_new, структура которой дополнена значениями трех полей «Index» - идентификатор стержня, «Average deviation» - разница между максимальной и минимальной толщиной стержня (среднее отклонение), «Smooth» - гладкость, категория стержня.

### Этап 2. Корреляционный анализ

На рисунке 2 представлена матрица корреляционной зависимости по датасету data\_new.

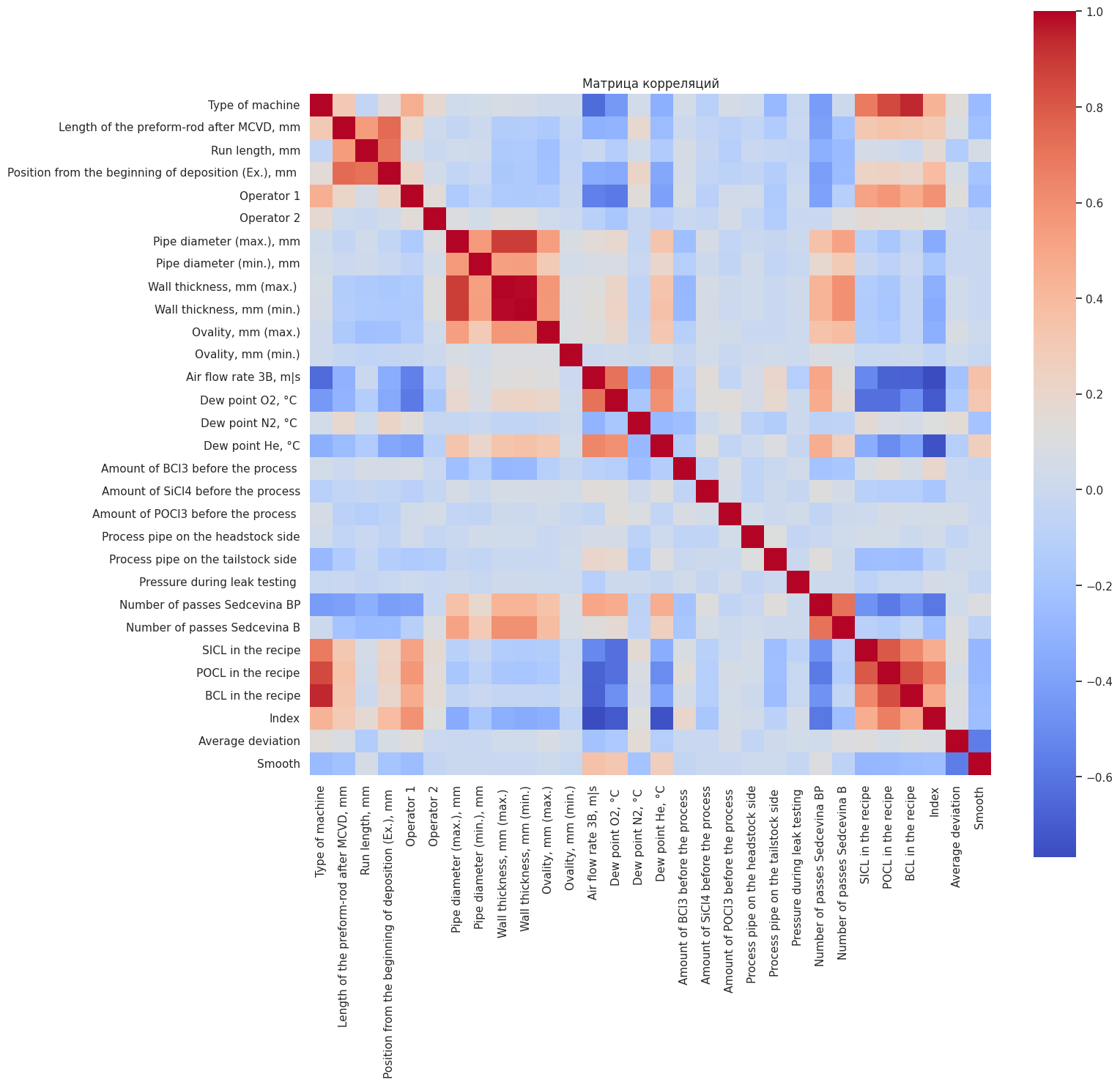


Рисунок 2 – Матрица корреляции по data\_new

Определены корреляционные зависимости:

* Среднее отклонение и категория стержня;
* BCl в рецепте и тип машины;
* PoCl в рецепте и тип машины;
* SiCl в рецепте и тип машины;
* количество проходов при осаждении трубы BP и BCl в рецепте;
* количество проходов при осаждении трубы BP и PoCl в рецепте;
* количество проходов при осаждении трубы BP и SiCl в рецепте;
* скорость воздушного потока и BCl в рецепте;
* скорость воздушного потока и PoCl в рецепте;
* скорость воздушного потока и SiCl в рецепте.

Можно сделать вывод, что скорость воздушного потока и количество проходов зависят от типа машины, поэтому следует разделить датасет на 2 по типу машины (data\_new\_classification1 и data\_new\_classification2).

