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 «Прикладной анализ данных»

**Анализ факторов, влияющих на категорию полученного осадка на химическом заводе**

Разработчики проекта:

Овчинников Трофим Михайлович,

Мазязин Максим Леонидович

Пермь, 2024

**Оглавление**

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

[Исходные данные 4](#_heading=h.1fob9te)

[Реализация проекта 5](#_heading=h.tyjcwt)

[Этап 1. Подготовка данных к анализу 5](#_heading=h.3dy6vkm)

[Этап 2. Предварительный анализ данных 6](#_heading=h.1t3h5sf)

[Этап 3. Моделирование 11](#_heading=h.4d34og8)

[Заключение 18](#_heading=h.3rdcrjn)

[Список использованных источников и литературы 19](#_heading=h.26in1rg)

[Приложения 21](#_heading=h.lnxbz9)

**ПАСПОРТ ПРОЕКТА**

**Название проекта:**

Анализ факторов, влияющих на категорию полученного осадка на химическом заводе.

**Сведения об авторах:**

Овчинников Трофим Михайлович, Мазязин Максим Леонидович

**Цель:**

построить классификатор, который предсказывает категорию полученного осадка (изомер №1, изомер №2, брак) по имеющимся факторным переменным.

**Задачи:**

1. Выполнить подготовку данных к анализу.
2. Выполнить предварительный анализ данных.
3. Выполнить моделирование.
4. Выполнить интерпретацию полученных результатов и сделать выводы о достижении цели.

**Краткое описание проекта:**

Требуется проанализировать данные о химических соединениях, выявить зависимости между факторами, построить модель для прогнозирования категорию полученного осадка и дать прогноз. Дать интерпретацию полученным результатам. Сделать выводы.

**Конкретные ожидаемые результаты:**

Построить классификатор по имеющимся факторным переменным.

**Исходные данные**

В предоставленном файле **train.csv** описаны параметры уже проведенных реакций и полученных результатов.

Список колонок анализируемого набора данных:

* **product —** результат реакции (***целевая переменная***):  
  0 – смесь веществ (брак),   
  1 – изомер продукта №1,   
  2 – изомер продукта №2.
* **var\_1 —** вес реагента №1 в граммах.
* **var\_2 —** форма реагента №1:  
  0 – порошок,  
  1 – крупные гранулы.
* **var\_3 —**вес реагента №2 в граммах.
* **var\_4 —**форма реагента №2:  
  0 – порошок,  
  1 – крупные гранулы.
* **var\_5 —**отклонение от температуры, указанной в инструкции (87,5**°**С).
* **var\_6 —**материал реактора:  
  0 – стекло,  
  1 – металл.
* **var\_7 —**отклонение от pH, указанного в инструкции (5,8).
* **var\_8 —**использование механического перемешивания:  
  0 – не использовалось,   
  1 – использовалось.
* **var\_9 —**вес катализатора реакции в граммах.
* **var\_10 —**очередность загрузки реагентов в раствор:  
  0 – первым загружен реагент №1,  
  1 – первым загружен реагент №2.
* **var\_11 —**время реакции в минутах.
* **var\_12 —**вес полученного осадка в граммах.

В файле **test.csv** представлена тестовая выборка (параметры реакции без результата). В качестве решения необходимо представить прогноз категории осадка по каждой строчке файла **test.csv.**

Файл **target\_test\_example.csv –** шаблон файл с прогнозом. Именно в таком формате необходимо представить прогноз для автоматизированной проверки точности прогнозирования. Точность прогноза будет оцениваться по метрике «макро-точности»: **средняя точность по каждому классу**

**Реализация проекта**

**Этап 1. Подготовка данных к анализу**

Загрузим данные в датафрейм и подключим необходимые библиотеки:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import sklearn as sk

Загрузим данные и убедимся, что все количественные столбцы имеют числовой тип:

df\_train = pd.read\_csv('.\\train.csv', sep=';')

df\_train.dtypes

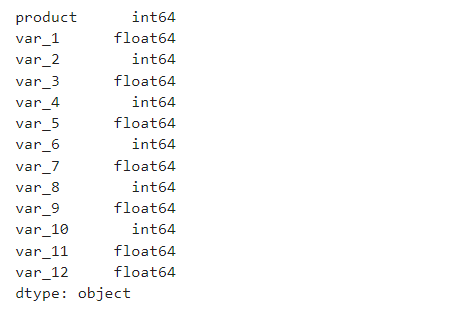


Рисунок 1. Типы данных колонок

Рассмотрим пропуски.

df\_train.info()

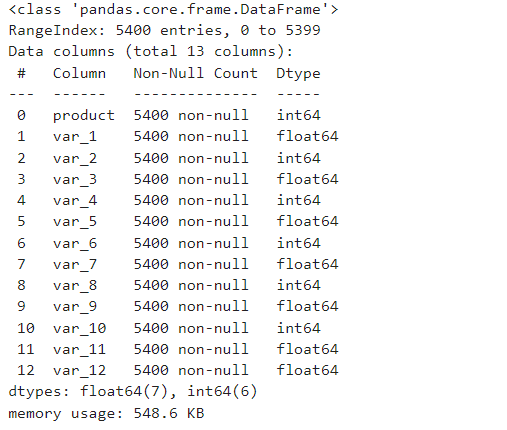


Рисунок 2. Типы данных колонок

Итоговый датафрейм:

df\_train



Рисунок 3. Датафрейм

**Этап 2. Предварительный анализ данных**

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

***Среднее арифметическое*** *равно сумме значений всех вариант выборки, деленной 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*.

