Федеральное государственное автономное

образовательное учреждение высшего образования

«Пермский государственный национальный исследовательский университет» (ПГНИУ)

Региональный институт непрерывного образования (РИНО ПГНИУ)

Цифровая кафедра

Выпускная аттестационная (квалификационная) работа

по курсу профессиональной переподготовки «Анализ данных»

**Прогнозирование зарплат сотрудников**

Разработчики проекта:

Токар Анна Александровна,

Вячин Олег Сергеевич,

Красников Андрей Александрович

Пермь, 2024

**Оглавление**

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

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

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

[Исходные данные 7](#_heading=h.2et92p0)

[Реализация проекта 8](#_heading=h.tyjcwt)

[Этап 1. Подготовка данных к анализу 8](#_heading=h.3dy6vkm)

[Этап 2. Предварительный анализ данных 12](#_heading=h.1t3h5sf)

[Этап 3. Корреляционный анализ данных 2](#_heading=h.4d34og8)3

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

[Этап 5. Прогнозирование 2](#_heading=h.17dp8vu)6

[Заключение 3](#_heading=h.3rdcrjn)8

[Список использованных источников и литературы 3](#_heading=h.26in1rg)9

[Приложения](#_heading=h.lnxbz9) 40

**ПАСПОРТ ПРОЕКТА**

**Название проекта:**Прогнозирование зарплат сотрудников.

**Сведения об авторах:**Токар Анна Александровна, Вячин Олег Сергеевич, Красников Андрей Александрович.

**Цель:**выполнить анализ данных о сотрудниках и построить модель зависимости заработной платы от различных факторов с помощью различных методов регрессии.

**Задачи:**

1. Обосновать актуальность проблемы.
2. Загрузить данные и подготовить их для анализа и прогнозирования.
3. Выполнить анализ данных, проверить наличие выбросов, оценить распределение данных на нормальность и корреляционный анализ факторов, влияющих на зарплату сотрудников.
4. Рассмотреть модели путем тестирования различных признаков и гиперпараметров, оценить их качество путем использования метрик и выбрать наилучшую.
5. Интерпретировать результаты и сформулировать выводы о степени влияния факторов на заработную плату сотрудников.

**Краткое описание проекта:**

Проект посвящён анализу данных о сотрудниках и прогнозированию заработной платы. Проведены обработка данных, анализ факторов, влияющих на зарплату, и тестирование моделей регрессии. Выбрана оптимальная модель для точного прогнозирования и оценки влияния факторов.

**Конкретные ожидаемые результаты:**

Построенная регрессионная модель, которая способна достаточно точно прогнозировать зарплаты сотрудников.

**СОДЕРЖАНИЕ ПРОЕКТА**

**Анализ проблемы исследования**

Основная цель оплаты труда – обеспечить объективно необходимое воспроизводство рабочей силы в соответствии с ее стоимостью и повысить уровень мотивации исполнителей к эффективному труду.

Объективной необходимостью и важным фактором повышения эффективности производства является создание у трудоспособных граждан высоких мотивов и стимулов к труду. Достигается это путем широкой дифференциации оплаты труда исходя из уровня квалификации и личного трудового вклада в производство. Для решения этой задачи хозяйственные субъекты могут самостоятельно разрабатывать и устанавливать формы и системы оплаты труда, критерии оценки трудового вклада соответствующих исполнителей. Для тех предприятий, организаций, учреждений, финансирование которых осуществляется из бюджета, данные функции возложены на органы государственного управления.

Оплата труда является важным инструментом управления персоналом, позволяющим обеспечивать баланс между интересами работников и работодателей. Система оплаты труда должна быть справедливой, прозрачной и учитывать профессиональные навыки, опыт, сложность выполняемых задач и результаты деятельности сотрудников.

Правильно выстроенная система оплаты труда способствует привлечению, удержанию и мотивации квалифицированных специалистов. Она стимулирует рост производительности труда, способствует повышению качества продукции и услуг, а также увеличивает конкурентоспособность предприятия на рынке.

В современных условиях ключевую роль играет использование аналитических методов для оценки факторов, влияющих на заработную плату. Анализ данных позволяет объективно оценить трудовой вклад сотрудников, минимизировать субъективные факторы при назначении зарплат и повысить прозрачность процессов принятия решений.

Для оценки факторов, влияющих на заработную плату, важно учитывать как объективные, так и субъективные показатели. Объективные показатели включают квалификацию, стаж, уровень образования и занимаемую должность, в то время как субъективные могут включать результаты аттестаций, оценки руководства и индивидуальные достижения сотрудников.

Точные прогнозы заработной платы требуют комплексного подхода, включающего сбор и обработку большого объема данных. Это позволяет выявить тенденции, установить зависимости между признаками и создать модели, способные предсказывать уровень зарплаты на основе различных факторов.

В 2023 году в России наблюдался значительный рост заработных плат. По данным Росстата, средняя номинальная начисленная заработная плата составила 73 709 рублей в месяц, что на 14,1% больше по сравнению с предыдущим годом. Реальная заработная плата, скорректированная с учётом инфляции, увеличилась на 7,8%, что является максимальным показателем за последние пять лет. Медианная заработная плата, отражающая уровень дохода среднего работника, составила 46 751 рубль, показав рост на 15,8% по сравнению с 2022 годом. Наибольший рост зарплат отмечен в отраслях, связанных с импортозамещением, таких как производство металлоизделий, компьютеров и транспортных средств.

Эти данные свидетельствуют о позитивных тенденциях на рынке труда России в 2023 году, что подчёркивает важность анализа факторов, влияющих на заработную плату, для разработки эффективных стратегий управления персоналом.

Таким образом, статистические данные за 2023 год демонстрируют значительное повышение уровня заработных плат в России, что подчёркивает актуальность изучения факторов, влияющих на их формирование. Применение методов анализа данных и построение моделей прогнозирования позволяют не только выявить ключевые детерминанты оплаты труда, но и обеспечить прозрачность и обоснованность управленческих решений. Это способствует повышению мотивации сотрудников, снижению текучести кадров и укреплению конкурентоспособности организаций на рынке труда.

**Исходные данные**

В данной работе рассматриваются различные факторы, которые могут отражать зарплату сотрудников и влиять на нее.

Список колонок анализируемого набора данных:

1. **Age** – возраст сотрудника.
2. **Gender** – пол сотрудника.
3. **Education Level** – образование сотрудника.
4. **Job Title** – название работы.
5. **Years of Experience** – количество лет опыта работы.
6. **Salary** – целевой признак, текущая зарплата сотрудника.

Необходимо проанализировать данные о сотрудниках и определить, возможно ли описать зависимость заработной платы от имеющихся факторных переменных с использованием методов регрессии.

Выдвинем гипотезу исследования: заработная плата сотрудника зависит от его возраста, пола, уровня образования, названия должности и количества лет профессионального опыта. Эти показатели могут быть использованы для построения модели регрессии, способной предсказывать уровень заработной платы с высокой точностью.

**Реализация проекта**

**Этап 1. Подготовка данных к анализу**

Импортируем необходимые библиотеки для дальнейшей работы, а также установим настройки:

import pandas as pd

import numpy as np

import plotly.express as px

from plotly.offline import iplot

import seaborn as sns

import matplotlib.pyplot as plt

sns.set\_palette('husl')

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

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

from sklearn import model\_selection, preprocessing, feature\_selection, ensemble, linear\_model, metrics, decomposition

import warnings

warnings.filterwarnings("ignore")

pd.set\_option('future.no\_silent\_downcasting', True)

pd.options.mode.copy\_on\_write = "warn"

from google.colab import drive

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

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

from sklearn import metrics

from collections import Counter

Теперь напишем код функции, которая будет вычислять все необходимые метрики для оценки работы моделей:

def Reg\_Models\_Evaluation\_Metrics (model,X\_train,y\_train,X\_test,y\_test,y\_pred):

    cv\_score = cross\_val\_score(estimator = model, X = X\_train, y = y\_train, cv = 10)

    # Calculating Adjusted R-squared

    r2 = model.score(X\_test, y\_test)

    # Number of observations is the shape along axis 0

    n = X\_test.shape[0]

    # Number of features (predictors, p) is the shape along axis 1

    p = X\_test.shape[1]

    # Adjusted R-squared formula

    adjusted\_r2 = 1-(1-r2)\*(n-1)/(n-p-1)

    RMSE = np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred))

    R2 = model.score(X\_test, y\_test)

    CV\_R2 = cv\_score.mean()

    print(f'RMSE:  {round(RMSE,4)}\n  R2: {round(R2,4)}\n Adjusted R2: {round(adjusted\_r2, 4)}\n Cross Validated R2: {round(cv\_score.mean(),4)}')

    return R2, adjusted\_r2, CV\_R2, RMSE

Загрузим датасет и выведем первые 5 записей:

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

table = pd.read\_csv('/content/drive/MyDrive/Salary Data.csv', encoding='utf-8', sep=',')

table.head()

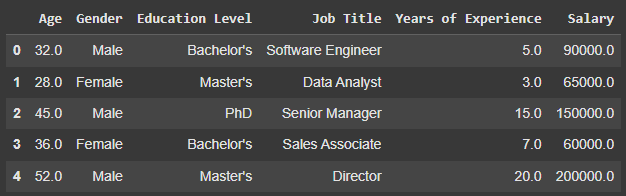


Рисунок 1. Исходный датасет

Укажем код функций, которые помогают при создании холста графиков:

def add\_line(x0 = 0, y0 = 0, x1 = 0, y1 = 0,

             line\_color = "#00DFA2", font\_color = "#3C486B",

             xposition = "right", text = "Text"):

    fig.add\_shape(type='line',

                  x0 = x0,

                  y0 = y0,

                  x1 = x1,

                  y1 = y1 + 2,

                  line = {

                      "color" : line\_color,

                      "width" : 3,

                      "dash" : "dashdot"

                  },

                  label={

                      "text" : f"\t{text}: {x1: 0.1f}\t".expandtabs(5),

                      "textposition": "end",

                      "yanchor" :"top",

                      "xanchor" :xposition,

                      "textangle" :0,

                      "font": {

                          "size": 14,

                          "color" :font\_color,

                          "family" : "arial"

                      },

                  }

                 )

def custome\_layout(title\_size = 28, hover\_font\_size = 16, showlegend = False):

    fig.update\_layout(

    showlegend = showlegend,

    title = {

        "font" :{

            "size" :title\_size,

            "family" : "tahoma"

        }

    },

    hoverlabel = {

        "bgcolor" :"#111",

        "font\_size" : hover\_font\_size,

        "font\_family" :"arial"

    }

)

Посмотрим на полученную информацию в датасете, а также глянем как работает стиль:

table.info()

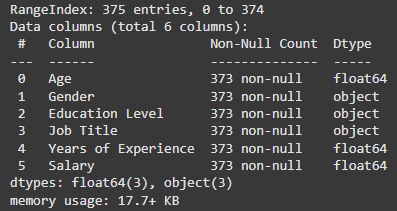


Рисунок 2. Типы данных колонок

Видим, что столбцы Age, Years of Experience и Salary числовые, когда все остальные являются строковыми. Также можно заметить, что присутствуют пустые значения, от которых нужно будет избавиться в дальнейшем.