На рисунке 3 представлена визуализация зависимости SiCl, PoCl, BCl в рецепте от толщины легированного слоя. Можно сделать вывод о том, что станок 1 (красный) использует химию для изготовления преформы намного меньше, чем станок 2 (оранжевый), что является еще одной причиной разделить датасет на 2. Также видно, что значения толщины легированного слоя колеблются практически в одном и том же диапазоне от 5.5 мм до 7.5 мм.

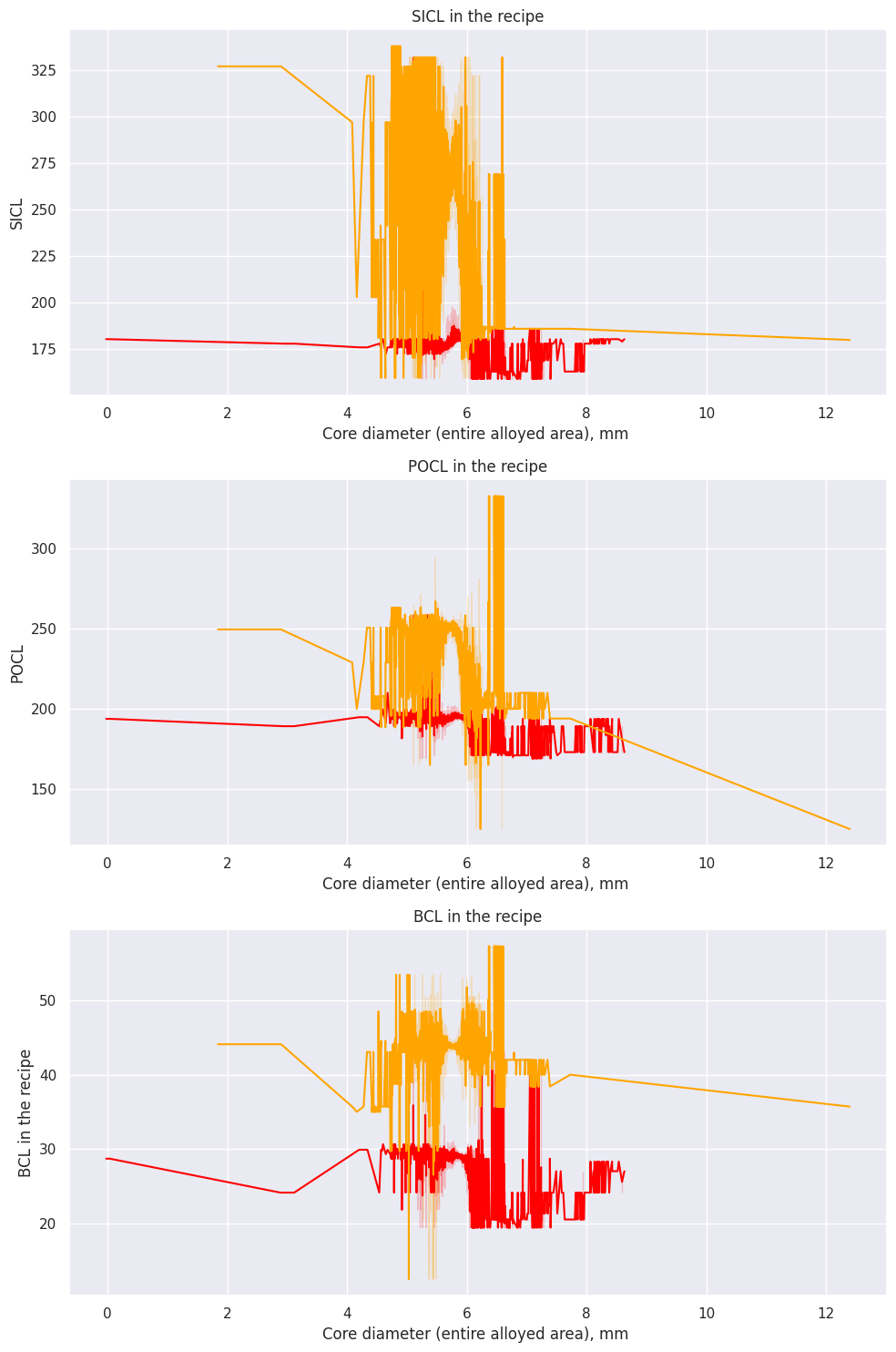


Рисунок 3 – Визуализация зависимости химии в рецепте от толщины слоя

Матрица корреляции по датасету data\_new\_classification1 отражает следующие зависимости (см. Рисунок 4):

* PoCl и BCL в рецепте от собственных характеристик стержня;
* Количество проходов BP и B от собственных характеристик стержня.
* Количество проходов BP и B от PoCl и BCL в рецепте.

