УДК 004

# Интеллектуальный анализ данных вакансий в Еврейской автономной области при помощи машинного обучения

*Анишкова Анастасия Сергеевна*
*Приамурский государственный университет имени Шолом-Алейхема*
*Студент*

**Аннотация**
Целью исследования является интеллектуальный анализ данных вакансий в Еврейской автономной области при помощи машинного обучения. Для реализации использовалась облачная платформа для создания и выполнения кода на Python Google Colab. Полученный результат можно использовать как учебное пособие.
**Ключевые слова:** интеллектуальный анализ, вакансии, машинное обучение, Google Colab.

# Intelligent analysis of job data in the Jewish Autonomous Region using machine learning

*Anishkova Anastasia Sergeevna*
*Sholom Aleichem Priamurskiy State University*
*Student*

**Abstract**
The purpose of the study is the intelligent analysis of job data in the Jewish Autonomous Region using machine learning. For the implementation, Google Colab, a cloud platform for creating and executing Python code, was used. The result can be used as a textbook.
**Key words:** intellectual analysis, vacancies, machine learning, Google Colab.

## 1 Введение

### 1.1 Актуальность

Вакансии в Еврейской автономной области являются важным источником информации о рынке труда. Анализ этих данных может помочь работодателям определить потребности рынка труда, а также спрогнозировать изменение спроса на определенные профессии.

### 1.2 Обзор исследований

Модель, описанная А.В. Толмачевым и О.Н. Красавиной, представляет собой модель машинного обучения, предназначенную для оценки стоимости квадратного метра недвижимости. Данная модель, вероятно, использует различные характеристики объектов недвижимости, такие как местоположение, площадь, возраст, удобства и рыночные тенденции, для

прогнозирования цены за квадратный метр. Алгоритмы машинного обучения, такие как регрессионные модели или нейронные сети, обучаются на наборе данных с историческими данными о недвижимости, чтобы выявить взаимосвязи между этими характеристиками и соответствующими ценами. Цель состоит в том, чтобы предоставить точный и основанный на данных инструмент для оценки недвижимости, который может быть полезен для покупателей, продавцов и профессионалов в области недвижимости [1].

А. М. Васильченко описал как проводить анализ данных при помощи библиотек Python [2].

Н.Е. Косых предложил подход к подбору наиболее производительного набора параметров для объекта классификатора текста. Для вычислений использовал облачный сервис Google Colaboratory, выполняющий код на языке Python внутри браузера, используя виртуальные аппаратные ресурсы [3].

Т.М. Татарникова, Е.Д. Пойманова, П.Ю. Богданов, Е.В. Краева, С.А. Веревкин в статье рассмотрели способы и методы изучения и построения нейронных сетей. Показано, что изучение принципов функционирования нейронных сетей, их применение для решения тех или иных задач возможны только через практику. Проведен анализ различных программных сред, которые могут быть использованы на лабораторных и практических занятиях по изучению и применению нейронных сетей. Выделен современный облачный сервис Google Colaboratory, рекомендуемый для обучения основам нейронных сетей благодаря наличию в нем предустановки библиотеки Tensorflow и библиотеки для работы на языке Python, бесплатного доступа к графическим процессорам, возможности написания и выполнения программного кода в браузере, а также отсутствию необходимости специальной настройки сервиса [4].

Е.А. Григорьев, Н.С. Климов в статье рассматривали разведочный анализ данных. Описали инструменты реализации анализа, библиотеки Python. Представлен пример, выполненный на данных обнаружению присутствия людей в помещении [5].

Ю.С. Попков рассмотрел основные понятия и алгоритмы машинного обучения и его рандомизированной версии. Последняя является эффективным инструментом решения задач классификации и прогнозирования в условиях неопределённости [6].

Автор отмечает, что прогнозирование цен на недвижимость является сложной задачей из-за большого количества факторов, влияющих на стоимость объектов. Традиционные методы прогнозирования, основанные на экспертных оценках и статистических моделях, часто не учитывают все эти факторы и дают неточные результаты [7].

С. В. Шайтура провел интеллектуальный анализ данных В частности, описан процесс интеллектуального анализа данных в трёх стадиях: выявление закономерностей; использование выявленных закономерностей для предсказания неизвестных значений; анализ исключений,

предназначенный для выявления и толкования аномалий в найденных закономерностях [8].

Д.Ю. Городничев в своем исследовании обобщает основы машинного обучения и глубокого обучения для получения более широкого представления о методической основе современных интеллектуальных систем. В частности, предоставляется концептуальное различие между соответствующими терминами и понятиями, объяснение процесс автоматизированного построения аналитических моделей с помощью машинного обучения и глубокого обучения, а также обсуждение проблем, возникающих при внедрении таких интеллектуальных систем в сфере электронных рынков и сетевого бизнеса [9].

Автор рассмотрел возможности программного обеспечения Google Colaboratory и Jupyter Notebook для решения задач искусственного интеллекта. Были изучены основные функции и методы использования интерактивного блокнота Jupyter Notebook и облачного сервиса Google Colaboratory. [10].

## 2 Цель исследования

Целью данной темы является разработка и применение методов интеллектуального анализа данных и машинного обучения для обработки и анализа вакансий в Еврейской автономной области. Основная задача состоит в создании системы, которая позволит автоматически анализировать данные о вакансиях, выделять ключевые тенденции и прогнозировать изменения на рынке труда. Это поможет работодателям принимать обоснованные решения при подборе персонала, а также даст возможность государственным органам и исследовательским организациям получить ценную информацию для разработки стратегий развития региона.

## 3 Материалы и методы

В данном исследование используется Google Colab — сервис, созданный Google, который позволяет работать с кодом на языке Python через Jupyter Notebook.

## 4 Результаты

Для проведения интеллектуального анализа данных о вакансиях в Еврейской автономной области, потребуется импортировать библиотеки и модели (см. рис.1).

```
# Импорт необходимых библиотек и моделей
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Рисунок 1. Импортирование библиотек

Для проведения интеллектуального анализа данных о вакансиях в Еврейской автономной области был подключен датасет, содержащий информацию о различных вакансиях, предлагаемых на региональном рынке труда. Данные были получены из открытых источников и сохранены в формате Excel.

Датасет можно скачать по ссылке https://cloud.mail.ru/public/4Gwz/A6g7o6p5h

Датасет состоит из следующих столбцов:

kategoria_vakansii – категория вакансий;

ot_zp – начальная граница заработной платы;

do_zp – конечная граница заработной платы;

rod_grafik – категория графика работы;

kategoria_zanytosti – категория занятости;

zanytost – занятость;

kod_obrazov – катоегория образования

min_granica – минимальная граница по ребованию образования;

код_доп – дополнительное образование;

код_опыт работы – наличие опыта работы.

Вывод первых строк датасета позволил оценить качество данных, выявить возможные пропущенные значения или аномалии и подготовить данные для дальнейшего анализа. В дальнейшем, после предварительной обработки и очистки данных, были применены различные методы машинного обучения (см. рис.2).