Оценим транспонированную таблицу с основными статистическими характеристиками числовых колонок:

table.describe().T

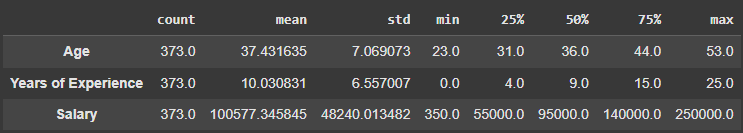


Рисунок 3. Описание числовых колонок датасета

По количественным столбцам была построена таблица со следующими параметрами:

1. **count** – количество непустых значений в колонке;
2. **mean** – среднее арифметическое значений;
3. **std** – стандартное отклонение;
4. **min** – минимальное значение;
5. **25%** - первый квартиль (нижние 25% значений);
6. **50%** - медиана, срединное значение;
7. **75%** - третий квартиль (верхние 25% значений);
8. **max** - максимальное значение.

Следующий шаг – удалить пустые значения:

table.dropna(inplace=True)

Далее удаляем дубликаты и сбрасываем индекс:

table.drop\_duplicates(inplace=True)

table.reset\_index(inplace=True, drop=True)

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

**Этап 2. Предварительный анализ данных**

Проведем исследования по колонкам. Разберем каждый фактор и построим графики распределения.

Посмотрим на график распределения по возрастам, для этого применим график «ящик с усами» для нахождения выбросов:

mean\_of\_age = table["Age"].mean()

median\_of\_age = table["Age"].median()

fig = px.box(

    y=table["Age"],

    title= "Распределение по возрастам",

    template="plotly\_dark",

    labels={"y" :"Возраст"},

)

custome\_layout()

iplot(fig)

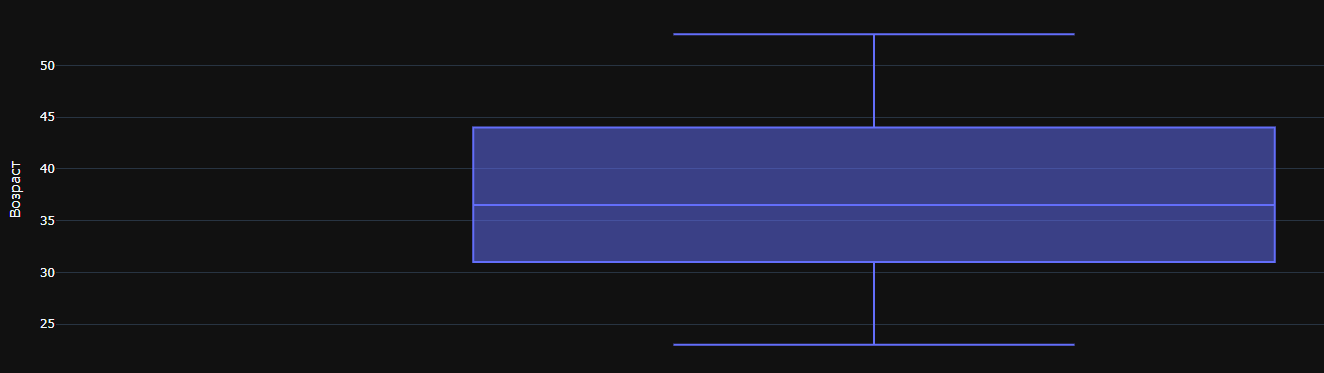


Рисунок 4. Распределение по возрастам

По графику можно заметить, что выбросов в столбце с возрастом нет.

Теперь построим гистограмму для более ясного понимания распределения возраста:

fig = px.histogram(

    table["Age"],

    nbins=25,

    title= "Распределение по возрастам",

    template="plotly\_dark",

    labels={"value" :"Возраст"}

)

custome\_layout()

fig.update\_traces(

    textfont = {

        "size" : 20,

        "family" :"tahoma",

        "color": "#fff"

    },

    hovertemplate = "Возраст: %{x}<br>Количество: %{y}",

    marker=dict(line=dict(color='#000', width=0.1))

)

# Adding Mean Line

add\_line(x0=mean\_of\_age, y0=0, x1=mean\_of\_age, y1=30+2, line\_color="#E97777",font\_color="#E97777",

         text="Mean", xposition="left")

# Adding Median Line

add\_line(x0=median\_of\_age, y0=0, x1=median\_of\_age, y1=30+2, line\_color="#FFE5F1",

         font\_color="#fff", xposition="right", text="Median")

iplot(fig)

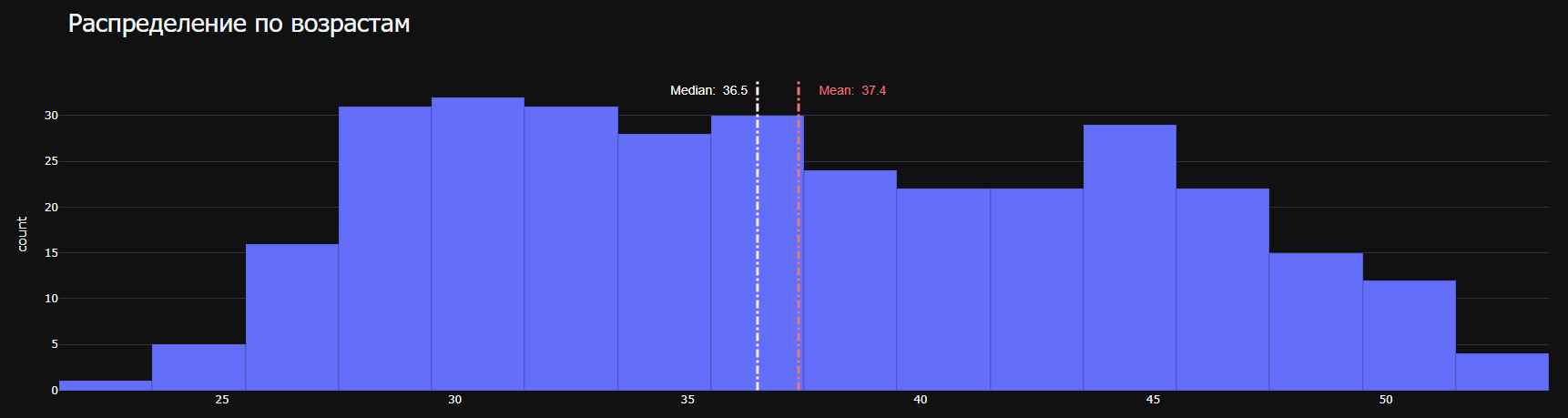


Рисунок 5. Распределение возраста на графике гистограммы

На графике построена медиана, а также среднее значение возраста сотрудников, принявших участие в сборе информации. Средний возраст составил 37.4, когда медиана составила 36.5 лет.

Теперь рассмотрим гендерное соотношение сотрудников:

gender = table["Gender"].value\_counts(normalize=1) \* 100

gender.apply(lambda x: f"{x:0.2f}%")

fig = px.bar(data\_frame = gender,

             x = gender.index,

             y = gender,

             color = gender.index,

             title = "Гендерное соотношение в процентах",

             color\_discrete\_sequence=["#45FFCA", "#FF9B9B"],

             labels= {"index" :"Gender", "y": "Процент(%)"},

             template="plotly\_dark",

             text = gender.apply(lambda x: f"{x:0.0f}%"))

custome\_layout()

fig.update\_traces(

    textfont = {

        "size" : 16,

        "family" :"arial",

        "color": "#222"

    },

    hovertemplate = "Пол: %{x}<br>Процент: %{y:0.1f}%",

)

iplot(fig)

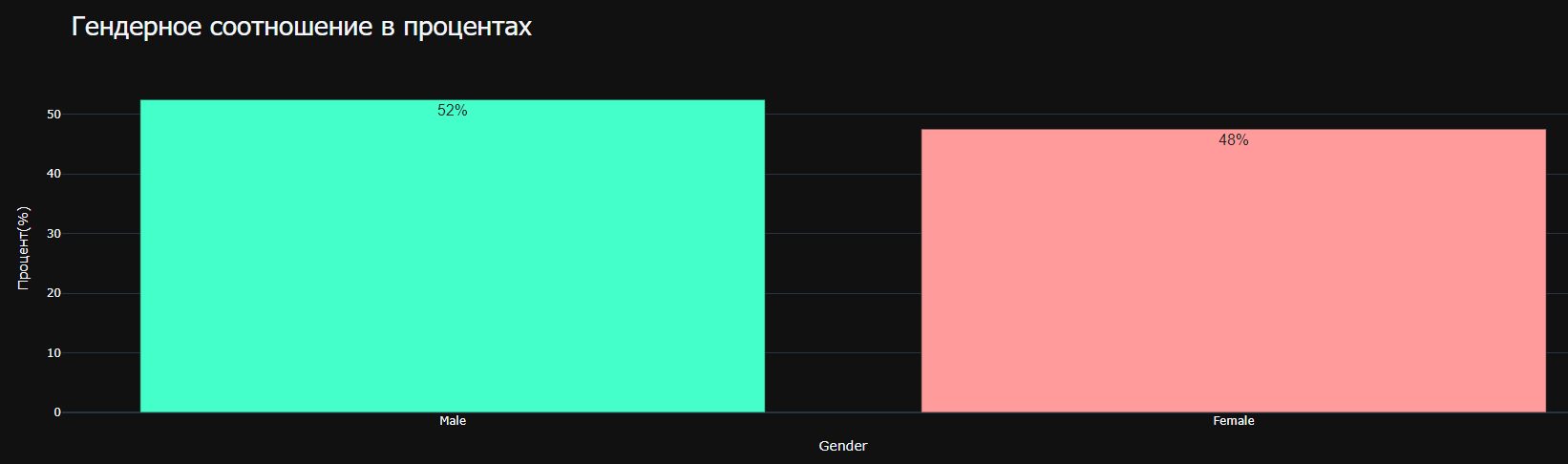


Рисунок 6. Гендерное соотношение сотрудников в процентах

В текущем датасете процент опрошенных мужчин составил 52%, когда женщин 48%. Соотношение мужчин и женщин почти составило 1:1.