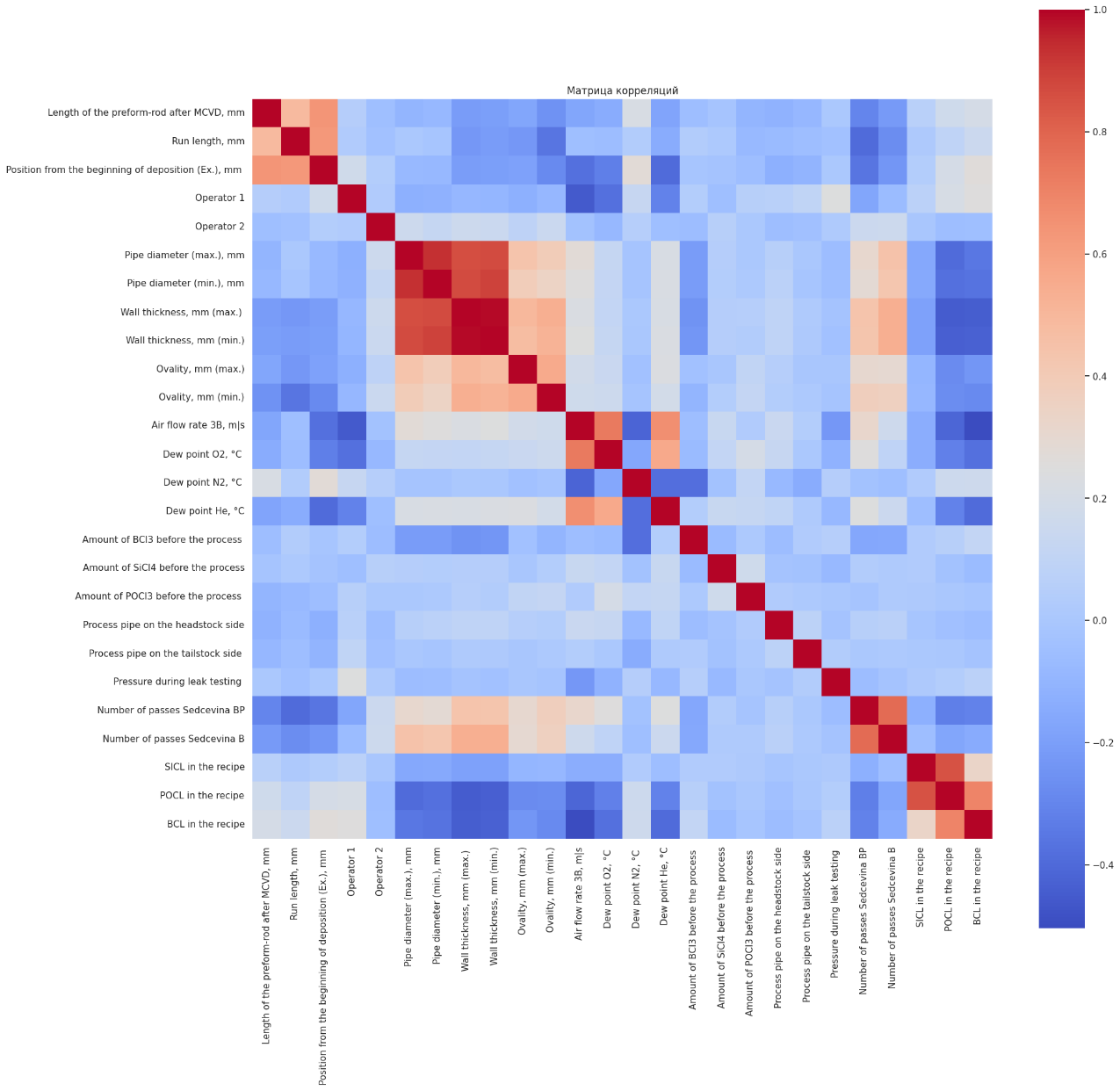


Рисунок 4 – Матрица корреляции по data\_new\_classification1

Матрица корреляции по датасету data\_new\_classification2 отражает следующие зависимости (см. Рисунок 5):

* Количество проходов BP и B от собственных характеристик стержня;
* SiCl и PoCl в рецепте от собственных характеристик стержня;
* Точка росы He и O2 от собственных характеристик стержня;
* SiCl и PoCl в рецепте от точки росы O2.