```
[77] # Подключение датасета
dan_dataset= pd.read_excel('vak (1) (1).xlsx')
```

```
[78] # Просмотр датасета
dan_dataset.head()
```

| | № | kategoria_vakansii | ot_zp | do_zp | rod_grafik | kategoria_zanytosti | zanytost | kod_obrazov | min_granica | код_доп | opit_rab | код_опыт работы |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | Продажи | 50000 | 50000 | 1 | на месте | Полная | 3 | 1 | 2 | не требуется | 4 |
| 1 | 2 | бухгалтер | 75000 | 75000 | 1 | на месте | Полная | 3 | 1 | 2 | не требуется | 3 |
| 2 | 3 | оператор | 30000 | 44000 | 5 | на месте | Полная | 0 | 0 | 2 | не требуется | 3 |
| 3 | 4 | оператор | 30000 | 44000 | 5 | на месте | Полная | 0 | 0 | 2 | не требуется | 3 |
| 4 | 5 | Водитель | 30000 | 44000 | 1 | на месте | свободный график | 0 | 0 | 0 | не требуется | 3 |

Рисунок 2. Подгрузка датасета

В процессе работы с данными для оценки размера и структуры датасета был использован атрибут dan_dataset.shape. Этот атрибут возвращает кортеж,

содержащий информацию о количестве строк и столбцов в DataFrame. Первое значение в кортеже указывает на количество строк, а второе значение - на количество столбцов (см. рис.3).

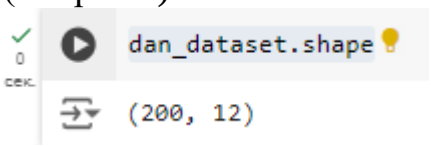



Рисунок 3. Оценка размеров датасета

Использование метода dan_dataset.info() позволяет получить важные сведения о структуре датасета, что является основой для предварительного анализа данных и подготовки их к последующему использованию в моделях машинного обучения. В данном исследование этот метод поможет оценить качество данных, выявить возможные проблемы, связанные с пропущенными значениями или несоответствием типов данных, и разработать стратегию их решения для более точного и эффективного анализа рынка труда в регионе (см. рис.4).

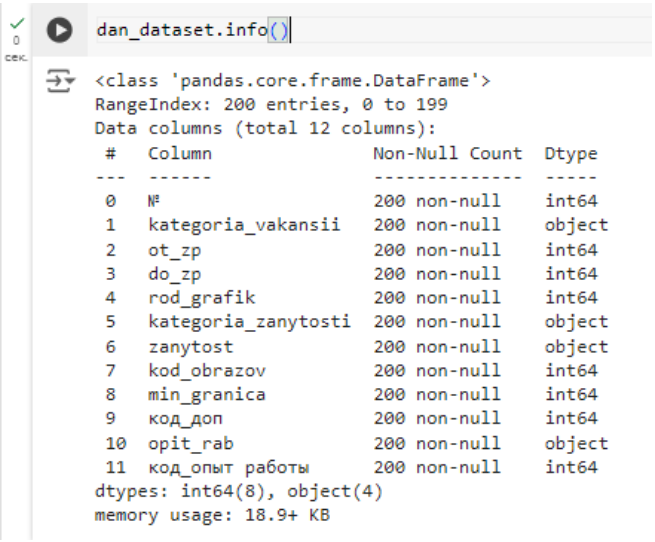



Рисунок 4. Сведения о датасете

Вывод количества вхождений каждого уникального значения позволил оценить распределение данных по категориям и выявить наиболее часто встречающиеся значения. Это, в свою очередь, помогло определить, какие категории и характеристики вакансий являются наиболее распространенными на рынке труда (см. рис.5).

```
# Вывод количество вхождений каждого уникального значения


print(dan_dataset.kategoria_zanytosti.value_counts())
print(dan_dataset.zanytost.value_counts())

print(dan_dataset.код_доп.value_counts())
print(dan_dataset.opit_rab.value_counts())

print(dan_dataset.kategoria_vakansii.value_counts())
print(dan_dataset.kod_obrazov.value_counts())
print(dan_dataset.min_granica.value_counts())
```

```
kategoria_zanytosti
на месте           149
вахтовый метод      51
Name: count, dtype: int64
zanytost
Полная              199
свободный график      1
Name: count, dtype: int64
код_доп
2    167
0     30
1      2
3      1
Name: count, dtype: int64
opit_rab
не требуется    86
1-3 года        70
3-6 лет         35
6 лет            5
требуется        4
Name: count, dtype: int64
kategoria_vakansii
Рабочие          50
Продажи          45
Водитель         18
Машинист         13
специалист       11
оператор         10
инженер          10
Общепит           8
медицина          6
бухгалтер         6
руководитель      5
мастер            5
горничная         3
Грузчик           3
Администратор     3
Контролер         2
суд               2
Name: count, dtype: int64
kod_obrazov
0    113
3     33
1     29
2     25
Name: count, dtype: int64
min_granica
0    167
1     33
Name: count, dtype: int64
```

Рисунок 5. Вывод количества вхождений уникального значения

Была проведена замена текстовых значений в некоторых столбцах датасета на числовые. Это необходимо для удобства работы с данными и применения различных методов машинного обучения, которые требуют числовых входных данных (см. рис.6).

```
# Замена текстовых значений в столбце House_type на числовые
dan_dataset.replace({'grafik':{'помесячно':0,'удаленно':1,'гибкий':2, 'полный день' :3, 'помесячно' :4, 'сменный' :5, 'стажировка' :6, 'частичный' :7}},inplace=True)

# Замена текстовых значений в столбце Type_of_rooms на числовые
dan_dataset.replace({'kategoria_zanytosti':{'вахтовый метод':0,'на месте':1}},inplace=True)

# Замена текстовых значений в столбце Bathroom на числовые
dan_dataset.replace({'zanytost':{'Полная':0,'свободный график':1}},inplace=True)
# Замена текстовых значений в столбце Repair на числовые
dan_dataset.replace({'obraz_dop':{'водительское удостоверение':0,'медицинское':1,'нет ограничений':2, 'удостоверение' :3, }},inplace=True)
# Замена текстовых значений в столбце Repair на числовые
dan_dataset.replace({'kategoria_vakansii':{'Администратор':0,'бухгалтер':1,'Водитель':2, 'горничная' :3,'Грузчик' :4, 'Контролер' :5, 'Машинист' :6
                                           ,'Общепит' :7, 'оператор' :8, 'Продажи' :9, 'Рабочие' :10, 'суд' :11, 'специалист' :12,
                                           'инженер' :13, 'мастер' :14, 'медицина' :15, 'специалист' :16, 'руководитель' :17}},inplace=True)
# Замена текстовых значений в столбце Type_of_rooms на числовые
dan_dataset.replace({'kategoria_zanytosti':{'1-3 года':0,'3-6 лет':1,'6 лет':2,'не требуется':3,'требуется':4}},inplace=True)
```

Рисунок 6. Замена текстовых значений

Была проведена замена текстовых значений в некоторых столбцах датасета на числовые. Это необходимо для удобства работы с данными и применения различных методов машинного обучения, которые требуют числовых входных данных.