**Медианой** является значение признака, находящееся в середине ранжированного ряда. Медиана находится по формуле

**Выборочная дисперсия** находится по формуле *.*

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

.

Оно показывает, как расположена основная часть вариант относительно среднего арифметического.

df\_train.describe()

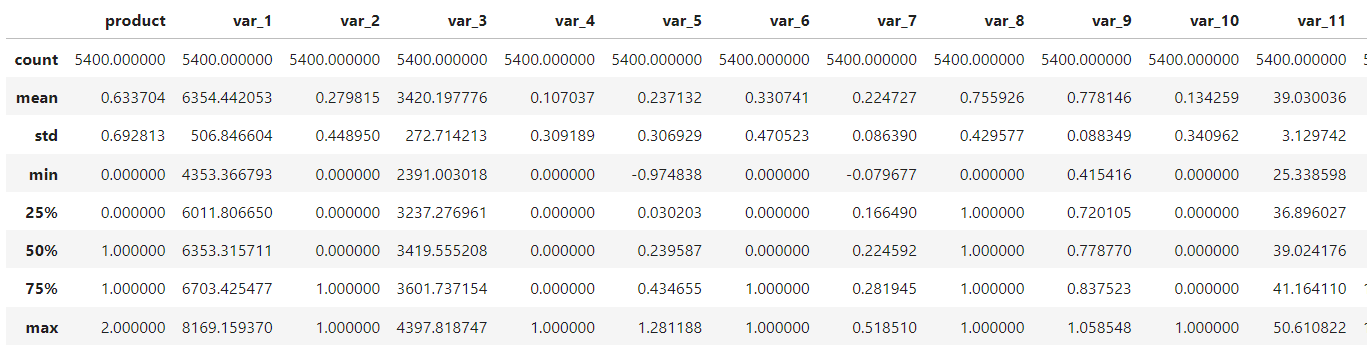


Рисунок 4. Описательные статистики по колонкам

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)). 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).

Построим отдельные диаграммы для каждой колонки:

fig, ax = plt.subplots(4, 3, figsize=(40, 30))

i = 0

for axi in ax:

for axj in axi:

axj.boxplot(df\_train[df\_train.columns[i]])

axj.set\_title(df\_train.columns[i])