Продолжим работать с категориальными данными, рассмотрим соотношение образования в датасете:

education = table["Education Level"].value\_counts(normalize=1) \* 100

education.apply(lambda x: f"{x:0.2f}%")

fig = px.bar(data\_frame = education,

             x = education.index,

             y = education,

             color = education.index,

             title = "Соотношение образования",

             color\_discrete\_sequence=["#45FFCA", "#D09CFA", "#FF9B9B"],

             labels= {"index" :"Education", "y": "Процент по образованию"},

             template="plotly\_dark",

             text = education.apply(lambda x: f"{x:0.0f}%"))

custome\_layout()

fig.update\_traces(

    textfont = {

        "size" : 16,

        "family" :"arial",

        "color": "#222"

    },

    hovertemplate = "Образование: %{x}<br>Проценты: %{y:0.1f}%",

)

iplot(fig)

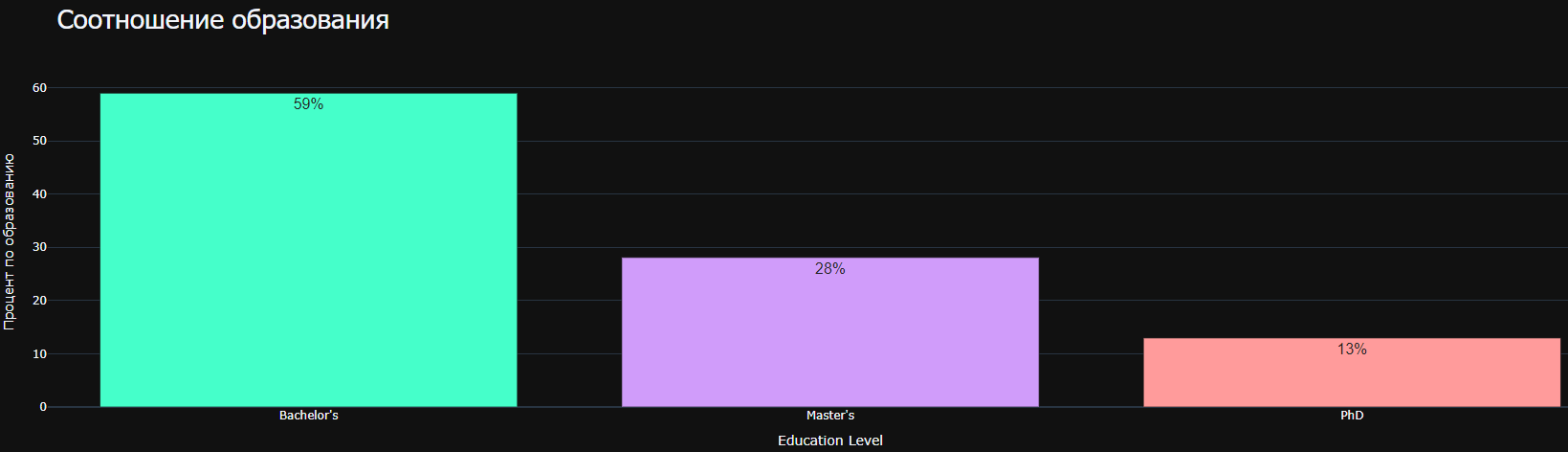


Рисунок 7. Соотношение образования сотрудников

По полученным данным можем заключить, что подавляющее большинство сотрудников (59%) имеет бакалаврское образование, 28% опрошенных сотрудников имеет магистерское образование, 13% опрошенных имеют образование кандидата наук.

Рассмотрим распределение опыта работы сотрудников:

fig = px.box(

    y=table["Years of Experience"],

    title= "Распределение опыта, лет",

    template="plotly\_dark",

    labels={"y" :"Лет опыта"},

)

custome\_layout()

iplot(fig)

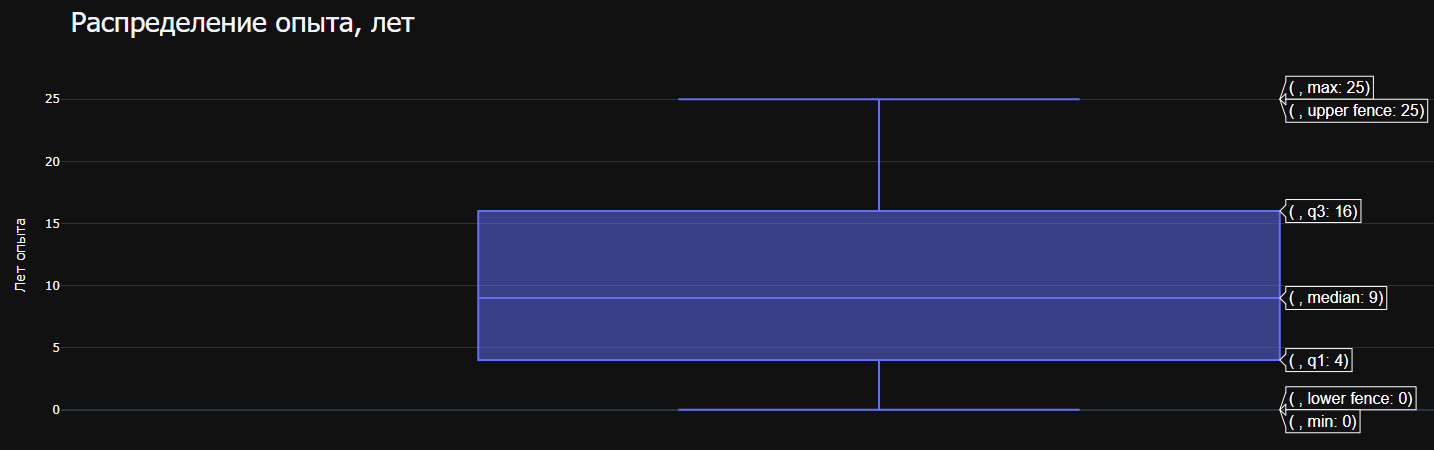


Рисунок 8. Распределение опыта сотрудников

По графику можем заметить, что выбросов также не обнаружено. Это говорит о хорошо подготовленных данных и о том, что все значения находятся в нормальном распределении.

Теперь построим график зависимости зарплаты от уровня образования сотрудников:

fig = px.box(

    x = table["Education Level"], y = table["Salary"],

    title= "Зарплата Vs. Уровень образования",

    template="plotly\_dark",

    labels={"x": "Уровень образования", "y" :"Зарплата"}

)

custome\_layout(hover\_font\_size=13)

iplot(fig)



Рисунок 9. Распределение зарплаты к образованию сотрудника

На графике можем заметить выбросы на квалификации бакалавра и кандидата наук. Также распределение заработной платы сотрудников с квалификацией магистра и кандидата наук довольно близки.

Далее рассмотрим график средней зарплаты в зависимости от пола:

salary\_by\_gender = table.groupby("Gender")["Salary"].mean().sort\_values(ascending=False)

salary\_by\_gender.apply(lambda x: f"${x:,.2f}")

fig = px.bar(data\_frame = salary\_by\_gender,

             x = salary\_by\_gender.index,

             y = salary\_by\_gender,

             color = salary\_by\_gender.index,

             title = "Средняя зарплата по полам",

             color\_discrete\_sequence=["#45FFCA", "#D09CFA", "#FF9B9B"],

             labels= {"index" :"Education", "y": "Частота"},

             template="plotly\_dark",

             text\_auto = "0.4s"

            )

custome\_layout()

fig.update\_traces(

    textfont = {

        "size" : 16,

        "family" :"arial",

        "color": "#222"

    },

    hovertemplate = "Пол: %{x}<br>Частота: $%{y:0.4s}",

)

iplot(fig)

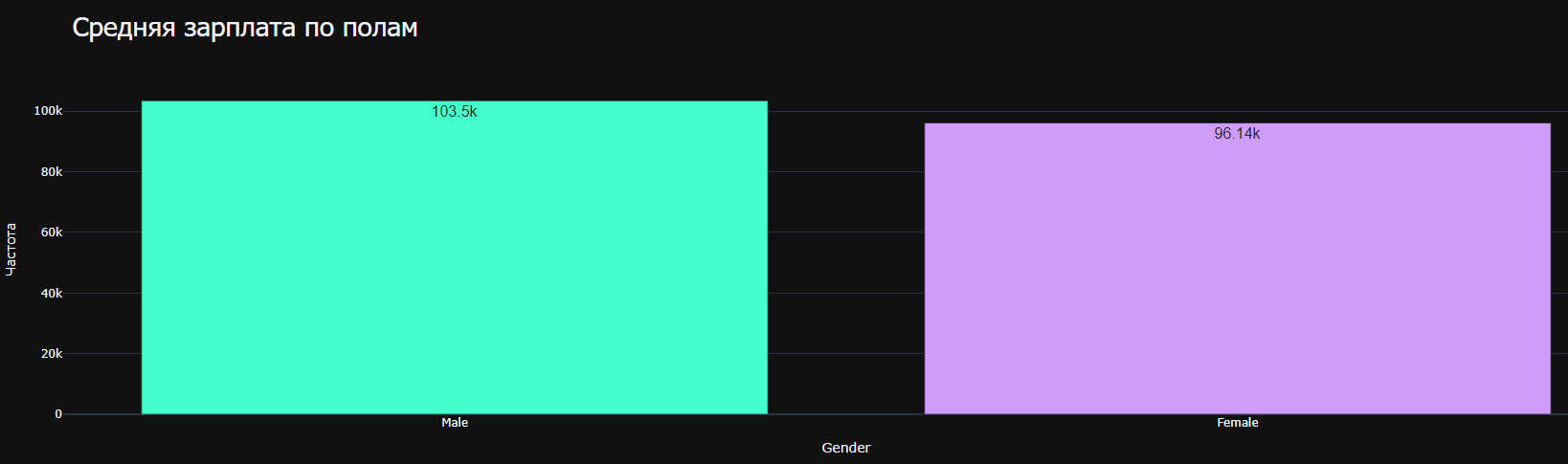


Рисунок 10. Средняя зарплата в зависимости от пола

По полученной информации можем сделать выводы, что средняя заработная плата сотрудников мужского пола больше, чем заработная плата сотрудников женского пола на 7.36 тыс. у.е., что в процентном соотношении составляет разницу в 7.8%.