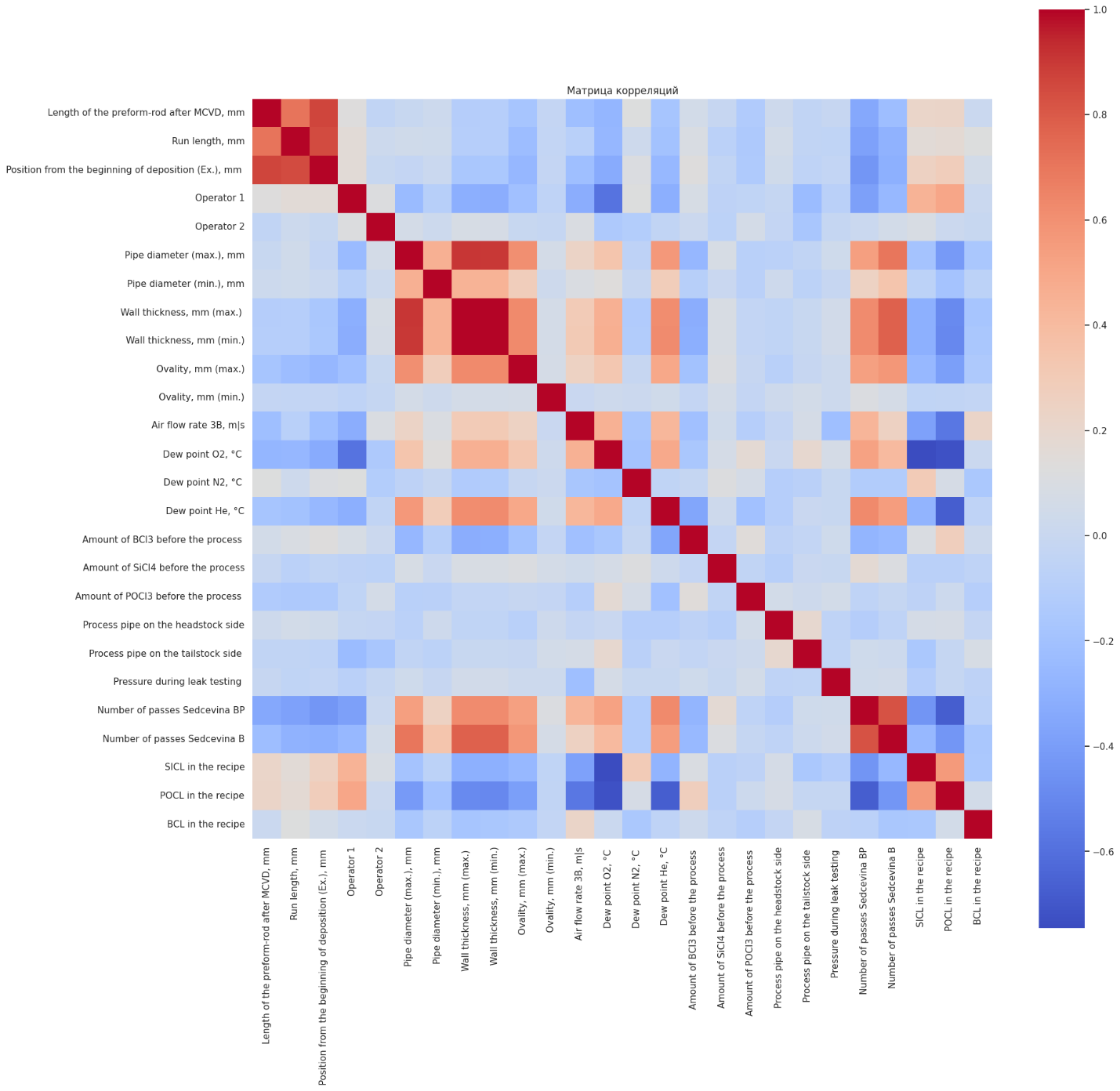


Рисунок 5 – Матрица корреляции для data\_new\_classification2

### Этап 3. Моделирование и прогнозирование

## Построение модели прогнозирования для станка 1

На рисунке 6 представлены уникальные значения категорий стержня и их количество строк в датасете data\_new\_classification1. Датасет не сбалансирован. Для балансировки используется метод SMOTE, результат которой представлен на рисунке 7.

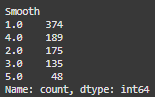


Рисунок 6 – Количество строк на каждую категорию по data\_new\_classification1 до балансировки

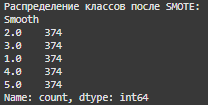


Рисунок 7 – Количество строк на каждую категорию по data\_new\_classification1 после балансировки

Разобьём данные на обучающую и тестовую выборки, используя соотношение 8 : 2.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_resampled, y\_resampled, test\_size=0.2, random\_state=42, stratify=y\_resampled)

На рисунке 8 представлен результат метрик моделей классификации. Было принято решение использовать метод опорных векторов SVC, т.к. наилучший показатель 79%.

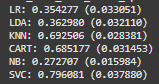


Рисунок 8 – Метрики моделей классификации

Обучаем модель SVC с GridSearchCV для подбора параметров.

param\_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001], 'kernel': ['rbf'

grid = GridSearchCV(SVC(), param\_grid, refit=True, verbose=2)

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

Выполняем предсказания для тестового набора данных X\_test с использованием модели, созданной с помощью функции grid.

grid\_predictions = grid.predict(X\_test)

Оцениваем качество модели после выполнения предсказания.

print(classification\_report(y\_test, grid\_predictions))

cm = confusion\_matrix(y\_test, grid\_predictions)

print(cm)

На рисунке 9 представлена матрица ошибок, отражающая качество классификационной модели. Можно сделать вывод, что модель преимущественно ошибается в прогнозировании 1 категории. Категория 5 имеет высокое количество правильных предсказаний, что указывает на хорошую работу модели.

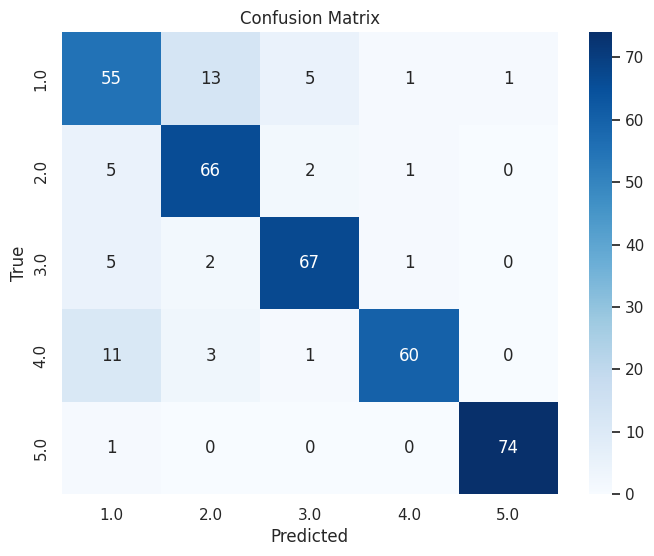


Рисунок 9 – Матрица ошибок модели SVC по data\_new\_classification1

## Построение модели прогнозирования для станка 2

На рисунке 10 представлены уникальные значения категорий стержня и их количество строк в датасете data\_new\_classification2. Датасет не сбалансирован. Для балансировки используется метод SMOTE, результат которой представлен на рисунке 11.