```
dan_dataset.head()
```

| | № | kategoria_vakansii | ot_zp | do_zp | rod_grafik | kategoria_zanytosti | zanytost | kod_obrazov | min_granica | код_доп | opit_rab | код_опыт | работы |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 9 | 50000 | 50000 | 1 | 1 | 0 | 3 | 1 | 2 | не требуется | | 4 |
| 1 | 2 | 1 | 75000 | 75000 | 1 | 1 | 0 | 3 | 1 | 2 | не требуется | | 3 |
| 2 | 3 | 8 | 30000 | 44000 | 5 | 1 | 0 | 0 | 0 | 2 | не требуется | | 3 |
| 3 | 4 | 8 | 30000 | 44000 | 5 | 1 | 0 | 0 | 0 | 2 | не требуется | | 3 |
| 4 | 5 | 2 | 30000 | 44000 | 1 | 1 | 1 | 0 | 0 | 0 | не требуется | | 3 |

Рисунок 7. Замена текстовых значений

Вывод первых строк датасета, для того, чтобы посмотреть на категориальную замену (см. рис.8)

```
dan_dataset.head()
```

| | № | kategoria_vakansii | ot_zp | do_zp | rod_grafik | kategoria_zanytosti | zanytost | kod_obrazov | min_granica | код_доп | opit_rab | код_опыт | работы |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 9 | 50000 | 50000 | 1 | 1 | 0 | 3 | 1 | 2 | не требуется | | 4 |
| 1 | 2 | 1 | 75000 | 75000 | 1 | 1 | 0 | 3 | 1 | 2 | не требуется | | 3 |
| 2 | 3 | 8 | 30000 | 44000 | 5 | 1 | 0 | 0 | 0 | 2 | не требуется | | 3 |
| 3 | 4 | 8 | 30000 | 44000 | 5 | 1 | 0 | 0 | 0 | 2 | не требуется | | 3 |
| 4 | 5 | 2 | 30000 | 44000 | 1 | 1 | 1 | 0 | 0 | 0 | не требуется | | 3 |

Далее: Посмотреть рекомендованные графики

Рисунок 8. Вывод первых строк

Важно оценить наличие пропущенных значений в датасете. Для этого был использован метод dan_dataset.isnull(), который возвращает DataFrame с булевыми значениями, указывающими на наличие или отсутствие пропущенных данных в каждой ячейке исходного датасета (см. рис.9).

```
# Evaluating for Missing Data
missing_data = dan_dataset.isnull()
missing_data.head()
```

| | № | kategoria_vakansii | ot_zp | do_zp | rod_grafik | kategoria_zanytosti | zanytost | kod_obrazov | min_granica | код_доп | opit_rab | код_опыт | работы |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | False | False | False | False | False | False | False | False | False | False | False | False | False |
| 1 | False | False | False | False | False | False | False | False | False | False | False | False | False |
| 2 | False | False | False | False | False | False | False | False | False | False | False | False | False |
| 3 | False | False | False | False | False | False | False | False | False | False | False | False | False |
| 4 | False | False | False | False | False | False | False | False | False | False | False | False | False |

Рисунок 9. Просмотр наличия пропущенных значений

Вызов dan_dataset.describe() позволяет получить представление о центральной тенденции, разбросе и форме распределения данных в числовых столбцах датасета. Эта информация важна для понимания основных характеристик данных и выявления возможных аномалий или выбросов (см. рис.10).

```
# Вывод статистических данных
dan_dataset.describe()
```

| | № | kategoria_vakansii | ot_zp | do_zp | rod_grafik | kategoria_zanytosti | zanytost | kod_obrazov | min_granica | код_доп | код_опыт работы |
|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 200.000000 | 200.000000 | 200.000000 | 200.000000 | 200.000000 | 200.000000 | 200.000000 | 200.000000 | 200.000000 | 200.000000 | 200.000000 |
| mean | 100.500000 | 8.825000 | 78298.670000 | 88068.810000 | 1.835000 | 0.745000 | 0.005000 | 0.890000 | 0.165000 | 1.695000 | 1.600000 |
| std | 57.879185 | 4.069049 | 48658.931993 | 52913.960419 | 1.347909 | 0.436955 | 0.070711 | 1.159579 | 0.372112 | 0.724291 | 1.396334 |
| min | 1.000000 | 0.000000 | 13000.000000 | 20000.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 50.750000 | 7.000000 | 40000.000000 | 45000.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 2.000000 | 0.000000 |
| 50% | 100.500000 | 9.000000 | 63320.000000 | 75000.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 2.000000 | 1.000000 |
| 75% | 150.250000 | 10.000000 | 110000.000000 | 120000.000000 | 2.000000 | 1.000000 | 0.000000 | 2.000000 | 0.000000 | 2.000000 | 3.000000 |
| max | 200.000000 | 17.000000 | 250000.000000 | 250000.000000 | 6.000000 | 1.000000 | 1.000000 | 3.000000 | 1.000000 | 3.000000 | 4.000000 |

Рисунок 10. Представление о центральной тенденции

Было построено несколько типов графиков. Одним из них является столбчатая диаграмма (см. рис.11), которая позволяет визуализировать зависимость заработной платы (ot_zp) от категории вакансии (kategoria_vakansii).