Теперь рассмотрим влияние уровня образования на заработную плату сотрудников:

salary\_by\_education = table.groupby("Education Level")["Salary"].mean().sort\_values(ascending=False)

salary\_by\_education.apply(lambda x: f"${x:,.2f}")

fig = px.bar(data\_frame = salary\_by\_education,

             x = salary\_by\_education.index,

             y = salary\_by\_education,

             color = salary\_by\_education.index,

             title = "Зарплата в зависимоти от образования",

             color\_discrete\_sequence=["#45FFCA", "#D09CFA", "#FF9B9B"],

             labels= {"index" :"Образование", "y": "Процентное соотношение"},

             template="plotly\_dark",

             text\_auto = "0.4s"

            )

custome\_layout()

fig.update\_traces(

    textfont = {

        "size" : 16,

        "family" :"arial",

        "color": "#222"

    },

    hovertemplate = "Уровень образования: %{x}<br>Average Salary: $%{y:0.4s}",

)

iplot(fig)

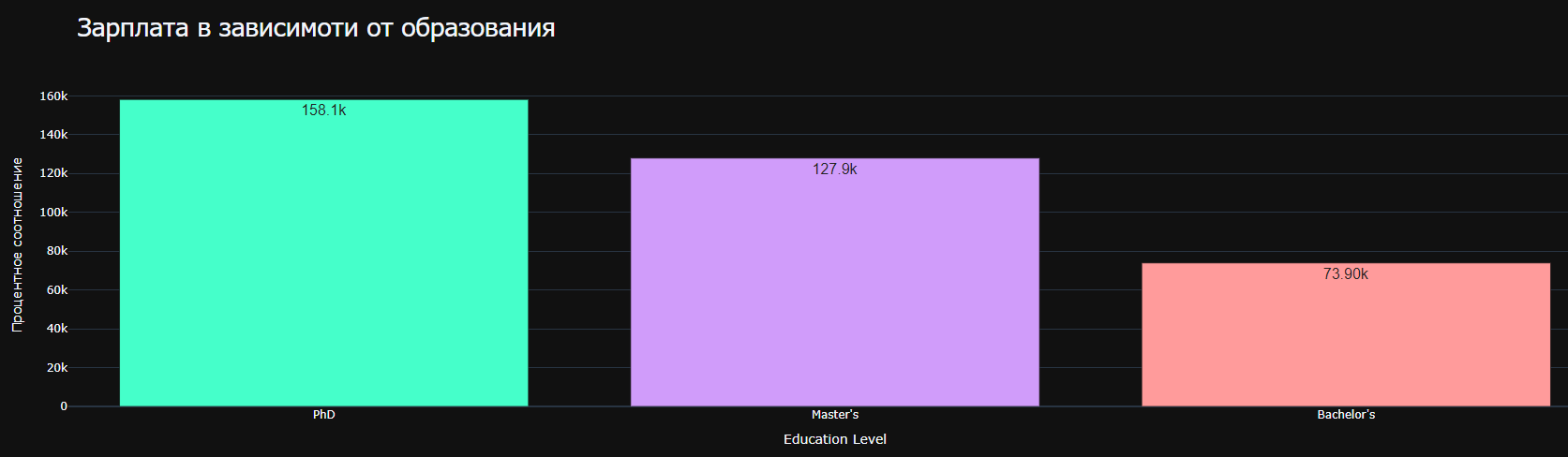


Рисунок 11. Размер заработной платы в зависимости от полученного образования

По графику можем заметить, что среднее значение зарплаты в зависимости от образования значительно выше у кандидатов наук (158.1 тыс. у.е.), у магистров среднее значение составляет 127.9 тыс. у.е., у бакалавров составляет 73.9 тыс. у.е.

Следующим шагом напишем функцию для преобразования опыта в категориальные данные и построим на основе этого преобразования график зависимости заработной платы от опыта:

def groupping\_exp(exp):

    if exp >= 0 and exp <= 5:

        return "0-5 years"

    elif exp > 5 and exp <= 10:

        return "6-10 years"

    elif exp > 10 and exp <= 15:

        return "11-15 years"

    elif exp > 15 and exp <= 20:

        return "16-20 years"

    else:

        return "20+"

salary\_by\_exp = table.groupby(table["Years of Experience"].apply(groupping\_exp))["Salary"].mean().sort\_values(ascending=False)

salary\_by\_exp.apply(lambda x: f"${x:,.2f}")

fig = px.bar(data\_frame = salary\_by\_exp,

             x = salary\_by\_exp.index,

             y = salary\_by\_exp,

             color = salary\_by\_exp.index,

             title = "Зарплата в зависимости от рабочего стажа",

             color\_discrete\_sequence=["#45FFCA", "#D09CFA", "#FF9B9B", "#F875AA", "#3EDBF0"],

             labels= {"index" :"Education", "y": "Зарплата"},

             template="plotly\_dark",

             text\_auto = "0.4s"

            )

custome\_layout()

fig.update\_traces(

    textfont = {

        "size" : 16,

        "family" :"arial",

        "color": "#222"

    },

    hovertemplate = "Рабочий стаж: %{x}<br>Зарплата $%{y:0.4s}",

)

iplot(fig)

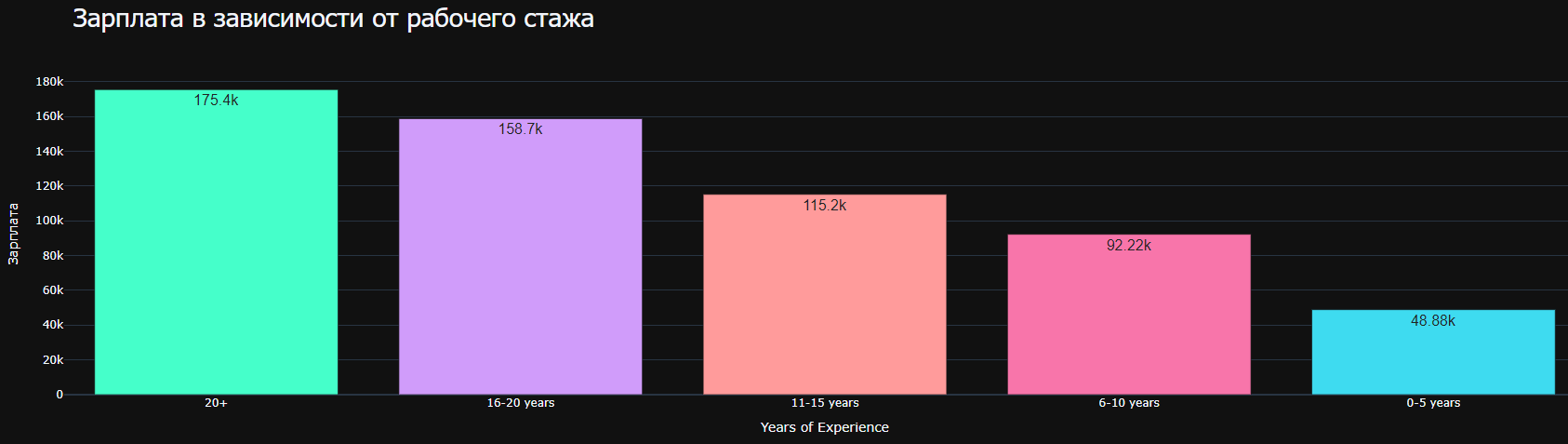


Рисунок 12. Заработная плата в зависимости от рабочего стажа

По результатам графика видим следующие значения: для сотрудника с опытом более 20 лет в среднем выплачивается заработная плата размером 175.4 тыс. у.е., для сотрудников с опытом от 16 до 20 лет заработная плата составляет 158.7 тыс. у.е., для сотрудников с опытом от 11 до 15 лет средняя заработная плата составляет 115.2 тыс. у.е., для сотрудников с опытом работы от 6 до 10 лет заработная плата составляет 92.22 тыс. у.е., для сотрудников с опытом работы менее 6 лет составляет 48.88 тыс. у.е.

Эти данные демонстрируют положительную зависимость между стажем работы и уровнем заработной платы. С увеличением профессионального опыта сотрудники получают более высокие зарплаты, что подтверждает важность накопления знаний и навыков в процессе работы. Наиболее заметный рост заработной платы наблюдается у сотрудников с опытом более 15 лет, что может быть связано с занятием руководящих должностей или высококвалифицированных позиций.

Таким образом, стаж работы выступает одним из ключевых факторов, влияющих на уровень заработной платы.

Следующим шагом в работе будет преобразование категориальных данных в количественные, а затем построение графиков рассеяния зависимости заработной платы от других факторов:

table['Gender'] = pd.factorize(table['Gender'])[0]

table['Education Level'] = pd.factorize(table['Education Level'])[0]

table['Job Title'] = pd.factorize(table['Job Title'])[0]

for index, feature\_name in enumerate(table.columns):

  plt.figure(figsize=(5, 4))

  plt.scatter(table[feature\_name],table['Salary'])

  plt.ylabel("Salary", size=12)

  plt.xlabel(feature\_name, size=12)

  plt.show()

Так как графиков много, выделим самые значимые.