i += 1

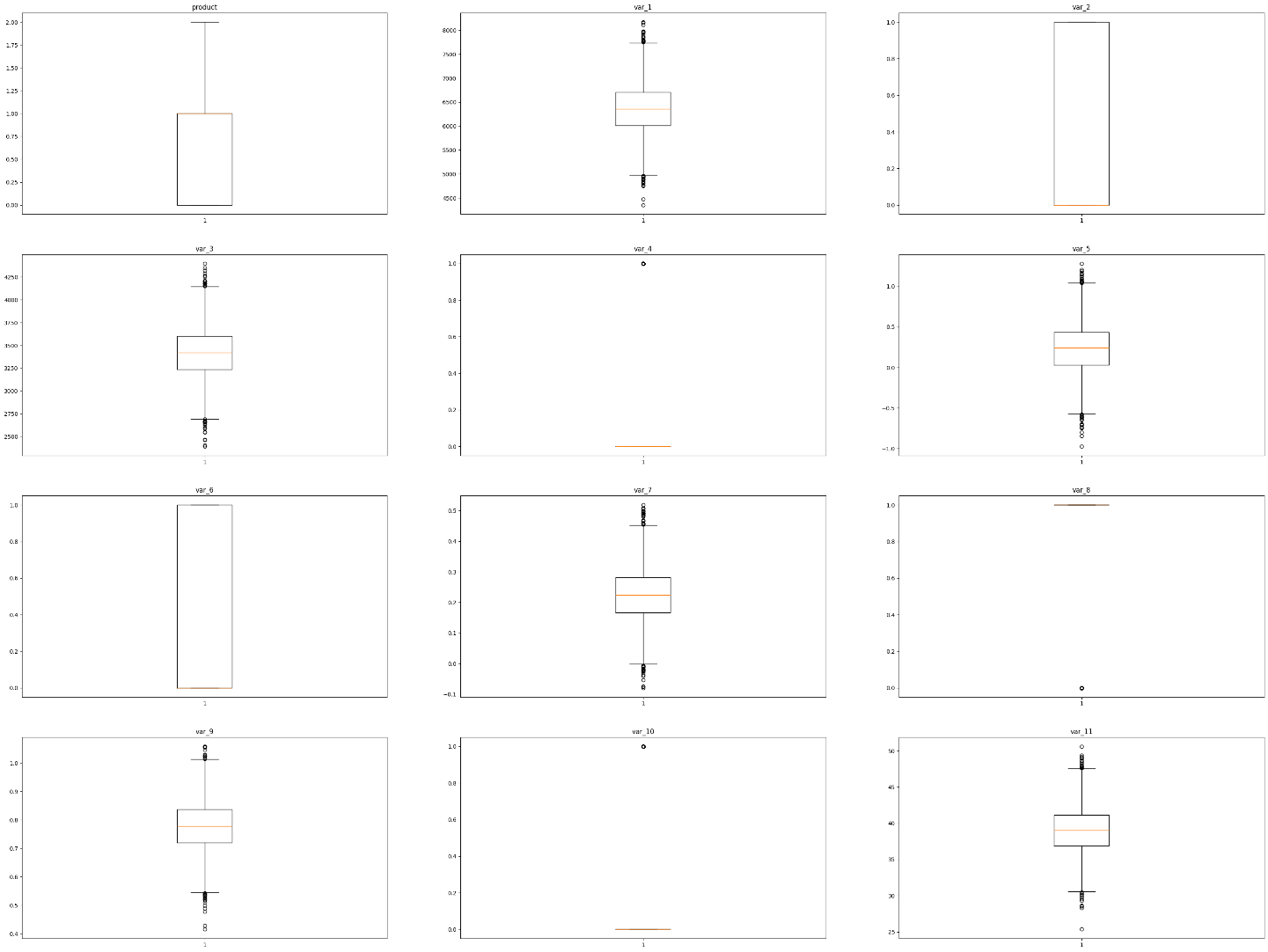


Рисунок 5. Диаграммы boxplot всех числовых колонок

Выбросов много, и они не сконцентрированы в узком диапазоне значений, а рассеяны по широкому диапазону, тогда можно ничего с ними не делать.

plt.rcParams["figure.figsize"] = 15, 8

i = 1

for col in df.columns[:-1]:

plt.subplot(2, 4, i)

plt.boxplot(df[col])

plt.title(col)

i += 1

plt.tight\_layout();

Проверим данные на нормальность распределения двумя способами:

* 1. Построим гистограмму и сделать предположение о том, являются ли данные нормально распределенными.
  2. Выполним статистический тест на нормальность и убедимся, что выдвинутое ранее предположение о нормальности верно или ошибочно.

***Гистограмма****,* представляющая собой совокупность примыкающих друг к другу прямоугольников, основание каждого из которых равно ширине интервала группировки, а площадь − частости этого интервала.

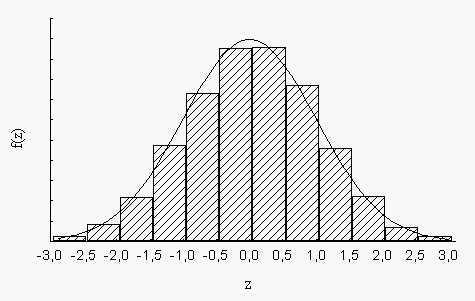


Рисунок 6. Гистограмма

Сначала построим гистограммы для всех числовых колонок:

fig, ax = plt.subplots(4, 3, figsize=(40, 30))

i = 0

for axi in ax:

for axj in axi:

axj.hist(df\_train[df\_train.columns[i]], edgecolor='k')

axj.set\_title(df\_train.columns[i])