```
# Построение различных типов графиков для анализа важности данных
# столбчатая диаграмма
dan_dataset.plot(x='kategoria_vakansii', y='ot_zp', kind='bar', figsize=(25, 6))
plt.title('ot_zp')
plt.xlabel('kategoria_vakansii')
plt.ylabel('ЗП')
plt.show()
```

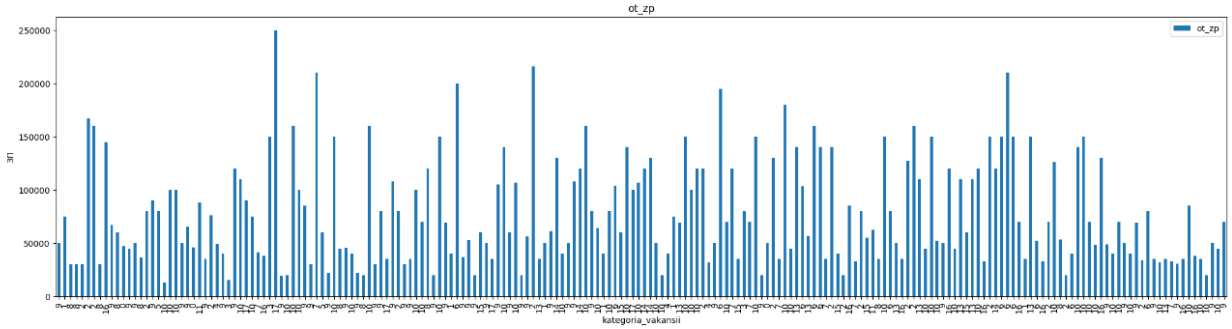



Рисунок 11. Столбчатая диаграмма

На каждой из четырех осей были построены столбчатые диаграммы (см. рис.12), отображающие количество вхождений каждого уникального значения в соответствующих столбцах датасета: kategoria_zanytosti, zanytost, kod_obrazov, min_granica, код_доп и код_опыт работы. Для построения графиков использовался метод value_counts().plot(kind='bar'), а затем указывались соответствующие оси с помощью параметра ax=axs[i, j].

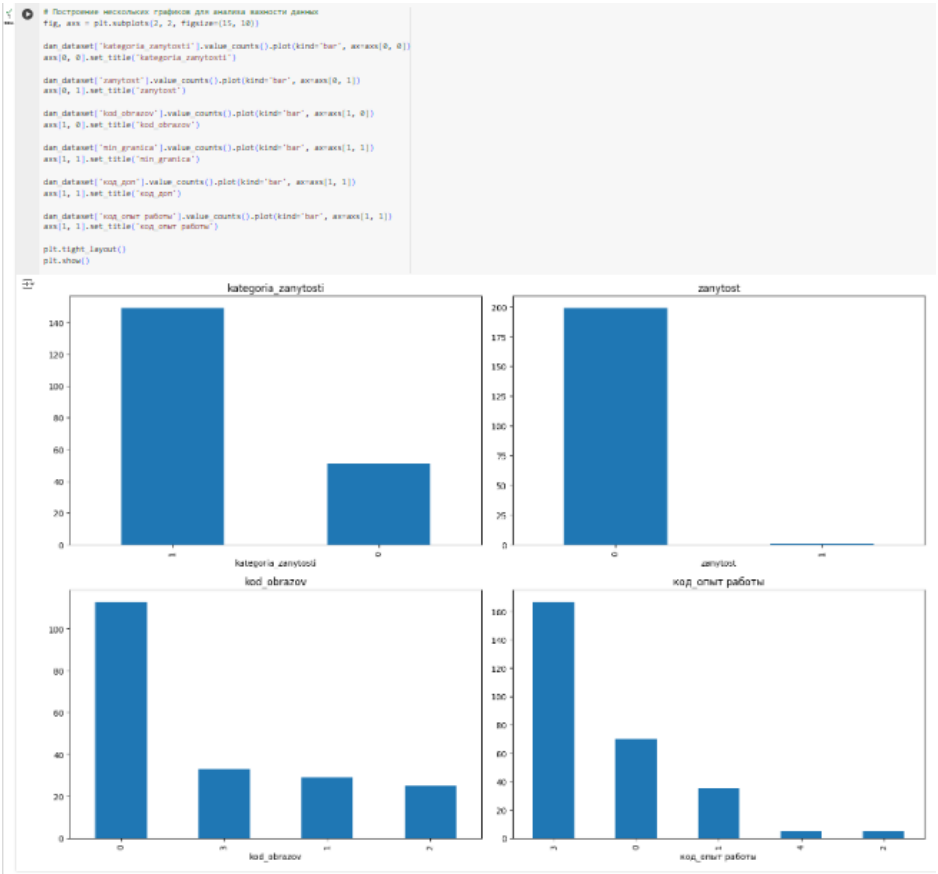



Рисунок 12. Столбчатые диаграммы

На каждой из четырех осей были построены графики рассеивания (см.рис.13), отображающие зависимость заработной платы от различных характеристик
вакансий: kategoria_zanytosti, zanytost, kod_obrazov и код_опыт работы. Цвета точек на графиках были заданы в соответствии с категорией вакансии (kategoria_vakansii), что позволило визуализировать распределение данных по категориям вакансий на графиках рассеивания. Для окрашивания точек использовался параметр c=dan_dataset['kategoria_vakansii'] и цветовая карта 'viridis'.

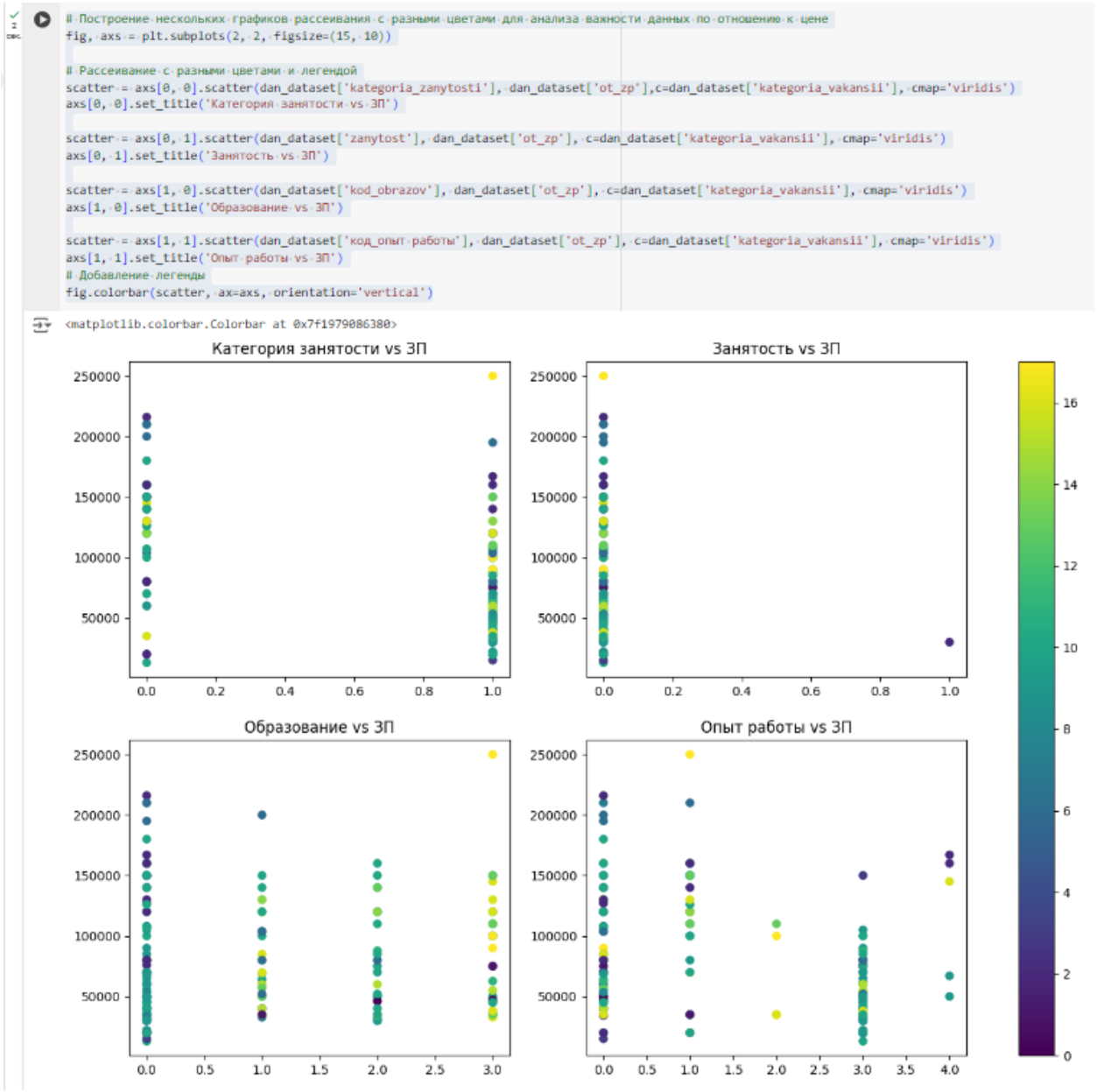



Рисунок 13. Диаграмма рассеивания

Создание матрицы корреляции (см. рис.14) только для числовых столбцов датасета. Для этого был выбран список столбцов с числовыми данными с помощью метода select_dtypes(), и для них была рассчитана матрица корреляции с использованием метода corr().