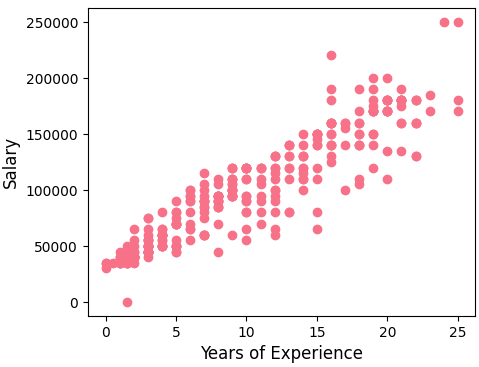


Рисунок 13. Зависимость заработной платы от опыта работы

По графику можем заметить, что с увеличением опыта работы заработная плата сотрудников повышается. Это указывает на прямую зависимость между стажем работы и уровнем зарплаты. Сотрудники с большим опытом ценятся выше, что отражается на их вознаграждении. Однако возможны отклонения, связанные с другими факторами, такими как должность и уровень образования.

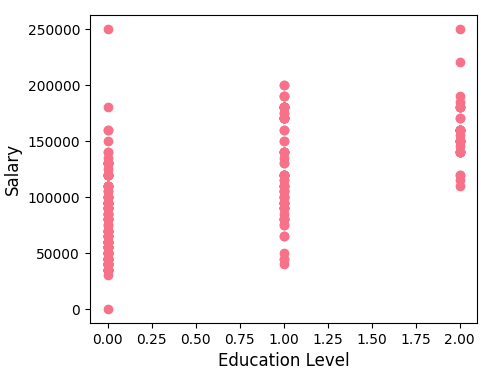


Рисунок 14. Зависимость заработной платы от уровня образования

На графике заметно, что кандидаты наук получают в среднем больше, чем магистры или бакалавры, но также видим и выбросы у бакалавров. Это свидетельствует о влиянии уровня образования на заработную плату: более высокая степень обычно приводит к большему доходу. Тем не менее, наличие выбросов у бакалавров может указывать на успешных специалистов, чьи зарплаты превышают средние значения благодаря другим факторам, таким как опыт или редкие навыки.

Анализ графиков показал, что стаж работы и уровень образования являются важными факторами, влияющими на заработную плату. С увеличением опыта зарплата сотрудников возрастает, а более высокая степень образования в среднем обеспечивает больший доход. Тем не менее, наличие выбросов свидетельствует о том, что на заработную плату могут влиять и другие переменные, такие как профессиональные навыки, должностные обязанности и отрасль занятости.

**Этап 3. Корреляционный анализ данных**

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

Выполним корреляционный анализ данных с помощью матрицы корреляции:

sns.heatmap(table.corr())



Рисунок 15. Тепловая карта матрицы корреляции

Анализ матрицы корреляции показал, что между заработной платой и возрастом, а также между заработной платой и стажем работы наблюдается сильная положительная корреляция (около 0.9). Это подтверждает, что с увеличением профессионального опыта и возраста уровень зарплаты существенно растёт. В то же время корреляция между уровнем образования и заработной платой составляет около 0.7, что также указывает на значительное влияние образования на доход, хотя и менее выраженное по сравнению с возрастом и стажем.

Анализ корреляции показал, что такие признаки, как пол сотрудника (Gender) и название должности (Job Title), имеют низкую корреляцию с уровнем заработной платы. Это указывает на то, что сами по себе эти факторы не оказывают значительного влияния на доход, однако в сочетании с другими переменными они могут вносить дополнительный вклад в формирование заработной платы. Низкие значения корреляции также могут быть связаны с высокой вариативностью должностей и гендерным распределением в различных профессиональных сферах.

**Этап 4. Моделирование**

Следующим этапом будет разбиение датасета на обучающую и тестовую выборки. В качестве целевой переменной выбрана заработная плата сотрудников.

Делаем разбиение:

X = table.drop(['Salary'], axis=1)

y = table['Salary']

print(f'X shape: {X.shape} | y shape: {y.shape} ')

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=15)

Создаем объект KFold для выполненияя кросс-валидации:

kf = KFold(n\_splits=10, shuffle=True, random\_state=30)

Параметр n\_splits=10 указывает, что данные будут разделены на 10 частей. Параметр shuffle=True означает, что данные будут случайным образом перемешаны перед разбиением, чтобы избежать систематической ошибки, если данные имеют порядок. Параметр random\_state=30 задаёт фиксированное начальное значение для генератора случайных чисел, что обеспечивает воспроизводимость результатов. Кросс-валидация помогает более объективно оценить модель, проверяя её на различных подмножествах данных.

Создаем объект RandomForestRegressor, представляющий собой модель случайного леса для задачи регрессии.

rf = RandomForestRegressor(n\_estimators=500, random\_state=11)

Параметр n\_estimators=500 указывает, что модель будет использовать 500 деревьев решений в лесу, что обычно повышает точность, но увеличивает вычислительные затраты.

Выполняем кросс-валидацию модели случайного леса с использованием объекта KFold:

scores = cross\_val\_score(rf, X, y, cv=kf)

print(f"Cross Validation Score: {np.mean(scores)\*100:0.2f}%")

Функция cross\_val\_score оценивает модель, применяя её к данным (X - признаки, y - целевая переменная) на основе 10 фолдов, разделённых с помощью кросс-валидации. Она возвращает массив с оценками модели для каждого фолда.

В результате получили вывод "Cross Validation Score: 88.37%". Это указывает на то, что мы получили довольно хорошую точность модели для регрессионной задачи.

Теперь обучим модель случайного на тренировочных данных:

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

После выполнения метода fit модель готова для прогнозирования на новых данных.

Оценим точность модели случайного леса на тренировочных данных:

score = rf.score(X\_train, y\_train)\*100

print(f"Model Score: {np.round(score, 2)}%")

На выводе получаем 98.34%, это указывает на то, что модель объясняет 98.34% вариации целевой переменной на тренировочных данных, что свидетельствует о высоком качестве модели и ее способности хорошо подстраиваться под обучающий набор данных.

**Этап 5. Прогнозирование**

На данном этапе необходимо рассмотреть какие будут результаты прогнозирования целевой переменной, а также обучить другие виды моделей и сравнить их результаты между собой, выявив наиболее точную модель.

Первым делом спрогнозируем заработную плату сотрудников с помощью нашей обученной модели:

predicted\_salary = np.round(rf.predict(X\_test))

И посмотрим на полученные результаты:

d = {

    "Actual\_Salary" : y\_test,

    "Predicted\_Salary" : predicted\_salary,

    "error": predicted\_salary - y\_test

}

predected\_df = pd.DataFrame(d)

predected\_df.head()

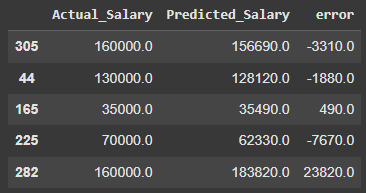


Рисунок 16. Результаты прогнозирования

Пример значений показывает, что ошибки предсказания колеблются в диапазоне от -7670.0 до 23820.0. Большинство ошибок относительно невелики, что свидетельствует о высокой точности модели. Однако присутствуют более крупные отклонения, такие как ошибка в 23820.0, что может указывать на влияние неучтённых факторов или ограничение модели в предсказании высоких зарплат. Это подчеркивает важность дальнейшего анализа ошибок и возможного улучшения модели.

Вычислим коэффициент детерминации R2 для тестового набора данных:

score = r2\_score(y\_test, predicted\_salary)\*100

print(f"Model Score: {np.round(score, 2)}%")

Результат: Model Score: 93.07%. Модель объясняет 93.07% вариации в тестовых данных, что является хорошим показателем для задачи регрессии.

Найдем корень средней квадратичной ошибки (RMSE) – метрику, оценивающую среднее отклонение предсказанных значений от фактических:

rmse = np.sqrt(mean\_squared\_error(y\_test, predicted\_salary))

print(f"Error Ratio: {rmse:.3f}")

Результат: Error Ratio: 14559.607. Это значение указывает на точность модели: чем ниже RMSE, тем лучше. Высокое значение может сигнализировать о том, что модель нуждается в улучшении или что в данных присутствуют значительные отклонения.

Построим график предсказанных значений и текущих.

fig = px.scatter(

    predected\_df,

    x = "Actual\_Salary",

    y = "Predicted\_Salary",

    color = "error",

    opacity=0.8,

    title= "Predicted Vs. Actual",

    template="plotly\_dark",

    trendline="ols"

)

fig.update\_layout(

    title = {

        "font" :{

            "size" : 28,

            "family" : "tahoma"

        }

    }

)

iplot(fig)



Рисунок 17. Предсказанные значения и актуальные

Далее обучим несколько других моделей и проверим насколько хорошо они показывают на нашем датасете.

Начнем с линейной регрессии:

from sklearn.linear\_model import LinearRegression

modelLR = LinearRegression()

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

y\_pr\_LR = modelLR.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelLR, X\_train, y\_train, X\_test, y\_test, y\_pr\_LR)

По результатам работы функции Reg\_Models\_Evaluation\_Metrics были получены следующие результаты метрик:

RMSE: 16546.3886

R2: 0.9105

Adjusted R2: 0.903

Cross Validated R2: 0.876

Несмотря на высокий R², более высокий RMSE указывает на то, что модель линейной регрессии делает более значительные ошибки предсказания по сравнению со случайным лесом. Это может быть связано с линейной природой модели, которая не всегда улавливает сложные зависимости в данных.