i += 1

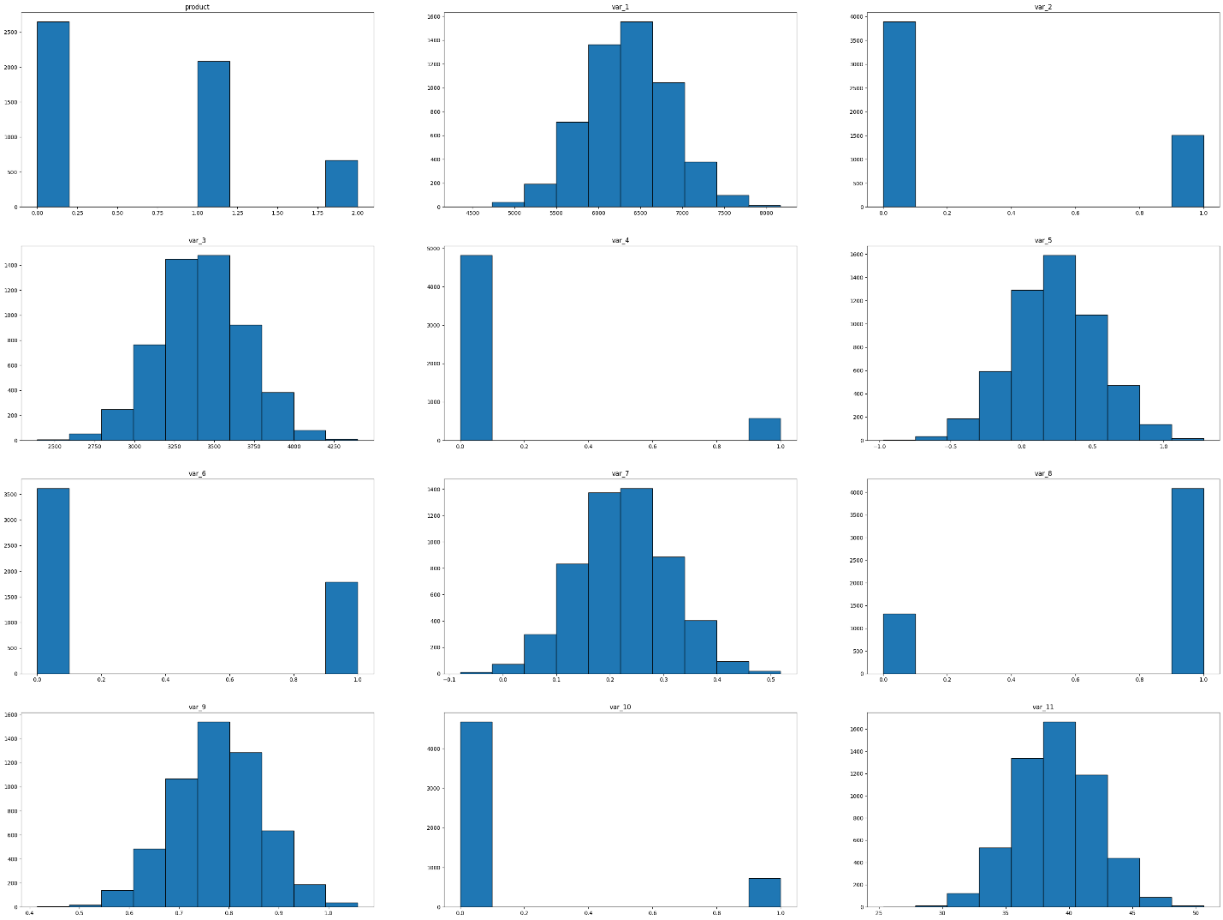


Рисунок 7. Гистограммы для всех числовых колонок

Визуально видно, что тут есть нормально распределенные колонки, но проверим через критерии согласия.

***Критерии согласия з***аключаются в проверке предположения о том, что результаты наблюдений могут быть описаны с помощью определенного закона распределения (в нашем случае нормального распределения).На формальном языке проверяется гипотеза: H0 - наши данные согласуются с нормальным распределением. Если p-value меньше заданного уровня значимости (обычно 0,05 или 0,01), то основная гипотезу отвергается.

from scipy.stats import normaltest

print('p-values:')

for i, item in enumerate(df\_train.columns):

print(f'{item}: {np.round(normaltest(df\_train).pvalue[i], 3)}')

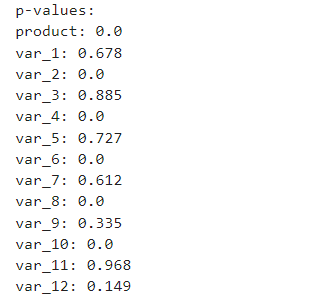


Рисунок 8. p-value для колонок

Как видим, p-value для колонок с гистограммами имеющие нормальное распределения, сделанным после визуального анализа – **подтвердилось.**

**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.

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

import seaborn as sns

sns.heatmap(df\_train.corr())

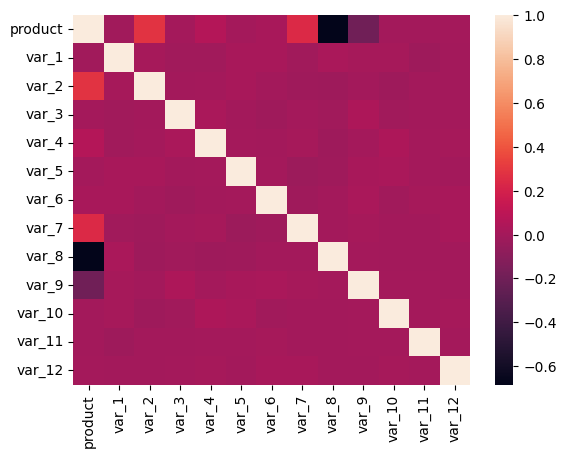


Рисунок 9. Тепловая карта матрицы корреляции

Выполнить нормализацию числовых столбцов – факторных признаков путем приведения их значений к диапазону от 0 до 1.

df\_scl = df\_train.copy()

df\_scl[df\_scl.columns[1::]]=(df\_scl[df\_scl.columns[1::]] - df\_scl[df\_scl.columns[1::]].min())/(df\_scl[df\_scl.columns[1::]].max() - df\_scl[df\_scl.columns[1::]].min())

df\_scl



Рисунок 10. Нормализованный датафрейм

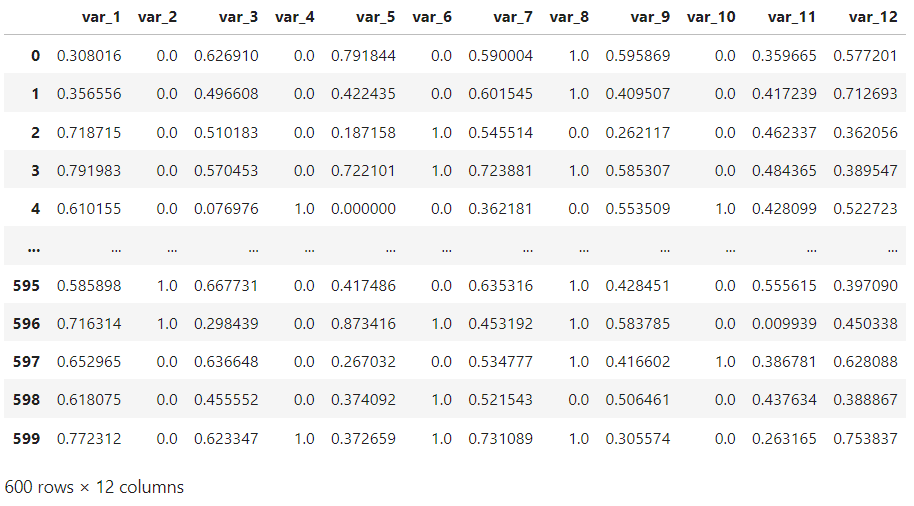
**Этап 3. Моделирование**

Разбиваем данные на обучающую и тестовую выборки (соотношение выбрать самостоятельно, обучающая выборка должна быть больше), предварительно перемешав:

df\_test = pd.read\_csv('.\\test.csv', sep=';')

df\_test = (df\_test - df\_test.min())/(df\_test.max() - df\_test.min())

df\_test

 Рисунок 11. Тестовый датафрейм

На обучающей выборке построим несколько моделей, применив различные алгоритмы классификации:

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

df1, df2 = train\_test\_split(df\_scl)

X = df1[df1.columns[1::]]

y = df1[df1.columns[0]]

X\_test = df2[df2.columns[1::]]

y\_test = df2[df2.columns[0]]

f1\_scores = []

## SVM:

from sklearn import svm

svm\_clf = svm.SVC()

svm\_clf.fit(X, y)

from sklearn.metrics import f1\_score

f1\_scores.append(['SVM', f1\_score(y\_test, svm\_clf.predict(X\_test), average='micro')])

## Stochastic Gradient Descent:

from sklearn.linear\_model import SGDClassifier as SGDC

sgdc\_clf = SGDC(loss="hinge", penalty="l2", max\_iter=100)

sgdc\_clf.fit(X,y)

f1\_scores.append(['Stochastic Gradient Descent', f1\_score(y\_test, sgdc\_clf.predict(X\_test), average='micro')])

## KNeighborsClassifier:

from sklearn.neighbors import KNeighborsClassifier as KNC

knc\_clf = KNC(n\_neighbors=5)

knc\_clf.fit(X, y)

f1\_scores.append(['KNeighborsClassifier',f1\_score(y\_test, knc\_clf.predict(X\_test), average='micro')])

## RadiusNeighborsClassifier:

from sklearn.neighbors import RadiusNeighborsClassifier as RNC

rnc\_clf = RNC()

rnc\_clf.fit(X, y)

f1\_scores.append(['RadiusNeighborsClassifier', f1\_score(y\_test, rnc\_clf.predict(X\_test), average='micro')])

## GaussianProcessClassifier:

from sklearn.gaussian\_process import GaussianProcessClassifier as GPC

gpc\_clf = GPC()

gpc\_clf.fit(X, y)

f1\_scores.append(['GaussianProcessClassifier', f1\_score(y\_test, gpc\_clf.predict(X\_test), average='micro')])

## DecisionTreeClassifier:

from sklearn.tree import DecisionTreeClassifier as DTC

dtc\_clf = DTC()

dtc\_clf.fit(X, y)

f1\_scores.append(['DecisionTreeClassifier', f1\_score(y\_test, dtc\_clf.predict(X\_test), average='micro')])

## AdaBoostClassifier:

from sklearn.ensemble import AdaBoostClassifier as ABC

abc\_clf = ABC()

abc\_clf.fit(X,y)

f1\_scores.append(['AdaBoostClassifier', f1\_score(y\_test, abc\_clf.predict(X\_test), average='micro')])

## BaggingClassifier:

from sklearn.ensemble import BaggingClassifier as bag

bag\_clf = bag()

bag\_clf.fit(X, y)

f1\_scores.append(['BaggingClassifier', f1\_score(y\_test, bag\_clf.predict(X\_test), average='micro')])

## ExtraTreesClassifier:

from sklearn.ensemble import ExtraTreesClassifier as ETC

etc\_clf = ETC()

etc\_clf.fit(X, y)

f1\_scores.append(['ExtraTreesClassifier', f1\_score(y\_test, etc\_clf.predict(X\_test), average='micro')])

## GradientBoostingClassifier:

from sklearn.ensemble import GradientBoostingClassifier as GBC

gbc\_clf = GBC()

gbc\_clf.fit(X, y)

f1\_scores.append(['GradientBoostingClassifier', f1\_score(y\_test, gbc\_clf.predict(X\_test), average='micro')])

## RandomForestClassifier:

from sklearn.ensemble import RandomForestClassifier as RFC

rfc\_clf = RFC()

rfc\_clf.fit(X,y)

f1\_scores.append(['RandomForestClassifier', f1\_score(y\_test, rfc\_clf.predict(X\_test), average='micro')])

Статистика критерия (F1-мера) вычисляется по формуле

Максимально возможное значение F-показателя равно 1,0, что указывает на идеальную точность и полноту выборки, а минимально возможное значение равно 0, если точность или полнота выборки равны нулю.

Вычислим показатель качества классификации F1-мера на тестовой выборке:

f1\_scores

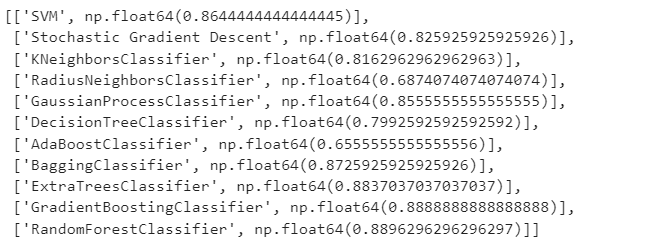
**

Рисунок 12. Результаты f1\_scores

Результат: F1-мера у всех методов стремиться к 1, ближе всего к единице **RandomForestClassifier**.