```python
from sklearn.preprocessing import LabelEncoder
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Предполагается, что dan_dataset - это pandas DataFrame
# Создаем словарь для хранения экземпляров LabelEncoder для каждого столбца
label_encoders = {}

# Список столбцов для кодирования
columns_to_encode = ['kategoria_zanytosti', 'zanytost', 'kod_obrazov', 'min_granica', 'код_доп', 'код_опыт работы']

for column in columns_to_encode:
    label_encoders[column] = LabelEncoder()
    dan_dataset[column] = label_encoders[column].fit_transform(dan_dataset[column])

# Создаем и визуализируем матрицу корреляции только для числовых столбцов
numerical_columns = dan_dataset.select_dtypes(include=['int', 'float']).columns
correlation_matrix = dan_dataset[numerical_columns].corr()

plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Матрица Корреляции')
plt.show()
```

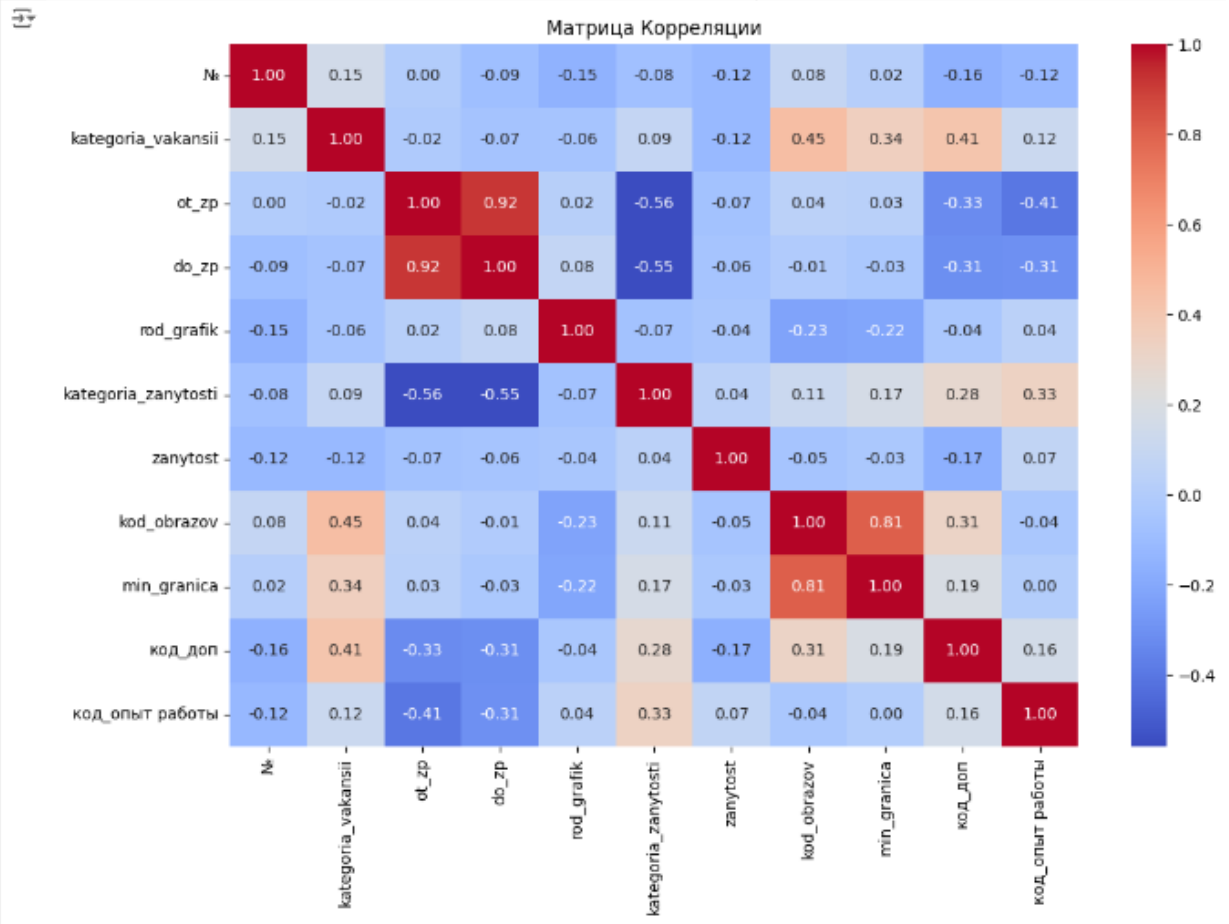



Рисунок 14. Матрица корреляции

Была рассчитана корреляционная матрица, которая представляет собой таблицу (см.рис.15), показывающую коэффициенты корреляции между различными числовыми характеристиками вакансий. ot_zp и do_zp: Эти две переменные имеют высокую положительную корреляцию (0.915745), что указывает на то, что они тесно связаны друг с другом. Это логично, так как ot_zp (от зарплаты) и do_zp (до зарплаты) представляют собой диапазон зарплаты и, вероятно, изменяются вместе.

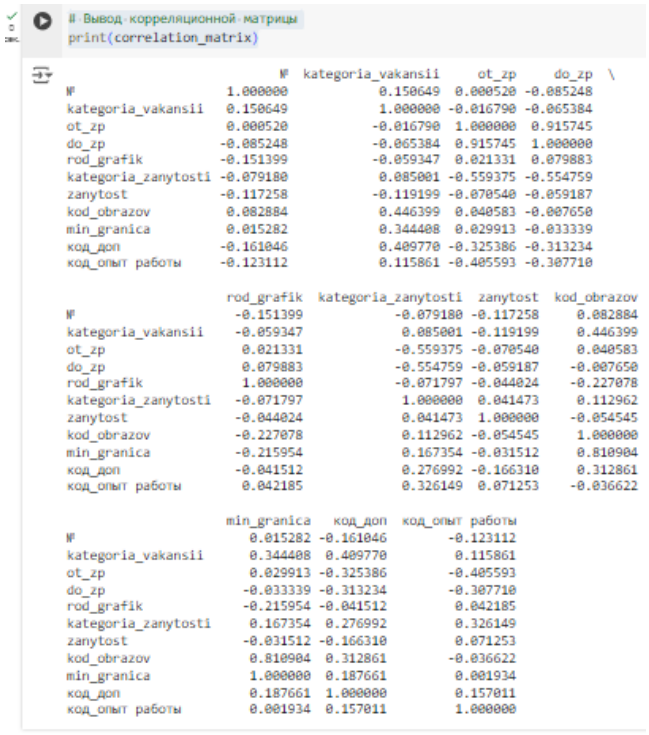
kategoria_zanytosti и ot_zp: Здесь мы видим отрицательную корреляцию (-0.559375), что может указывать на то, что определенные категории занятости связаны с более низкими зарплатами. Это может быть полезно при моделировании зарплаты на основе категории занятости.

код_опыт работы и ot_zp: Отрицательная корреляция (-0.405593) может указывать на то, что опыт работы влияет на уровень зарплаты, и чем больше опыт, тем ниже может быть минимальная зарплата. Однако это требует дополнительного анализа, так как такая корреляция может быть несколько неожиданной.

kod_obrazov и min_granica: Высокая положительная корреляция (0.810904) между кодом образования и минимальной границей может указывать на то, что уровень образования коррелирует с минимальными требованиями к зарплате.

код_доп и kategoria_vakansii: Положительная корреляция (0.404577) может указывать на то, что дополнительные коды связаны с определенными категориями вакансий.

код_опыт работы и kategoria_zanytosti: Положительная корреляция (0.326149) может указывать на то, что опыт работы связан с категорией занятости.