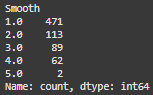


Рисунок 10 – Количество строк на каждую категорию по data\_new\_classification1 до балансировки

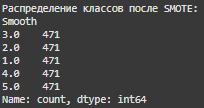


Рисунок 11 – Количество строк на каждую категорию по data\_new\_classification1 после балансировки

Разобьём данные на обучающую и тестовую выборки, используя соотношение 8 : 2.

X\_train2, X\_test2, y\_train2, y\_test2 = train\_test\_split(X\_resampled2, y\_resampled2, test\_size=0.2, random\_state=42, stratify=y\_resampled2).

На рисунке 12 представлен результат метрик моделей классификации. Было принято решение использовать метод опорных векторов SVC, т.к. наилучший показатель 94%.

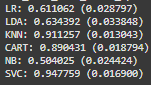


Рисунок 12 – Метрики моделей классификации

Обучаем модель SVC с GridSearchCV для подбора параметров.

param\_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001], 'kernel': ['rbf']} # пример сетки параметров

grid = GridSearchCV(SVC(), param\_grid, refit=True, verbose=2)

grid.fit(X\_train2, y\_train2)

Выполняем предсказания для тестового набора данных X\_test2 с использованием модели, созданной с помощью функции grid.

grid\_predictions = grid.predict(X\_test2)

Оцениваем качество модели после выполнения предсказания.

print(classification\_report(y\_test2, grid\_predictions))

cm = confusion\_matrix(y\_test2, grid\_predictions)

print(cm)

На рисунке 13 представлена матрица ошибок, отражающая качество классификационной модели. Для классов 1, 2, 4 и 5 наблюдается очень высокая доля правильных предсказаний, что говорит о том, что модель хорошо справляется с определением этих классов. Класс 1 имеет 3 ошибки, где 3 объекта были ошибочно классифицированы как класс 2. Это указывает на то, что модель может путать эти два класса. Класс 3 был ошибочно классифицирован 2 раза как класс 2 и 1 раз как класс 4. Это говорит о том, что модели стоит уделить больше внимания распознаванию этого класса.

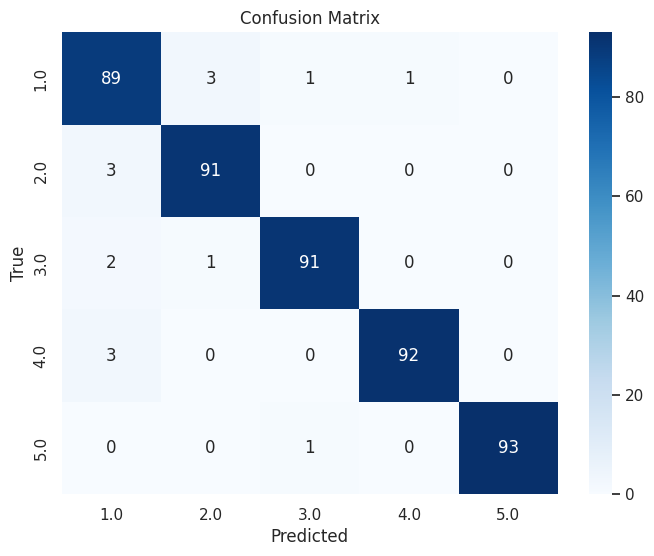


Рисунок 13 – Матрица ошибок модели SVC по data\_new\_classification2

## Заключение

Исследование на предмет выявления знаний нейросетевой системы прогнозирования качества преформы-стержня на основе данных о толщине легированного слоя, предназначенной для изготовления оптоволокна, представляет собой важный шаг в области оптимизации производственных процессов.

В ходе работы была построена модель классификации, которая позволяет прогнозировать качество (гладкость) преформы-стержня. Основные характеристики модели включают:

- Признаки. Модель базируется на разнообразных параметрах, включая характеристики трубы и химию рецептов, таких как PoCl, BCl, SiCl.

- Корреляционные зависимости. Комплексное исследование корреляционных зависимостей между признаками позволило более глубоко понять влияющие факторы на качество преформы, что может служить основой для дальнейших улучшений в технологии.

- Точность модели. Полученные результаты показывают высокую точность моделирования: 86% на датасете для первого станка и 96% на датасете для второго станка. Это указывает на то, что нейросетевая модель может быть эффективно использована для оценки и прогнозирования качества преформы.

Таким образом, проведенное исследование демонстрирует огромный потенциал использования нейросетевых алгоритмов в производственной сфере, позволяя не только оптимизировать процессы, но и повысить качество конечной продукции. Результаты могут также послужить основой для дальнейших исследований, направленных на улучшение параметров производства преформы-стержня и увеличение эффективности процесса изготовления оптоволокна.