Отрисуем график линейной регрессии:

X2 = table['Salary'].values[:,np.newaxis]

y2 = table['Years of Experience'].values

model2 = LinearRegression()

model2.fit(X2, y2)

plt.scatter(X2, y2,color='g')

plt.plot(X2, model2.predict(X2),color='k')

plt.ylabel('Years of Experience', size=12)

plt.xlabel('Salary', size=12)

plt.show()

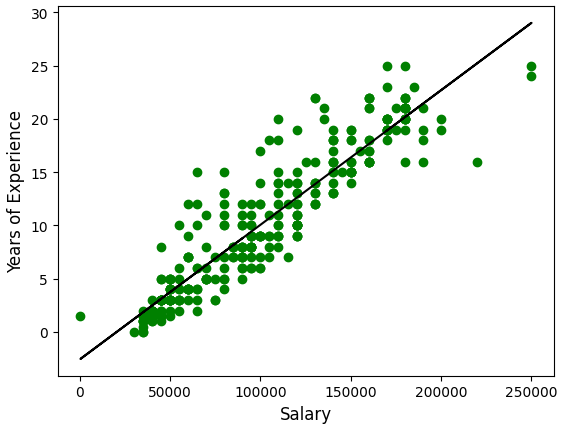


Рисунок 18. Предсказанные значения линейной регрессии

Теперь рассмотрим полиномиальную регрессию:

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=3, include\_bias=False)

poly\_features = poly.fit\_transform(X)

X\_p\_train, X\_p\_test, y\_p\_train, y\_p\_test = train\_test\_split(poly\_features, y, test\_size=0.3, random\_state=42)

modelPR = LinearRegression()

modelPR.fit(X\_p\_train, y\_p\_train)

y\_pr\_PR = modelPR.predict(X\_p\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelPR, X\_p\_train, y\_p\_train, X\_p\_test, y\_p\_test, y\_pr\_PR)

В результате получаем следующие метрики:

RMSE: 22222.908

R2: 0.7883

Adjusted R2: 0.5111

Cross Validated R2: 0.8777

Добавление полиномиальных признаков привело к значительному ухудшению RMSE и корректированного R², что свидетельствует о переобучении и избыточной сложности модели. Возможно, степень полинома была выбрана слишком высокой, что сделало модель менее обобщающей для новых данных. Сложные зависимости в данных лучше обрабатываются более гибкими моделями, такими как случайный лес.

Построим график для полиномиальных признаков:

X2 = table['Salary'].values[:,np.newaxis]

y2 = table['Years of Experience'].values

poly\_features = poly.fit\_transform(X2)

model2 = LinearRegression()

model2.fit(poly\_features, y2)

plt.scatter(X2, y2,color='g')

plt.plot(X2, model2.predict(poly.fit\_transform(X2)),color='k')

plt.ylabel('Years of Experience', size=12)

plt.xlabel('Salary', size=12)

plt.show()

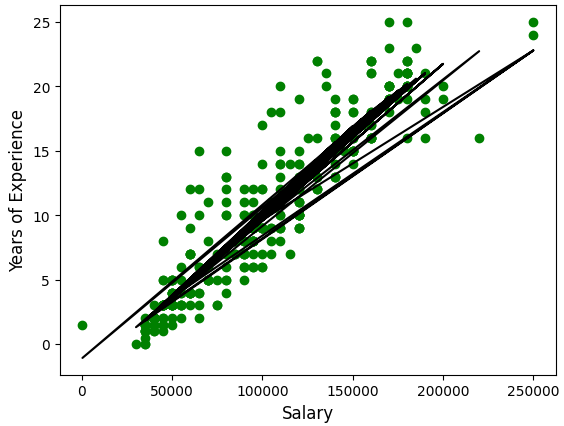


Рисунок 19. Полиномиальные признаки линейной регрессии

Рассмотрим регрессию дерева принятие решений:

from sklearn.tree import DecisionTreeRegressor

modelDT = DecisionTreeRegressor()

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

y\_dt\_pred = modelDT.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelDT, X\_train, y\_train, X\_test, y\_test, y\_dt\_pred)

Получаем следующие результаты метрик:

RMSE: 15787.3477

R2: 0.9186

Adjusted R2: 0.9117

Cross Validated R2: 0.7655

Модель хорошо объясняет вариацию данных на обучающей выборке, но его высокая точность на обучающей выборке и более низкий кросс-валидированный R² (0.7655) указывают на переобучение. Для улучшения можно попробовать ограничить глубину дерева, минимальный размер листьев или использовать ансамблевые методы, такие как случайный лес.

Построим регрессию случайного леса:

from sklearn.ensemble import RandomForestRegressor

modelRF = RandomForestRegressor(n\_estimators=100)

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

y\_rf\_pred = modelRF.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelRF, X\_train, y\_train, X\_test, y\_test, y\_rf\_pred)

Получаем следующие результаты метрик:

RMSE: 14774.6863

R2: 0.9287

Adjusted R2: 0.9226

Cross Validated R2: 0.8699

Регрессия случайного леса показала наилучшие результаты среди всех протестированных моделей. Низкий RMSE и высокие значения R² указывают на то, что модель успешно обрабатывает сложные зависимости в данных. Также можно попробовать оптимизацию гиперпараметров, чтобы ещё больше повысить точность и устойчивость модели.

Применим регрессию опорных векторов:

from sklearn.svm import SVR

modelSVR = SVR(kernel='linear')

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

y\_rf\_pred = modelSVR.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelSVR, X\_train, y\_train, X\_test, y\_test, y\_rf\_pred)

Получаем следующие метрики:

RMSE: 38747.4996

R2: 0.5094

Adjusted R2: 0.4678

Cross Validated R2: 0.4666

Высокий RMSE и низкие значения R² предполагают, что модель плохо справляется с предсказанием заработной платы. Это может быть связано с неправильным выбором ядра, гиперпараметров или отсутствием линейной зависимости между признаками и целевым признаком. Улучшение может потребовать настройки гиперпараметров или использования нелинейного ядра (например, RBF).

Построим график регрессии опорных векторов:

plt.scatter(X\_train["Years of Experience"], y\_train, color='darkorange',

            label='data')

plt.plot(X\_train["Years of Experience"], modelSVR.predict(X\_train), color='cornflowerblue',

         label='prediction')

plt.legend()

plt.show()

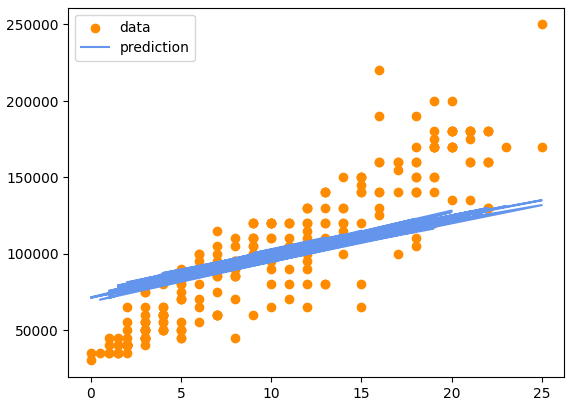


Рисунок 20. Метод опорных векторов с линейным ядром

Построим регрессию опорных векторов с использованием полиномиального ядра:

from sklearn.svm import SVR

modelSVR = SVR(kernel='poly')

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

y\_rf\_pred = modelSVR.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelSVR, X\_train, y\_train, X\_test, y\_test, y\_rf\_pred)

Получаем следующие результаты метрик:

RMSE: 55462.5935

R2: -0.0051

Adjusted R2: -0.0903

Cross Validated R2: -0.0424

SVR с полиномиальным ядром показал катастрофически низкие результаты. Возможно, степень полинома была слишком высокой, что привело к сильному переобучению на обучающей выборке или неадекватному описанию зависимости. Для улучшения стоит рассмотреть использование другого ядра или изменение гиперпараметров.

Построим график регрессии опорных векторов с полиномиальным ядром:

plt.scatter(X\_train["Years of Experience"], y\_train, color='darkorange',

            label='data')

plt.plot(X\_train["Years of Experience"], modelSVR.predict(X\_train), color='cornflowerblue',

         label='prediction')

plt.legend()

plt.show()

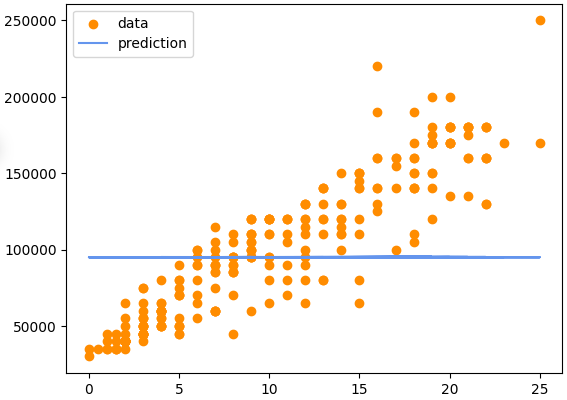


Рисунок 21. Метод опорных векторов с полиномиальным ядром

Рассмотрим регрессию опорных векторов с ядром, в качестве которого будет радиальная базисная функция:

modelSVR = SVR(kernel='rbf')

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

y\_rf\_pred = modelSVR.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelSVR, X\_train, y\_train, X\_test, y\_test, y\_rf\_pred)

Получаем следующие метрики:

RMSE: 55467.8078

R2: -0.0053

Adjusted R2: -0.0905

Cross Validated R2: -0.0427

Полученные метрики говорят о том, что модель содержит неправильно настроенные гиперпараметры, либо данные содержат слабую линейную зависимость.

Рассмотрим модель ElasticNet:

from sklearn.linear\_model import ElasticNet

modelEN = ElasticNet(alpha=0.1, l1\_ratio=0.8)

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

y\_en\_pred = modelEN.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelEN, X\_train, y\_train, X\_test, y\_test, y\_en\_pred)

Получаем следующие метрики:

RMSE: 16522.3274

R2: 0.9108

Adjusted R2: 0.9032

Cross Validated R2: 0.8757

ElasticNet успешно сбалансировал точность и способность к обобщению. Это объясняется использованием L1 и L2-регуляризации, которые предотвращают переобучение и устраняют незначимые признаки. Дальнейшее улучшение можно достичь путём подбора гиперпараметров alpha и l1\_ratio.