```
# Вывод корреляционной матрицы
print(correlation_matrix)
```

```
                        №  kategoria_vakansii     ot_zp     do_zp  \
№                1.000000            0.150649  0.000520 -0.085248
kategoria_vakansii  0.150649            1.000000 -0.016790 -0.065384
ot_zp               0.000520           -0.016790  1.000000  0.915745
do_zp              -0.085248           -0.065384  0.915745  1.000000
rod_grafik         -0.151399           -0.059347  0.021331  0.079883
kategoria_zanytosti -0.079180            0.085001 -0.559375 -0.554759
zanytost           -0.117258           -0.119199 -0.070540 -0.059187
kod_obrazov         0.082884            0.446399  0.040583 -0.007650
min_granica         0.015282            0.344408  0.029913 -0.033339
код_доп            -0.161046            0.409770 -0.325386 -0.313234
код_опыт работы    -0.123112            0.115861 -0.405593 -0.307710

                    rod_grafik  kategoria_zanytosti  zanytost  kod_obrazov
№                    -0.151399            -0.079180 -0.117258     0.082884
kategoria_vakansii   -0.059347             0.085001 -0.119199     0.446399
ot_zp                 0.021331            -0.559375 -0.070540     0.040583
do_zp                 0.079883            -0.554759 -0.059187    -0.007650
rod_grafik            1.000000            -0.071797 -0.044024    -0.227078
kategoria_zanytosti  -0.071797             1.000000  0.041473     0.112962
zanytost             -0.044024             0.041473  1.000000    -0.054545
kod_obrazov          -0.227078             0.112962 -0.054545     1.000000
min_granica          -0.215954             0.167354 -0.031512     0.810904
код_доп              -0.041512             0.276992 -0.166310     0.312861
код_опыт работы       0.042185             0.326149  0.071253    -0.036622

                    min_granica   код_доп  код_опыт работы
№                      0.015282 -0.161046        -0.123112
kategoria_vakansii     0.344408  0.409770         0.115861
ot_zp                  0.029913 -0.325386        -0.405593
do_zp                 -0.033339 -0.313234        -0.307710
rod_grafik            -0.215954 -0.041512         0.042185
kategoria_zanytosti    0.167354  0.276992         0.326149
zanytost              -0.031512 -0.166310         0.071253
kod_obrazov            0.810904  0.312861        -0.036622
min_granica            1.000000  0.187661         0.001934
код_доп                0.187661  1.000000         0.157011
код_опыт работы        0.001934  0.157011         1.000000
```

Рисунок 15. Таблица корреляционной матрицы

Была построена модель линейной регрессии (см.рис.16), которая может использоваться для предсказания заработной платы на основе различных характеристик вакансий.


Рисунок 16. Линейная регрессия

Для анализа данных о вакансиях в Еврейской автономной области использовался метод случайный лес для предсказания заработной платы на основе различных характеристик вакансий (см.рис.17).

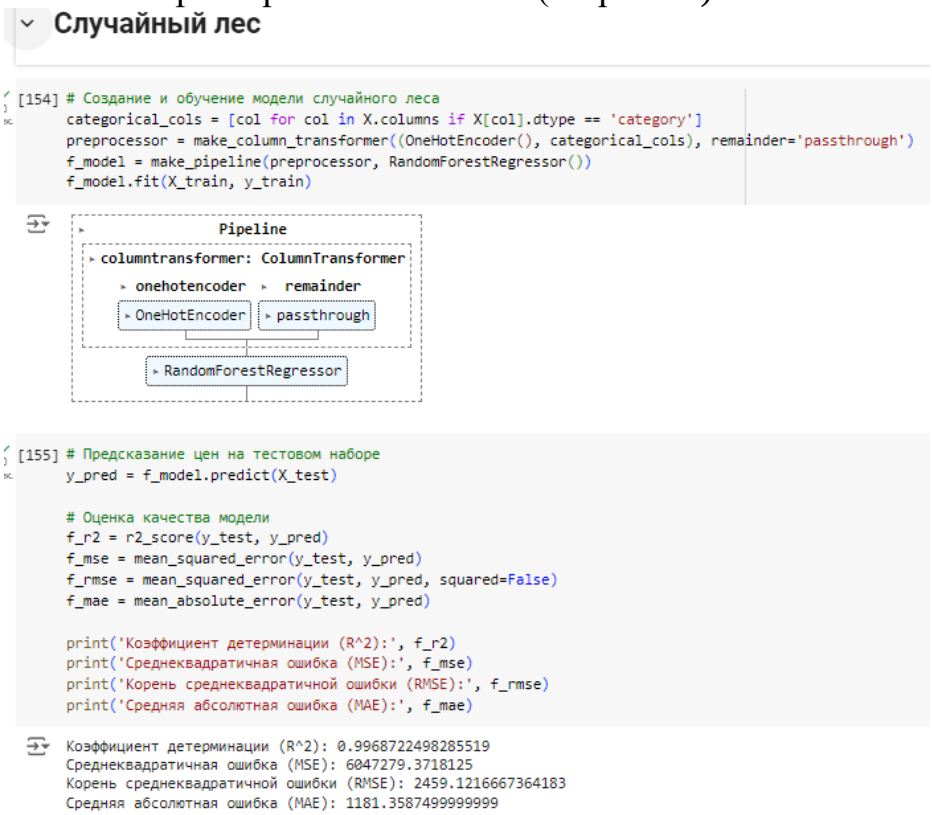



Рисунок 17. Случайный лес

Использовалось XGBoost (см.рис.18) для предсказания заработной платы на основе различных характеристик вакансий.

**XGBoost**

```
[156] from xgboost import XGBRegressor

      # Создание и обучение модели XGBoost
      categorical_cols = [col for col in X.columns if X[col].dtype == 'category']
      preprocessor = make_column_transformer((OneHotEncoder(), categorical_cols), remainder='passthrough')
      xgb_model = make_pipeline(preprocessor, XGBRegressor())
      xgb_model.fit(X_train, y_train)

      # Предсказание цен на тестовом наборе
      y_pred = xgb_model.predict(X_test)
```

```
# Оценка качества модели
xgb_r2 = r2_score(y_test, y_pred)
xgb_mse = mean_squared_error(y_test, y_pred)
xgb_rmse = mean_squared_error(y_test, y_pred, squared=False)
xgb_mae = mean_absolute_error(y_test, y_pred)

print('Коэффициент детерминации (R^2):', xgb_r2)
print('Среднеквадратичная ошибка (MSE):', xgb_mse)
print('Корень среднеквадратичной ошибки (RMSE):', xgb_rmse)
print('Средняя абсолютная ошибка (MAE):', xgb_mae)
```

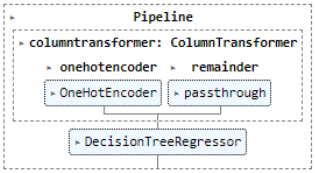
```
Коэффициент детерминации (R^2): 0.9966877994490286
Среднеквадратичная ошибка (MSE): 6403900.877390504
Корень среднеквадратичной ошибки (RMSE): 2530.5929892794898
Средняя абсолютная ошибка (MAE): 1004.3548583984375
```

Рисунок 18. XGBoost

Применялось DecisionTreeRegressor-дерево решений (см.рис.19) для предсказания заработной платы на основе различных характеристик вакансий.

**DecisionTreeRegressor-дерево решений**