Используя 2 числовых фактора в качестве координат, строим диаграмму рассеяния на тестовой выборке, при этом, верно, классифицированные един. объекты обозначить закрашенным кружком, верно классифицированные нул. объекты – незакрашенным кружком, неверно классифицированные объекты – крестиком:

X0True = X\_test[(y\_test == rfc\_clf.predict(X\_test)) \* (y\_test == 0)]

X1True = X\_test[(y\_test == rfc\_clf.predict(X\_test)) \* (y\_test == 1)]

X2True = X\_test[(y\_test == rfc\_clf.predict(X\_test)) \* (y\_test == 2)]

XFalse = X\_test[(y\_test != rfc\_clf.predict(X\_test))]

plt.scatter(X0True.var\_9, X0True.var\_11, marker='o', label='Истинные 0')

plt.scatter(X1True.var\_9, X1True.var\_11, marker='>', label='Истинные 1')

plt.scatter(X2True.var\_9, X2True.var\_11, marker='<', label='Истинные 2')

plt.scatter(XFalse.var\_9, XFalse.var\_11, marker='x', label='Ложные')

plt.legend()

plt.xlabel('Вес катализатора реакции')

plt.ylabel('Время реакции')

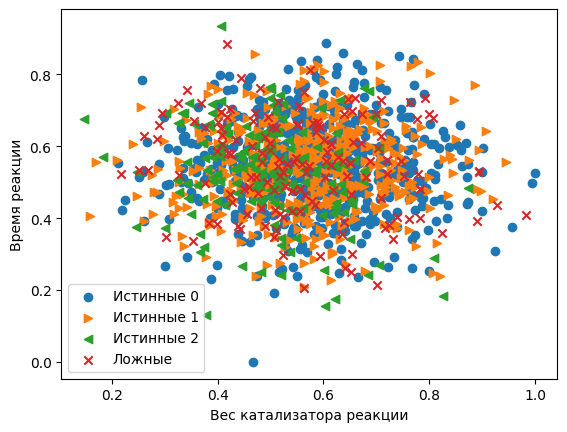


Рисунок 13. Диаграмма рассеяния

Вычислить другие показатели качества классификации: accuracy, precision, recall, построить матрицу ошибок:

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

print(classification\_report(y\_test, rfc\_clf.predict(X\_test)))

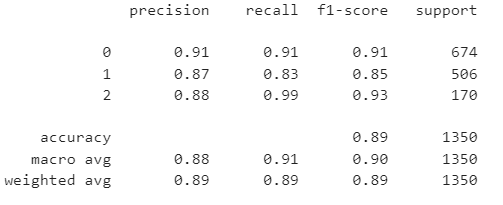


Рисунок 10. Показатели качества классификации

sns.heatmap(confusion\_matrix(y\_test, rfc\_clf.predict (X\_test)))

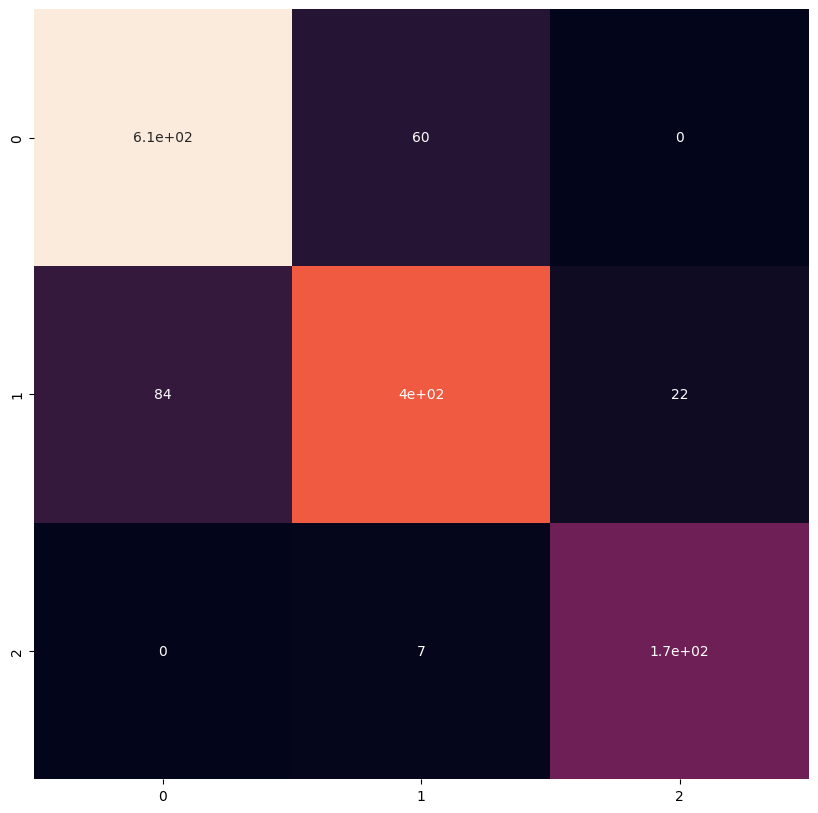
**

Рисунок 14. Матрица ошибок

rfc\_clf.predict(df\_test)