## Список использованных источников и литературы

Бессмертный, И. А. Интеллектуальные системы : учебник и практикум для академического бакалавриата / И. А. Бессмертный, А. Б. Нугуманова, А. В. Платонов. — Москва : Издательство Юрайт, 2019. — 243 с. — (Бакалавр. Академический курс). — ISBN 978-5-534-01042-8. — Текст : электронный // ЭБС Юрайт [сайт]. (дата обращения: 01.10.2024).

Ясницкий, Л. Н. Интеллектуальные системы : учебник / Л. Н. Ясницкий. — 2-е изд. — Москва : Лаборатория знаний, 2020. — 222 c. — ISBN 978-5-00101-897-1. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. (дата обращения: 01.10.2024).

Сузи, Р. А. Язык программирования Python : учебное пособие / Р. А. Сузи. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. — 350 c. — ISBN 978-5-4497-0705-5. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. (дата обращения: 01.10.2024).

Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для среднего профессионального образования / Д. Ю. Федоров. — 4-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 214 с. — (Профессиональное образование). — ISBN 978-5-534-15731-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. (дата обращения: 01.10.2024).

EDA. Поисковый анализ данных на Python. [Электронный ресурс]. URL: https://translated.turbopages.org/proxy\_u/en-ru.ru.7ef26421-67684fec-482e8cf6-74722d776562/https/www.geeksforgeeks.org/exploratory-data-analysis-in-python/(дата обращения: 01.12.2024).

Статья «Технология производства оптического волокна». URL: <https://science.fandom.com/ru/wiki/Технология_производства_оптического_волокна> (дата обращения: 02.12.2024).

Базаров, Е.Н. Теоретические основы волоконно-оптической техники.: учеб. пособие для вузов / Е.Н. Базаров, В.Д. Бурков, А.Д. Шатров. – М.: МГУЛ, 2004. – 131 с (дата обращения: 02.12.2024)..

Бурков, В.Д. Физико-технологические основы волоконно-оптической техники /В.Д. Бурков, Г.А. Иванов. – М.: ГОУ ВПО МГУЛ, 2007. – 222 с (дата обращения: 02.12.2024)..

Никоноров, Н.В. Материалы и технологии волоконной оптики: специальные оптические волокна. Учебное пособие, курс лекций / Н.В. Никоноров, А.И. Сидоров. – СПб.: СПбГУ ИТМО, 2009. – 130 с (дата обращения: 02.12.2024).

Nagel, S.R. An Overview of the Modified Chemical Vapor Deposition (MCVD) Process and Performance / S.R. Nagel, J.B. Macchesney, K.L. Walker // IEEE journal of quantum electronics. — 1982. — Vol. MTT-30. — No. 4. — P. 305-322 (дата обращения: 02.12.2024).

# Приложения

Приложение 1

Программный код

# -\*- coding: utf-8 -\*-

"""Легированный слой 2.0.ipynb

Automatically generated by Colab.

Original file is located at

    https://colab.research.google.com/drive/12aBSdMpYfnCOHOFVYTHXFeaRxkfibkrK

"""

from google.colab import drive

drive.mount('/content/drive/')

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

sns.set()

"""# 1 Преобработка и анализ входных данных

"""

data = pd.read\_csv('/content/drive/MyDrive/MO Leshekhva/Fiberoptic/Легированный слой.csv') # считали исходный датасет

data.head()

data.describe() # статистические данные о датасете

data.info()

"""Вывод 1.1: было обнаружено, что максимальная длина преформы-стрежня достигает 936 мм, а минимальная 199, что не предусматривает стандартная рецептура по изготовлению трубы. Было принято решение удалить изделие длиной более 650 мм и менее 350 мм, т.к. трубы нетипичной длины обычно не используются по причине высоких рисков неправильного хода процесса.

Изделия такой длины исключения.

"""

data = data.loc[(data['Length of the preform-rod after MCVD, mm'] >= 350) & (data['Length of the preform-rod after MCVD, mm'] <= 650)]

data\_new = pd.read\_csv('/content/drive/MyDrive/MO Leshekhva/Fiberoptic/Легированный слой Новый.csv') # сосздаем пустой датасет с такой же структурой как у data

data\_new.head()

"""В датасете отображены замеры легированного слоя на нескольких участках преформы-стержня с шагом 20 см. Чтобы классифицировать преформы на классы 1-5, необходимо проверить значение разницы максимальной и минимальной толщины этого стержня, исключая первые и последний замеры.

Было принято решение написать функцию av\_dev\_sm, которая определяет класс преформы на основе данных о легированном слое.