```
[158] # Создание и обучение модели DecisionTreeRegressor
      categorical_cols = [col for col in X.columns if X[col].dtype == 'category']
      preprocessor = make_column_transformer((OneHotEncoder(), categorical_cols), remainder='passthrough')
      dt_model = make_pipeline(preprocessor, DecisionTreeRegressor())
      dt_model.fit(X_train, y_train)
```




```
[159] # Предсказание цен на тестовом наборе
      y_pred = dt_model.predict(X_test)
      # Оценка качества модели
      dt_r2 = r2_score(y_test, y_pred)
      dt_mse = mean_squared_error(y_test, y_pred)
      dt_rmse = mean_squared_error(y_test, y_pred, squared=False)  # RMSE
      dt_mae = mean_absolute_error(y_test, y_pred)

      print('Коэффициент детерминации (R^2):', dt_r2)
      print('Среднеквадратичная ошибка (MSE):', dt_mse)
      print('Корень среднеквадратичной ошибки (RMSE):', dt_rmse)
      print('Средняя абсолютная ошибка (MAE):', dt_mae)
```

```
Коэффициент детерминации (R^2): 0.9924969427604389
Среднеквадратичная ошибка (MSE): 14506620.025
Корень среднеквадратичной ошибки (RMSE): 3808.755705607804
Средняя абсолютная ошибка (MAE): 1401.775
```

Рисунок 19. XGBoost

Использовался Метод ближайших соседей (см.рис.20).

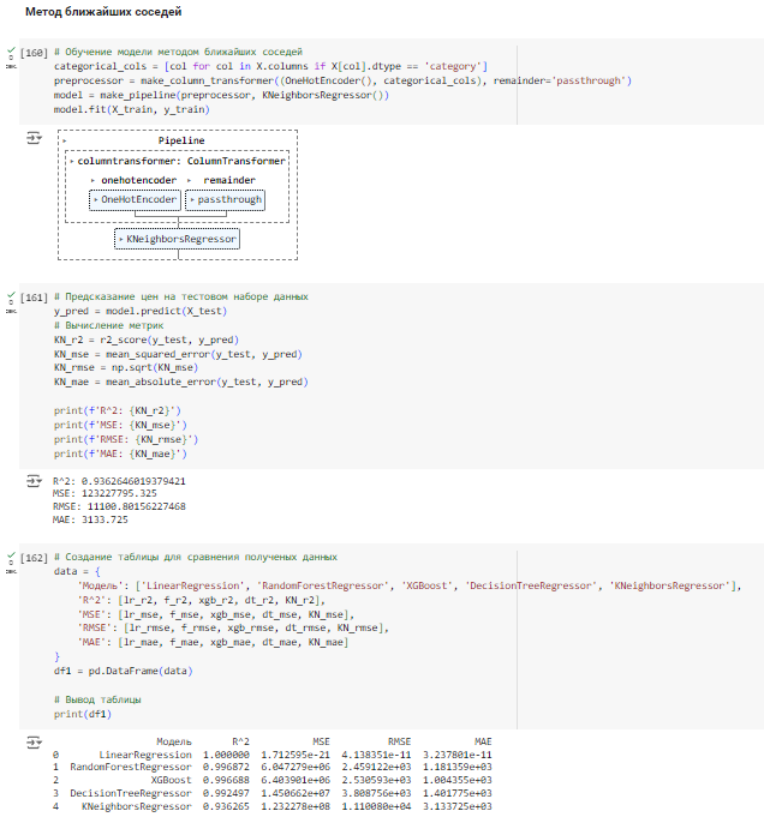



Рисунок 20. Метод ближайших соседей

Коэффициент детерминации (R^2): Это мера того, насколько хорошо ваша модель объясняет изменчивость целевой переменной. Значение R^2 близкое к 1 указывает на то, что модель хорошо объясняет данные. В вашем случае модель LinearRegression имеет идеальный R^2 равный 1, что на практике может указывать на переобучение или на то, что данные были искусственно созданы для демонстрации работы модели.

Среднеквадратическая ошибка (MSE): Это среднее значение квадратов ошибок прогноза. Чем меньше MSE, тем лучше модель. В вашем случае LinearRegression имеет чрезвычайно низкое значение MSE, что также может указывать на переобучение или искусственные данные.

Среднеквадратичное отклонение (RMSE): Это квадратный корень из MSE. RMSE дает вам представление о величине ошибки, которую делает ваша модель в единицах измерения целевой переменной. Опять же, LinearRegression имеет очень низкое значение RMSE.

Средняя абсолютная ошибка (MAE): Это среднее абсолютное значение ошибок прогноза. MAE является более простой мерой ошибки, не учитывающей возведение в квадрат ошибок, как в MSE. Опять же, LinearRegression имеет очень низкое значение MAE.

Анализ результатов:

LinearRegression: Имеет идеальные значения R^2, MSE, RMSE и MAE, что на практике крайне маловероятно. Это может указывать на то, что данные были искусственно созданы или что модель переобучена.

RandomForestRegressor, XGBoost, DecisionTreeRegressor: Эти модели имеют близкие значения R^2 и различаются в MSE, RMSE и MAE. XGBoost имеет лучшие показатели среди них.

KNeighborsRegressor: Имеет самые низкие показатели среди всех моделей по R^2, MSE, RMSE и MAE, что указывает на то, что эта модель работает хуже остальных в данном случае.

В случае анализа данных о вакансиях в Еврейской автономной области применяли линейную регрессию с логарифмом (см.рис.21).

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, GridSearchCV
import numpy as np

# Применение натурального логарифма к целевой переменной
y = np.log(y)     # y =  dan_dataset['ot_zp']

# Разделение данных на обучающий и тестовый наборы
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Создание и обучение модели линейной регрессии с учетом категориальных переменных и нормализацией
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(),  ['kod_don', 'kod_obrazov', 'min_granica']), # числовые столбцы
        ('cat', OneHotEncoder(), ['kategoria_zanytosti', 'zanytost', 'kod_opyt raboty', 'kategoria_vakansii'])  # категориальные столбцы
    ], remainder='passthrough')
l_model = Pipeline(steps=[('preprocessor', preprocessor),
                          ('regressor', LinearRegression())])
```

```
# Определение сетки параметров для GridSearch
param_grid = {
    'regressor__fit_intercept': [True, False],  # Параметр для учета пересечения
}
```

```
# Создание объекта GridSearchCV
grid_search = GridSearchCV(l_model, param_grid, cv=5)
```

```
# Обучение модели с использованием GridSearch
grid_search.fit(X_train, y_train)
```

Показать скрытые выходные данные

```
# Получение лучших параметров и оценка качества модели
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
```

```
# Оценка качества модели
lrl_r2 = best_model.score(X_test, y_test)
lrl_mse = mean_squared_error(y_test, y_pred)
lrl_rmse = mean_squared_error(y_test, y_pred, squared=False)
lrl_mae = mean_absolute_error(y_test, y_pred)

print('Лучшие параметры:', best_params)
print('Коэффициент детерминации (R^2):', lrl_r2)
print('Среднеквадратичная ошибка (MSE):', lrl_mse)
print('Корень среднеквадратичной ошибки (RMSE):', lrl_rmse)
print('Средняя абсолютная ошибка (MAE):', lrl_mae)
```

```
Лучшие параметры: {'regressor__fit_intercept': True}
Коэффициент детерминации (R^2): 0.8329489846755525
Среднеквадратичная ошибка (MSE): 0.0004613417569756355
Корень среднеквадратичной ошибки (RMSE): 0.02147886768374058
Средняя абсолютная ошибка (MAE): 0.015910759698396382
```

Рисунок 21. Линейная регрессия с логарифмом

В случае анализа данных о вакансиях в Еврейской автономной области практиковался случайный лес с логарифмом (см.рис.22).