Построим график ElasticNet:

plt.scatter(X\_train["Years of Experience"], y\_train, color='darkorange',

            label='data')

plt.plot(X\_train["Years of Experience"], modelEN.predict(X\_train), color='cornflowerblue',

         label='prediction')

plt.legend()

plt.show()

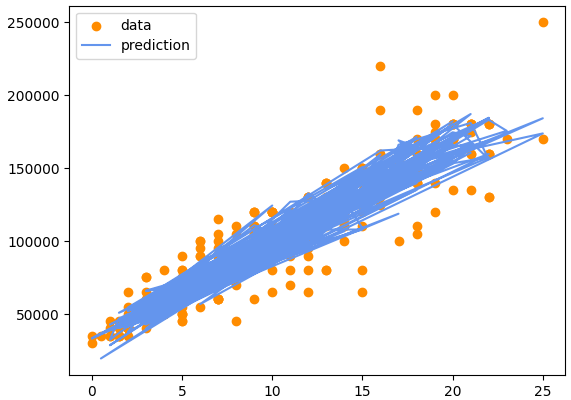


Рисунок 22. Предсказание ElasticNet

**Заключение**

В ходе работы были проанализированы данные о сотрудниках с использованием различных методов регрессии, включая линейную регрессию, ElasticNet, случайный лес, решающие деревья и метод опорных векторов с разными ядрами. Основная цель заключалась в построении модели, способной предсказывать уровень заработной платы на основе таких факторов, как возраст, пол, уровень образования, название должности и количество лет профессионального опыта.

Результаты показали, что наиболее значимыми признаками оказались возраст, количество лет опыта и уровень образования, что подтвердилось высокой корреляцией между этими показателями и зарплатой. Наилучшие результаты были получены с помощью моделей RandomForestRegressor и ElasticNet, обеспечивших минимальные значения RMSE и высокие показатели R², что указывает на хорошую способность моделей объяснять вариации данных и предсказывать зарплату.

Модели на основе метода опорных векторов (SVR) с различными ядрами показали худшие результаты. Это свидетельствует о том, что зависимости в данных плохо описываются сложными нелинейными функциями без дополнительной настройки гиперпараметров или масштабирования данных.

Таким образом, выдвинутая гипотеза подтвердилась: заработная плата сотрудников действительно зависит от возрастных и профессиональных характеристик, таких как уровень образования, профессиональный опыт и возраст. Применение методов регрессии позволило построить модели с высокой точностью, демонстрируя, что такие подходы могут быть эффективно использованы для прогнозирования заработной платы в различных профессиональных сферах.

**Список использованных источников и литературы**

1. Метиз Е. Изучаем Python: программирование игр, визуализация данных, веб-приложения. 3-е изд. — Москва: ДМК Пресс, 2022. — 480 с.
2. Любанович Б. Простой Python: современный стиль программирования. 2-е изд. — Москва: ДМК Пресс, 2022. — 384 с.
3. Волков С. Основы работы с библиотекой NumPy в Python [Электронный ресурс] / С. Волков // Habr. — 2018. — URL: <https://habr.com/ru/articles/353050/>
4. Маккини У. Python и анализ данных / пер. с анг. А. А. Слинкина. – М.: ДМК Пресс, 2020. – 540 с.
5. Саммерфилд М., Python на практике [Электронный ресурс] / Марк Саммерфилд - М. : ДМК Пресс, 2014. - 338 с. - ISBN 978-5-97060-095-5 - Режим доступа: http://www.studentlibrary.ru/book/ISBN9785970600955.html
6. Malik S. Data Analysis Using Python [Электронный ресурс] / S. Malik // Medium. — 2020. — URL: <https://medium.com/analytics-vidhya/data-analysis-using-python-24f1c1962a5>.
7. Как начать анализировать данные в Pandas: первые шаги [Электронный ресурс] // SkillFactory Blog. — URL: <https://blog.skillfactory.ru/kak-nachat-analizirovat-dannye-v-pandas-pervye-shagi/?ysclid=m4xu4bqbe9710425269>
8. 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].
9. 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].

**Приложения**

Приложение 1

Программный код

import pandas as pd

import numpy as np

import plotly.express as px

from plotly.offline import iplot

import seaborn as sns

import matplotlib.pyplot as plt

sns.set\_palette('husl')

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

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

from sklearn import model\_selection, preprocessing, feature\_selection, ensemble, linear\_model, metrics, decomposition

import warnings

warnings.filterwarnings("ignore")

pd.set\_option('future.no\_silent\_downcasting', True)

pd.options.mode.copy\_on\_write = "warn"

# сразу напишем код функции, которая будет вычислять все нужные метрики

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

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

from sklearn import metrics

from collections import Counter

def Reg\_Models\_Evaluation\_Metrics (model,X\_train,y\_train,X\_test,y\_test,y\_pred):

    cv\_score = cross\_val\_score(estimator = model, X = X\_train, y = y\_train, cv = 10)

    # Calculating Adjusted R-squared

    r2 = model.score(X\_test, y\_test)

    # Number of observations is the shape along axis 0

    n = X\_test.shape[0]

    # Number of features (predictors, p) is the shape along axis 1

    p = X\_test.shape[1]

    # Adjusted R-squared formula

    adjusted\_r2 = 1-(1-r2)\*(n-1)/(n-p-1)

    RMSE = np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred))

    R2 = model.score(X\_test, y\_test)

    CV\_R2 = cv\_score.mean()

    print(f'RMSE:  {round(RMSE,4)}\n  R2: {round(R2,4)}\n Adjusted R2: {round(adjusted\_r2, 4)}\n Cross Validated R2: {round(cv\_score.mean(),4)}')

    return R2, adjusted\_r2, CV\_R2, RMSE

from google.colab import drive

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

table = pd.read\_csv('/content/drive/MyDrive/Учеба/10 трим/Машинное обучение/итоговый проект/Salary Data.csv', encoding='utf-8', sep=',')

table.head()

# Adding Line To Plotly Figure

def add\_line(x0 = 0, y0 = 0, x1 = 0, y1 = 0,

             line\_color = "#00DFA2", font\_color = "#3C486B",

             xposition = "right", text = "Text"):

    fig.add\_shape(type='line',

                  x0 = x0,

                  y0 = y0,

                  x1 = x1,

                  y1 = y1 + 2,

                  line = {

                      "color" : line\_color,

                      "width" : 3,

                      "dash" : "dashdot"

                  },

                  label={

                      "text" : f"\t{text}: {x1: 0.1f}\t".expandtabs(5),

                      "textposition": "end",

                      "yanchor" :"top",

                      "xanchor" :xposition,

                      "textangle" :0,

                      "font": {

                          "size": 14,

                          "color" :font\_color,

                          "family" : "arial"

                      },

                  }

                 )

def custome\_layout(title\_size = 28, hover\_font\_size = 16, showlegend = False):

    fig.update\_layout(

    showlegend = showlegend,

    title = {

        "font" :{

            "size" :title\_size,

            "family" : "tahoma"

        }

    },

    hoverlabel = {

        "bgcolor" :"#111",

        "font\_size" : hover\_font\_size,

        "font\_family" :"arial"

    }

)

table.info()

table.describe().T

# Удаляем пустые значения

table.dropna(inplace=True)

# Проверяем на дубликаты

table.drop\_duplicates(inplace=True)

# Reset Our Index to be Orderd

table.reset\_index(inplace=True, drop=True)

mean\_of\_age = table["Age"].mean()

median\_of\_age = table["Age"].median()

fig = px.box(

    y=table["Age"],

    title= "Распределение по возрастам",

    template="plotly\_dark",

    labels={"y" :"Возраст"},

)

custome\_layout()

iplot(fig)

fig = px.histogram(

    table["Age"],

    nbins=25,

    title= "Распределение по возрастам",

    template="plotly\_dark",

    labels={"value" :"Возраст"}

)

custome\_layout()

fig.update\_traces(

    textfont = {

        "size" : 20,

        "family" :"tahoma",

        "color": "#fff"

    },

    hovertemplate = "Возраст: %{x}<br>Количество: %{y}",

    marker=dict(line=dict(color='#000', width=0.1))

)

# Adding Mean Line

add\_line(x0=mean\_of\_age, y0=0, x1=mean\_of\_age, y1=30+2, line\_color="#E97777",font\_color="#E97777",

         text="Mean", xposition="left")

# Adding Median Line

add\_line(x0=median\_of\_age, y0=0, x1=median\_of\_age, y1=30+2, line\_color="#FFE5F1",

         font\_color="#fff", xposition="right", text="Median")

iplot(fig)

gender = table["Gender"].value\_counts(normalize=1) \* 100

gender.apply(lambda x: f"{x:0.2f}%")

fig = px.bar(data\_frame = gender,

             x = gender.index,

             y = gender,

             color = gender.index,

             title = "Гендерное соотношение в процентах",

             color\_discrete\_sequence=["#45FFCA", "#FF9B9B"],

             labels= {"index" :"Gender", "y": "Процент(%)"},

             template="plotly\_dark",

             text = gender.apply(lambda x: f"{x:0.0f}%"))

custome\_layout()

fig.update\_traces(

    textfont = {

        "size" : 16,

        "family" :"arial",

        "color": "#222"

    },

    hovertemplate = "Пол: %{x}<br>Процент: %{y:0.1f}%",

)

iplot(fig)

education = table["Education Level"].value\_counts(normalize=1) \* 100

education.apply(lambda x: f"{x:0.2f}%")

fig = px.bar(data\_frame = education,

             x = education.index,

             y = education,

             color = education.index,

             title = "Соотношение образования",

             color\_discrete\_sequence=["#45FFCA", "#D09CFA", "#FF9B9B"],

             labels= {"index" :"Education", "y": "Процент по образованию"},

             template="plotly\_dark",

             text = education.apply(lambda x: f"{x:0.0f}%"))

custome\_layout()

fig.update\_traces(

    textfont = {

        "size" : 16,

        "family" :"arial",

        "color": "#222"

    },

    hovertemplate = "Образование: %{x}<br>Проценты: %{y:0.1f}%",

)

iplot(fig)

fig = px.box(

    y=table["Years of Experience"],

    title= "Распределение опыта, лет",

    template="plotly\_dark",

    labels={"y" :"Лет опыта"},

)

custome\_layout()

iplot(fig)

fig = px.box(

    x = table["Education Level"], y = table["Salary"],

    title= "Зарплата Vs. Уровень образования",

    template="plotly\_dark",

    labels={"x": "Уровень образования", "y" :"Зарплата"}

)

custome\_layout(hover\_font\_size=13)

iplot(fig)