Классы (категории по качеству)

5 - Отлично

4 - Хорошо

3 - Среднее

2 - Плохо

1 - Обычно не используются

"""

from sys import call\_tracing

def av\_dev\_sm(data, data\_new):

    index = 0

    while index < len(data) - 2:

        count = 0

        sum\_dev = 0

        maximum = float('-inf')  # Изменено на -бесконечность

        minimum = float('inf')   # Изменено на +бесконечность

#########  как только мы встречаем замер на 20 см, то это означает, что мы измеряем новый стержень, но количество замеров разное,

#########  поэтому используем цикk пока не 20 см ищем максимальную и минимальную толщину легированного слоя в конкретном стержне

#########  после вычисляем разницу между максимальной и минимальной толщиной и присваиваем переменной cat значение, где cat - категория стрежня по качеству

#########  заполняем поля датасета data\_new 'Index', 'Average deviation', 'Smooth'

        # Проверяем текущую строку на условие

        if data['Position from tailstock'].values[index] == 20.0:

            row = data.iloc[index]

            number = index

            index += 1

            # Обрабатываем остальные строки пока не встретим 20 снова

            while index < len(data) - 1 and data['Position from tailstock'].values[index] != 20.0:

                maximum = max(maximum, data['Core diameter (entire alloyed area), mm '].values[index])

                minimum = min(minimum, data['Core diameter (entire alloyed area), mm '].values[index])

                index += 1

            dev = maximum - minimum

            if dev < 0.1:

                cat = 5

            elif dev < 0.15:

                cat = 4

            elif dev < 0.2:

                cat = 3

            elif dev < 0.3:

                cat = 2

            else:

                cat = 1

        row['Index'] = number # идентификатор стержня будет равен номеру строки замера на 20 см

        row['Average deviation'] = dev # разница между максимальной и минимальной толщиной

        row['Smooth'] = cat # категория стержня

        data\_new.loc[len(data\_new)] = row

    return data\_new

data\_new = av\_dev\_sm(data, data\_new)

data\_new.head()

data\_new.describe()

#матрица корреляции для датасета data\_new

corr\_matrix = data\_new.corr()

plt.figure(figsize=(15,15))

sns.heatmap(corr\_matrix, annot=False, fmt='.2f',cmap='coolwarm', square=True)

plt.title('Матрица корреляций')

plt.show()

"""Корреляцонные зависимости

BCl в рецепте и тип машины

PoCl в рецепте и тип машины

SiCl в рецепте и тип машины

\*

Количество проходов при осаждении трубы BP и BCl в рецепте

Количество проходов при осаждении трубы BP и PoCl в рецепте

Количество проходов при осаждении трубы BP и SiCl в рецепте

\*

Скорость воздушного потока и BCl в рецепте

Скорость воздушного потока и PoCl в рецепте

Скорость воздушного потока и SiCl в рецепте

\*

Скорость воздушного потока и Количество проходов зависят от типа машины. Поэтому следует разделить датасет на 2 по типу машины

\*

# 2 Разделение датасета data\_new по типу станка

"""

# Commented out IPython magic to ensure Python compatibility.

import numpy as np

import pandas as pd

import seaborn as sns

sns.set\_palette('husl')

import matplotlib.pyplot as plt

# %matplotlib inline

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.metrics import accuracy\_score

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

data\_type1 = data.loc[data['Type of machine'] == 1]

data\_type2 = data.loc[data['Type of machine'] == 2]

import matplotlib.pyplot as plt

import seaborn as sns

# Создание figure и подграфиков

fig, axs = plt.subplots(3, 1, figsize=(10, 15))

# Первый график для SICL

sns.lineplot(data=data\_type1, x='Core diameter (entire alloyed area), mm ', y='SICL in the recipe', color='red', ax=axs[0])

sns.lineplot(data=data\_type2, x='Core diameter (entire alloyed area), mm ', y='SICL in the recipe', color='orange', ax=axs[0])

axs[0].set\_title('SICL in the recipe')

axs[0].set\_ylabel('SICL')

# Второй график для POCL

sns.lineplot(data=data\_type1, x='Core diameter (entire alloyed area), mm ', y='POCL in the recipe', color='red', ax=axs[1])

sns.lineplot(data=data\_type2, x='Core diameter (entire alloyed area), mm ', y='POCL in the recipe', color='orange', ax=axs[1])

axs[1].set\_title('POCL in the recipe')

axs[1].set\_ylabel('POCL')

# Третий график для BCL in the recipe

sns.lineplot(data=data\_type1, x='Core diameter (entire alloyed area), mm ', y='BCL in the recipe', color='red', ax=axs[2])

sns.lineplot(data=data\_type2, x='Core diameter (entire alloyed area), mm ', y='BCL in the recipe', color='orange', ax=axs[2])

axs[2].set\_title('BCL in the recipe')

axs[2].set\_ylabel('BCL in the recipe')

# Настройка общего оформления

plt.tight\_layout()

plt.show()

"""Чтобы классифицировать стержень по категориям качества, используя нейросетевую модель, необходимо удалить столбец 'Average deviation', т.к. от него напрямую зависит категория стержня (см. функцию av\_dev\_sm). Также удалим 'Index', т.к. это просто идентификатор, который ни на что не влияет."""

data\_new\_classification = data\_new.drop(['Index', 'Average deviation'], axis = 1) # будем использовать этот датасет для решения задачи классификации

"""Качество преформы-стержня может зависеть не только от характеристик трубы или химии, но и от типа станка.

Было принято решение разделить датасет на 2 по типу станка.