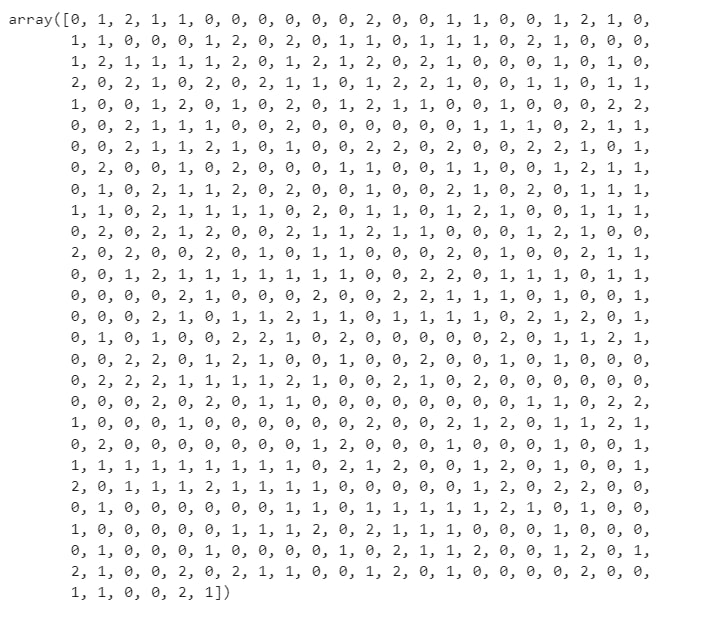


Рисунок 15. Прогнозируемый результат для файла ‘test.csv’

**Заключение**

Задачи работы были выполнены в то числе:

1. Выполнена подготовка данных к анализу (рис.3).
2. Выполнен предварительный анализ данных (рис. 5, 7, 9, 10).
3. Выполнено моделирование (SVM, Stochastic Gradient Descent, KNeighborsClassifier, RadiusNeighborsClassifier, GaussianProcessClassifier, DecisionTreeClassifier, AdaBoostClassifier, BaggingClassifier, ExtraTreesClassifier, GradientBoostingClassifier, RandomForestClassifier) и найдена f1-мера для них (рис. 12).

Цель работы была выполенапостроить классификатор, который предсказывает категорию полученного осадка (изомер №1, изомер №2, брак) по имеющимся факторным переменным - построена диаграмма рассеяния (рис. 13) и прогнозируемый результат (рис. 14)

**Список использованных источников и литературы**

1. Федин, Ф. О. Анализ данных. Часть 1. Подготовка данных к анализу : учебное пособие / Ф. О. Федин, Ф. Ф. Федин. — Москва : Московский городской педагогический универ-ситет, 2012. — 204 c. — ISBN 2227-8397. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт].
2. Амоа К.А. Разработка программных пакетов на языке Python: учебное пособие / К.А. Амоа, Н.А. Рындин, Ю.С. Скворцов. – Воронеж: Воронежский государственный технический университет, ЭБС АСВ, 2020. – 61 c. // Электронно-библиотечная система IPR BOOKS: [сайт].
3. Гуриков С. Р. Основы алгоритмизации и программирования на Python : учеб. пособие / С.Р. Гуриков. ? М. : ФОРУМ : ИНФРА-М, 2018. ? 343 с. ? (Высшее образование: Бакалав-риат). - Режим доступа: http://znanium.com/catalog/product/924699
4. Вандер П. Python для сложных задач: наука о данных и машинное обучение. — СПб.: Питер, 2018. — 576 с.
5. Васильев, А.Н. Программирование на Python в примерах и задачах / А,Н. Васильев. — Москва : Эксмо, 2021. — 616 с.
6. Сузи Р.А. Язык программирования Python: учебное пособие / Р.А. Сузи. – 3-е изд. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. – 350 c. // Электронно-библиотечная система IPR BOOKS: [сайт].
7. Волкова В.М., Программные системы статистического анализа. Обнаружение закономерностей в данных с использованием системы R и языка Python [Электронный ресурс]: учебное пособие / Волкова В.М. - Новосибирск : Изд-во НГТУ, 2017. - 74 с. - ISBN 978-5-7782-3183-2 - Режим доступа: http://www.studentlibrary.ru/book/ISBN9785778231832.html
8. Криволапов С.Я. Математика на Python : учебник / С.Я. Криволапов, М.Б. Хрипунова. — Москва: КНОРУС, 2022. — 456 с.
9. Федин, Ф. О. Анализ данных. Часть 1. Подготовка данных к анализу : учебное пособие / Ф. О. Федин, Ф. Ф. Федин. — Москва : Московский городской педагогический универ-ситет, 2012. — 204 c. — ISBN 2227-8397. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт].
10. Федин, Ф. О. Анализ данных. Часть 2. Инструменты DataMining : учебное пособие / Ф. О. Федин, Ф. Ф. Федин. — Москва : Московский городской педагогический университет, 2012. — 308 c. — ISBN 2227-8397. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт].
11. Федоров Д.Ю. Программирование на языке высокого уровня Python: учебное пособие / Д.Ю. Федоров. – 2-е изд.– М.: Юрайт, 2020. – 161 с.
12. Филлипс Т. Управление на основе данных. Как интерпретировать цифры и принимать качественные решения в бизнесе. – М.:Манн, Иванов и Фербер, 2017. - 117 с.
13. Фрэнкс, Б. Революция в аналитике: как в эпоху BigData улучшить ваш бизнес с помощью операционной аналитики / Б. Фрэнкс; Пер. с англ. И. Евстигнеевой; Ред. В. Мылов. – М.: Альпина Паблишер, 2016. – 315 с.