# Средняя зарплата кадого пола

salary\_by\_gender = table.groupby("Gender")["Salary"].mean().sort\_values(ascending=False)

salary\_by\_gender.apply(lambda x: f"${x:,.2f}")

fig = px.bar(data\_frame = salary\_by\_gender,

             x = salary\_by\_gender.index,

             y = salary\_by\_gender,

             color = salary\_by\_gender.index,

             title = "Средняя зарплата по полам",

             color\_discrete\_sequence=["#45FFCA", "#D09CFA", "#FF9B9B"],

             labels= {"index" :"Education", "y": "Частота"},

             template="plotly\_dark",

             text\_auto = "0.4s"

            )

custome\_layout()

fig.update\_traces(

    textfont = {

        "size" : 16,

        "family" :"arial",

        "color": "#222"

    },

    hovertemplate = "Пол: %{x}<br>Частота: $%{y:0.4s}",

)

iplot(fig)

#Влияние уровня образования на зарплату

salary\_by\_education = table.groupby("Education Level")["Salary"].mean().sort\_values(ascending=False)

salary\_by\_education.apply(lambda x: f"${x:,.2f}")

fig = px.bar(data\_frame = salary\_by\_education,

             x = salary\_by\_education.index,

             y = salary\_by\_education,

             color = salary\_by\_education.index,

             title = "Зарплата в зависимоти от образования",

             color\_discrete\_sequence=["#45FFCA", "#D09CFA", "#FF9B9B"],

             labels= {"index" :"Образование", "y": "Процентное соотношение"},

             template="plotly\_dark",

             text\_auto = "0.4s"

            )

custome\_layout()

fig.update\_traces(

    textfont = {

        "size" : 16,

        "family" :"arial",

        "color": "#222"

    },

    hovertemplate = "Уровень образования: %{x}<br>Average Salary: $%{y:0.4s}",

)

iplot(fig)

def groupping\_exp(exp):

    if exp >= 0 and exp <= 5:

        return "0-5 years"

    elif exp > 5 and exp <= 10:

        return "6-10 years"

    elif exp > 10 and exp <= 15:

        return "11-15 years"

    elif exp > 15 and exp <= 20:

        return "16-20 years"

    else:

        return "20+"

salary\_by\_exp = table.groupby(table["Years of Experience"].apply(groupping\_exp))["Salary"].mean().sort\_values(ascending=False)

salary\_by\_exp.apply(lambda x: f"${x:,.2f}")

fig = px.bar(data\_frame = salary\_by\_exp,

             x = salary\_by\_exp.index,

             y = salary\_by\_exp,

             color = salary\_by\_exp.index,

             title = "Зарплата в зависимости от рабочего стажа",

             color\_discrete\_sequence=["#45FFCA", "#D09CFA", "#FF9B9B", "#F875AA", "#3EDBF0"],

             labels= {"index" :"Education", "y": "Зарплата"},

             template="plotly\_dark",

             text\_auto = "0.4s"

            )

custome\_layout()

fig.update\_traces(

    textfont = {

        "size" : 16,

        "family" :"arial",

        "color": "#222"

    },

    hovertemplate = "Рабочий стаж: %{x}<br>Зарплата $%{y:0.4s}",

)

iplot(fig)

#Перевод категориальных параметров в количественные

table['Gender'] = pd.factorize(table['Gender'])[0]

table['Education Level'] = pd.factorize(table['Education Level'])[0]

table['Job Title'] = pd.factorize(table['Job Title'])[0]

for index, feature\_name in enumerate(table.columns):

  plt.figure(figsize=(5, 4))

  plt.scatter(table[feature\_name],table['Salary'])

  plt.ylabel("Salary", size=12)

  plt.xlabel(feature\_name, size=12)

  plt.show()

sns.heatmap(table.corr())

# подготавливаем датасет к обучению

X = table.drop(['Salary'], axis=1)

y = table['Salary']

print(f'X shape: {X.shape} | y shape: {y.shape} ')

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=15)

from sklearn.linear\_model import LinearRegression

# Линейная регрессия

modelLR = LinearRegression()

# Fit the model to the data

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

# Predict the response for a new data point

y\_pr\_LR = modelLR.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelLR, X\_train, y\_train, X\_test, y\_test, y\_pr\_LR)

# попробуем отрисовать, как выглядит наша линейная регрессия

X2 = table['Salary'].values[:,np.newaxis]

# target data is array of shape (n,)

y2 = table['Years of Experience'].values

model2 = LinearRegression()

model2.fit(X2, y2)

plt.scatter(X2, y2,color='g')

plt.plot(X2, model2.predict(X2),color='k')

plt.ylabel('Years of Experience', size=12)

plt.xlabel('Salary', size=12)

plt.show()

#ПОЛИНОМИНАЛЬНАЯ РЕГРЕССИЯ

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=3, include\_bias=False)

poly\_features = poly.fit\_transform(X)

X\_p\_train, X\_p\_test, y\_p\_train, y\_p\_test = train\_test\_split(poly\_features, y, test\_size=0.3, random\_state=42)

# Create a linear regression model

modelPR = LinearRegression()

# Fit the model to the data

modelPR.fit(X\_p\_train, y\_p\_train)

# Predict the response for a new data point

y\_pr\_PR = modelPR.predict(X\_p\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelPR, X\_p\_train, y\_p\_train, X\_p\_test, y\_p\_test, y\_pr\_PR)

X2 = table['Salary'].values[:,np.newaxis]

# target data is array of shape (n,)

y2 = table['Years of Experience'].values

poly\_features = poly.fit\_transform(X2)

model2 = LinearRegression()

model2.fit(poly\_features, y2)

plt.scatter(X2, y2,color='g')

plt.plot(X2, model2.predict(poly.fit\_transform(X2)),color='k')

plt.ylabel('Years of Experience', size=12)

plt.xlabel('Salary', size=12)

plt.show()

#РЕГРЕССИЯ ДЕРЕВА ПРИНЯТИЯ РЕШЕНИЙ

from sklearn.tree import DecisionTreeRegressor

# Create a stepwise regression model

modelDT = DecisionTreeRegressor()

# Fit the model to the data

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

# Predict the response for a new data point

y\_dt\_pred = modelDT.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelDT, X\_train, y\_train, X\_test, y\_test, y\_dt\_pred)

#РЕГРЕССИЯ СЛУЧАЙНОГО ЛЕСА

from sklearn.ensemble import RandomForestRegressor

# Create a stepwise regression model

modelRF = RandomForestRegressor(n\_estimators=100)

# Fit the model to the data

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

# Predict the response for a new data point

y\_rf\_pred = modelRF.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelRF, X\_train, y\_train, X\_test, y\_test, y\_rf\_pred)

#РЕГРЕССИЯ ОПОРНЫХ ВЕКТОРОВ

from sklearn.svm import SVR

# Create a support vector regression model

modelSVR = SVR(kernel='linear')

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

# Predict the response for a new data point

y\_rf\_pred = modelSVR.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelSVR, X\_train, y\_train, X\_test, y\_test, y\_rf\_pred)

plt.scatter(X\_train["Years of Experience"], y\_train, color='darkorange',

            label='data')

plt.plot(X\_train["Years of Experience"], modelSVR.predict(X\_train), color='cornflowerblue',

         label='prediction')

plt.legend()

plt.show()

from sklearn.svm import SVR

# Create a support vector regression model. Полиномиальное ядро

modelSVR = SVR(kernel='poly')

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

# Predict the response for a new data point

y\_rf\_pred = modelSVR.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelSVR, X\_train, y\_train, X\_test, y\_test, y\_rf\_pred)

plt.scatter(X\_train["Years of Experience"], y\_train, color='darkorange',

            label='data')

plt.plot(X\_train["Years of Experience"], modelSVR.predict(X\_train), color='cornflowerblue',

         label='prediction')

plt.legend()

plt.show()

#Регрессия ElasticNet

from sklearn.linear\_model import ElasticNet

# Create an elastic net regression model

modelEN = ElasticNet(alpha=0.1, l1\_ratio=0.8)

# Fit the model to the data

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

# Predict the response for a new data point

y\_en\_pred = modelEN.predict(X\_test)

\_ = Reg\_Models\_Evaluation\_Metrics(modelEN, X\_train, y\_train, X\_test, y\_test, y\_en\_pred)

plt.scatter(X\_train["Years of Experience"], y\_train, color='darkorange',

            label='data')

plt.plot(X\_train["Years of Experience"], modelEN.predict(X\_train), color='cornflowerblue',

         label='prediction')

plt.legend()

plt.show()

kf = KFold(n\_splits=10, shuffle=True, random\_state=30)

rf = RandomForestRegressor(n\_estimators=500, random\_state=11)

scores = cross\_val\_score(rf, X, y, cv=kf)

print(f"Cross Validation Score: {np.mean(scores)\*100:0.2f}%")

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

score = rf.score(X\_train, y\_train)\*100

print(f"Model Score: {np.round(score, 2)}%")

predicted\_salary = np.round(rf.predict(X\_test))

d = {

    "Actual\_Salary" : y\_test,

    "Predicted\_Salary" : predicted\_salary,

    "error": predicted\_salary - y\_test

}

predected\_df = pd.DataFrame(d)

predected\_df.head()

score = r2\_score(y\_test, predicted\_salary)\*100

print(f"Model Score: {np.round(score, 2)}%")

rmse = np.sqrt(mean\_squared\_error(y\_test, predicted\_salary))

print(f"Error Ratio: {rmse:.3f}")

fig = px.scatter(

    predected\_df,

    x = "Actual\_Salary",

    y = "Predicted\_Salary",

    color = "error",

    opacity=0.8,

    title= "Predicted Vs. Actual",

    template="plotly\_dark",

    trendline="ols"

)

fig.update\_layout(

    title = {

        "font" :{

            "size" : 28,

            "family" : "tahoma"

        }

    }

)

iplot(fig)