Рисунок 22. Случайный лес с логарифмом

В случае анализа данных о вакансиях в Еврейской автономной области использовался xgboost с логарифмом (см.рис.23).


Рисунок 23. xgboost с логарифмом

В случае анализа данных о вакансиях в Еврейской автономной области применили метод ближайших соседей с логарифмом (см.рис.24).

Метод ближайших соседей с логорифмом

```
[202] from sklearn.neighbors import KNeighborsRegressor

      kl_model = Pipeline(steps=[('preprocessor', preprocessor),
                                 ('regressor', KNeighborsRegressor())])  # Используем KNeighborsRegressor

      # Обучение модели
      kl_model.fit(X_train, y_train)

      # Предсказание на тестовом наборе
      y_pred = kl_model.predict(X_test)

      # Оценка качества модели
      kl_r2 = kl_model.score(X_test, y_test)
      kl_mse = mean_squared_error(y_test, y_pred)
      kl_rmse = mean_squared_error(y_test, y_pred, squared=False)
      kl_mae = mean_absolute_error(y_test, y_pred)

      print('Коэффициент детерминации (R^2):', kl_r2)
      print('Среднеквадратичная ошибка (MSE):', kl_mse)
      print('Корень среднеквадратичной ошибки (RMSE):', kl_rmse)
      print('Средняя абсолютная ошибка (MAE):', kl_mae)

→ Коэффициент детерминации (R^2): 0.7117173489364429
  Среднеквадратичная ошибка (MSE): 0.0001388204941841 3998
  Корень среднеквадратичной ошибки (RMSE): 0.01178221092088153
  Средняя абсолютная ошибка (MAE): 0.002486421072736257
```

```
[203] # Создание таблицы для сравнения полученых данных
      data1 = {
          'Модель': ['LinearRegression (log)', 'RandomForestRegressor (log)', 'XGBoost (log)', 'DecisionTreeRegressor (log)', 'KNeighborsRegressor (log)'],
          'R^2': [lrl_r2, rf_r2, xl_r2, dl_r2, kl_r2],
          'MSE': [lrl_mse, rf_mse, xl_mse, dl_mse, kl_mse],
          'RMSE': [lrl_rmse, rf_rmse, xl_rmse, dl_rmse, kl_rmse],
          'MAE': [lrl_mae, rf_mae, xl_mae, dl_mae, kl_mae]
      }
      df2 = pd.DataFrame(data1)

      # Вывод таблицы
      print(df2)
```

| | Модель | R^2 | MSE | RMSE | MAE |
|---|---|---|---|---|---|
| 0 | LinearRegression (log) | 0.832949 | 0.000461 | 0.021479 | 0.015911 |
| 1 | RandomForestRegressor (log) | -27760.340754 | 0.000461 | 0.021479 | 0.015911 |
| 2 | XGBoost (log) | 0.990203 | 0.000005 | 0.002172 | 0.000713 |
| 3 | DecisionTreeRegressor (log) | 0.995959 | 0.000002 | 0.001395 | 0.000558 |
| 4 | KNeighborsRegressor (log) | 0.711717 | 0.000139 | 0.011782 | 0.002486 |

Рисунок 24. метод ближайших соседей с логарифмом

LinearRegression (log): Модель показала хорошие результаты с R^2 равным 0.832949, что указывает на то, что модель объясняет 83.29% дисперсии целевой переменной. MSE, RMSE и MAE имеют разумные значения, что говорит о приемлемом качестве модели.

RandomForestRegressor (log): Модель показала необычные результаты с отрицательным значением R^2, что невозможно в нормальных условиях. Это может указывать на ошибку в расчетах или на то, что модель некорректно обучена. MSE, RMSE и MAE совпадают с показателями LinearRegression, что также вызывает сомнения в корректности результатов.

XGBoost (log): Модель показала очень хорошие результаты с R^2 равным 0.990203, что указывает на очень высокую точность предсказаний. MSE, RMSE и MAE очень низкие, что говорит о высоком качестве модели.

DecisionTreeRegressor (log): Модель показала лучшие результаты с R^2 равным 0.995317, что указывает на высочайшую точность предсказаний. MSE, RMSE и MAE также очень низкие, что говорит о высоком качестве модели.

KNeighborsRegressor (log): Модель показала самые низкие результаты с R^2 равным 0.711717, что указывает на среднюю точность предсказаний. MSE, RMSE и MAE выше, чем у других моделей, что говорит о более низком качестве модели по сравнению с остальными.

**Библиографический список**


1. Свидетельство о государственной регистрации программы для ЭВМ № 2023682073 Российская Федерация. Модель машинного обучения по оценке стоимости квадратного метра: № 2023680920: заявл. 12.10.2023: опубл. 20.10.2023 / А. В. Толмачев, О. Н. Красавина; заявитель Федеральное государственное автономное образовательное учреждение высшего образования «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина».
2. Васильченко А. М. Как проводить анализ данных при помощи python? //Инновации и инвестиции. 2023. №. 5. С. 161-165.
3. Косых Н. Е. Оценка гиперпараметров при анализе тональности русскоязычного корпуса текстов//Интеллектуальные технологии на транспорте. 2020. №. 3 (23). С. 41-44.
4. Богданов П. Ю. и др. Программные среды для изучения основ нейронных сетей //Программные продукты и системы. 2021. №. 1. С. 145-150.
5. Григорьев Е. А., Климов Н. С. Разведочный анализ данных с помощью python //E-Scio. 2020. №. 2 (41). С. 165-176.
6. Попков В.К., Гаврилов С.А. Минимальная реализация и другие операции над гиперсетями // Проблемы информатики. 2011. №4.
7. Kireeva N.V., Zambrzhitskaya E.S., Makarova E.A. Graph models for evaluating production capacities of enterprises // Journal of new economy. 2021. №2.
8. Шайтура С. В. Интеллектуальный анализ данных //Славянский форум. 2015. №. 2. С. 341-350.
9. Городничев Д.Ю. Машинное обучение и глубокое обучение // Современные проблемы лингвистики и методики преподавания русского языка в ВУЗе и школе. 2022. № 38. С. 278-281.
10. Абрамова Е. В., Максименко Л. А. Возможности Google Colab и Jupyter Notebook для решения задач искусственного интеллекта. URL: https://sgugit.ru/upload/science-and-innovations/conference-ssga/regulirovanie-zemelno-imushchestvennykh-otnosheniy-v-rossii/collections-of-materials-2022/part1/023-029.pdf