**Приложения**

Приложение 1

Программный код

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import sklearn as sk

df\_train = pd.read\_csv('.\\train.csv', sep=';')

df\_train.dtypes

df\_train.info()

df\_train

df\_train.describe()

fig, ax = plt.subplots(4, 3, figsize=(40, 30))

i = 0

for axi in ax:

for axj in axi:

axj.boxplot(df\_train[df\_train.columns[i]])

axj.set\_title(df\_train.columns[i])

i += 1

plt.rcParams["figure.figsize"] = 15, 8

i = 1

for col in df.columns[:-1]:

plt.subplot(2, 4, i)

plt.boxplot(df[col])

plt.title(col)

i += 1

plt.tight\_layout();

from scipy.stats import normaltest

print('p-values:')

for i, item in enumerate(df\_train.columns):

print(f'{item}: {np.round(normaltest(df\_train).pvalue[i], 3)}')

df\_test = pd.read\_csv('.\\test.csv', sep=';')

df\_test = (df\_test - df\_test.min())/(df\_test.max() - df\_test.min())

df\_test

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

df1, df2 = train\_test\_split(df\_scl)

X = df1[df1.columns[1::]]

y = df1[df1.columns[0]]

X\_test = df2[df2.columns[1::]]

y\_test = df2[df2.columns[0]]

f1\_scores = []

from sklearn import svm

svm\_clf = svm.SVC()

svm\_clf.fit(X, y)

from sklearn.metrics import f1\_score

f1\_scores.append(['SVM', f1\_score(y\_test, svm\_clf.predict(X\_test), average='micro')])

from sklearn.linear\_model import SGDClassifier as SGDC

sgdc\_clf = SGDC(loss="hinge", penalty="l2", max\_iter=100)

sgdc\_clf.fit(X,y)

f1\_scores.append(['Stochastic Gradient Descent', f1\_score(y\_test, sgdc\_clf.predict(X\_test), average='micro')])

from sklearn.neighbors import KNeighborsClassifier as KNC

knc\_clf = KNC(n\_neighbors=5)

knc\_clf.fit(X, y)

f1\_scores.append(['KNeighborsClassifier',f1\_score(y\_test, knc\_clf.predict(X\_test), average='micro')])

from sklearn.neighbors import RadiusNeighborsClassifier as RNC

rnc\_clf = RNC()

rnc\_clf.fit(X, y)

f1\_scores.append(['RadiusNeighborsClassifier', f1\_score(y\_test, rnc\_clf.predict(X\_test), average='micro')])

from sklearn.gaussian\_process import GaussianProcessClassifier as GPC

gpc\_clf = GPC()

gpc\_clf.fit(X, y)

f1\_scores.append(['GaussianProcessClassifier', f1\_score(y\_test, gpc\_clf.predict(X\_test), average='micro')])

from sklearn.tree import DecisionTreeClassifier as DTC

dtc\_clf = DTC()

dtc\_clf.fit(X, y)

f1\_scores.append(['DecisionTreeClassifier', f1\_score(y\_test, dtc\_clf.predict(X\_test), average='micro')])

from sklearn.ensemble import AdaBoostClassifier as ABC

abc\_clf = ABC()

abc\_clf.fit(X,y)

f1\_scores.append(['AdaBoostClassifier', f1\_score(y\_test, abc\_clf.predict(X\_test), average='micro')])

from sklearn.ensemble import BaggingClassifier as bag

bag\_clf = bag()

bag\_clf.fit(X, y)

f1\_scores.append(['BaggingClassifier', f1\_score(y\_test, bag\_clf.predict(X\_test), average='micro')])

from sklearn.ensemble import ExtraTreesClassifier as ETC

etc\_clf = ETC()

etc\_clf.fit(X, y)

f1\_scores.append(['ExtraTreesClassifier', f1\_score(y\_test, etc\_clf.predict(X\_test), average='micro')])

from sklearn.ensemble import GradientBoostingClassifier as GBC

gbc\_clf = GBC()

gbc\_clf.fit(X, y)

f1\_scores.append(['GradientBoostingClassifier', f1\_score(y\_test, gbc\_clf.predict(X\_test), average='micro')])

from sklearn.ensemble import RandomForestClassifier as RFC

rfc\_clf = RFC()

rfc\_clf.fit(X,y)

f1\_scores.append(['RandomForestClassifier', f1\_score(y\_test, rfc\_clf.predict(X\_test), average='micro')])

f1\_scores

X0True = X\_test[(y\_test == rfc\_clf.predict(X\_test)) \* (y\_test == 0)]

X1True = X\_test[(y\_test == rfc\_clf.predict(X\_test)) \* (y\_test == 1)]

X2True = X\_test[(y\_test == rfc\_clf.predict(X\_test)) \* (y\_test == 2)]

XFalse = X\_test[(y\_test != rfc\_clf.predict(X\_test))]

plt.scatter(X0True.var\_9, X0True.var\_11, marker='o', label='Истинные 0')

plt.scatter(X1True.var\_9, X1True.var\_11, marker='>', label='Истинные 1')

plt.scatter(X2True.var\_9, X2True.var\_11, marker='<', label='Истинные 2')

plt.scatter(XFalse.var\_9, XFalse.var\_11, marker='x', label='Ложные')

plt.legend()

plt.xlabel('Вес катализатора реакции')

plt.ylabel('Время реакции')

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

print(classification\_report(y\_test, rfc\_clf.predict(X\_test)))

sns.heatmap(confusion\_matrix(y\_test, rfc\_clf.predict (X\_test)))

rfc\_clf.predict(df\_test)