"""

data\_new\_classification1 = data\_new\_classification[data\_new\_classification['Type of machine'] == 1] # первый станок

data\_new\_classification2 = data\_new\_classification[data\_new\_classification['Type of machine'] == 2] # второй станок

#матрица корреляций для станка 1

matrix2 = data\_new\_classification1.drop(['Type of machine', 'Smooth'], axis=1)

corr\_matrix = matrix2.corr()

plt.figure(figsize=(20,20))

sns.heatmap(corr\_matrix, annot=False, fmt='.2f',cmap='coolwarm', square=True)

plt.title('Матрица корреляций')

plt.show()

#матрица корреляций для станка 2

matrix2 = data\_new\_classification2.drop(['Type of machine', 'Smooth'], axis=1)

corr\_matrix = matrix2.corr()

plt.figure(figsize=(20,20))

sns.heatmap(corr\_matrix, annot=False, fmt='.2f',cmap='coolwarm', square=True)

plt.title('Матрица корреляций')

plt.show()

"""# 3.1 Построение модели для датасета 1 станка"""

# выводим классы и количество строк, соответсвующих конкретному классу

unique\_values\_count = data\_new\_classification1['Smooth'].value\_counts()

print(unique\_values\_count)

"""Датасет не сбалансирован, для балансировки будем использовать метод SMOTE"""

import pandas as pd

from imblearn.over\_sampling import SMOTE

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

from imblearn.over\_sampling import RandomOverSampler

X = data\_new\_classification1.drop(['Smooth'], axis=1)

y = data\_new\_classification1['Smooth']

# Балансировка данных с помощью SMOTE

smote = SMOTE(random\_state=42, k\_neighbors=1) # Изменение k\_neighbors

X\_resampled, y\_resampled = smote.fit\_resample(X, y)

# Проверка балансировки

print("Распределение классов после SMOTE:")

print(pd.Series(y\_resampled).value\_counts())

import scipy.stats as stats

# делим выборку на тестовую и тренировочную (20% и 80%)

# Разбиение на тренировочную и тестовую выборки

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_resampled, y\_resampled, test\_size=0.2, random\_state=42, stratify=y\_resampled) # stratify для сохранения пропорций классов

stats.ttest\_ind(a=y\_train, b=y\_test, equal\_var=True)

# 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=1, shuffle=True)

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

  results.append(cv\_results)

  model\_names.append(name)

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

"""Наилучшая метрика у метода SVC (метод опорных векторов). Будем использовать ее."""

from sklearn.model\_selection import GridSearchCV

# Обучение модели SVC с GridSearchCV для подбора параметров

param\_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001], 'kernel': ['rbf']} # пример сетки параметров

grid = GridSearchCV(SVC(), param\_grid, refit=True, verbose=2)

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

print(f"Лучшие параметры: {grid.best\_params\_}")

print(f"Лучшая точность кросс-валидации: {grid.best\_score\_:.4f}")

# Предсказание на тестовой выборке

grid\_predictions = grid.predict(X\_test)

# Оценка модели

print(classification\_report(y\_test, grid\_predictions))

cm = confusion\_matrix(y\_test, grid\_predictions)

print(cm)

# Визуализация confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=grid.classes\_, yticklabels=grid.classes\_)

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.show()

"""# 3.2 Построение модели для датасета 2 станка"""

# выводим классы и количество строк, соответсвующих конкретному классу

unique\_values\_count = data\_new\_classification2['Smooth'].value\_counts()

print(unique\_values\_count)

"""Датасет не сбалансирован, для балансировки будем использовать метод SMOTE"""

import pandas as pd

from imblearn.over\_sampling import SMOTE

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

from imblearn.over\_sampling import RandomOverSampler

X = data\_new\_classification2.drop(['Smooth'], axis=1)

y = data\_new\_classification2['Smooth']

# Балансировка данных с помощью SMOTE

smote = SMOTE(random\_state=42, k\_neighbors=1) # Изменение k\_neighbors

X\_resampled2, y\_resampled2 = smote.fit\_resample(X, y)

# Проверка балансировки

print("Распределение классов после SMOTE:")

print(pd.Series(y\_resampled2).value\_counts())

import scipy.stats as stats

# делим выборку на тестовую и тренировочную (20% и 80%)

# Разбиение на тренировочную и тестовую выборки

X\_train2, X\_test2, y\_train2, y\_test2 = train\_test\_split(X\_resampled2, y\_resampled2, test\_size=0.2, random\_state=42, stratify=y\_resampled2) # stratify для сохранения пропорций классов

stats.ttest\_ind(a=y\_train2, b=y\_test2, equal\_var=True)

# 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=1, shuffle=True)

  cv\_results = cross\_val\_score(model, X\_resampled2, y\_resampled2, cv=kfold, scoring='accuracy')

  results.append(cv\_results)

  model\_names.append(name)

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

"""Наилучшая модель SVC метод опорных векторов. Также будем использовать ее."""

from sklearn.model\_selection import GridSearchCV

# Обучение модели SVC с GridSearchCV для подбора параметров

param\_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001], 'kernel': ['rbf']} # пример сетки параметров

grid = GridSearchCV(SVC(), param\_grid, refit=True, verbose=2)

grid.fit(X\_train2, y\_train2)

print(f"Лучшие параметры: {grid.best\_params\_}")

print(f"Лучшая точность кросс-валидации: {grid.best\_score\_:.4f}")

# Предсказание на тестовой выборке

grid\_predictions = grid.predict(X\_test2)

# Оценка модели

print(classification\_report(y\_test2, grid\_predictions))

cm = confusion\_matrix(y\_test2, grid\_predictions)

print(cm)

# Визуализация confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=grid.classes\_, yticklabels=grid.classes\_)

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.show()