Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Цифровая кафедра

ИТОГОВЫЙ ПРОЕКТ

по программе дополнительного профессионального образования –

программе профессиональной переподготовки

«Разработка прикладного программного обеспечения»

на тему   
«Анализ данных»

Выполнил(и) \_\_\_\_\_\_\_\_\_\_ /И.О. Фамилия/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Киров 2025

**Содержание**

[1. Постановка задачи анализа данных 3](#_Toc142982460)

[2. Подготовка набора данных 3](#_Toc142982461)

[3. Предварительная обработка данных 4](#_Toc142982462)

[4. Исследовательский анализ данных 8](#_Toc142982463)

[5. Построение модели линейной регрессии и её исследование 12](#_Toc142982464)

[6. Использование метода k-ближайших соседей 15](#_Toc142982465)

[7. Выводы по полученным результатам анализа данных 17](#_Toc142982466)

[Список литературы 18](#_Toc142982467)

[Приложение 19](#_Toc142982468)

# **1. Постановка задачи анализа данных**

На основе имеющихся данных об уровне заработной платы в области Data Science построить модель регрессии и выполнить прогнозирование размера ожидаемой оплаты труда для различных должностей и опыта работы в данной сфере.

# **2. Подготовка набора данных**

Для анализа возьмём набор данных с названием «Data Science Salaries 2023» с ресурса <https://www.kaggle.com/datasets/arnabchaki/data-science-salaries-2023>. Данные размещены в таблице, содержащей 11 столбцов:

1. work\_year – год, когда была выплачена зарплата,
2. experience\_level – уровень опыта работы в течение года,
3. employment\_type – тип найма (полная ставка, совмещение и т.п.),
4. job\_title – должность,
5. salary – общая сумма выплаченной брутто-зарплаты,
6. salary\_currency – валюта выплачиваемой зарплаты в виде кода валюты ISO 4217,
7. salary\_in\_usd – заработная плата в долларах США,
8. employee\_residence – основная страна проживания сотрудника в течение рабочего года в виде кода страны ISO 3166,
9. remote\_ratio – общий объём удаленной работы,
10. company\_location – страна главного офиса или филиала работодателя,
11. company\_size – среднее количество сотрудников, которые работали в компании в течение года.

Структурированные данные ещё называют «панельными» или panel data, для работы с ними средствами языка Python используется библиотека pandas.

Для хранения данных в pandas используют специальную табличную структуру данных DataFrame. Каждый элемент такой структуры является строкой таблицы и имеет порядковый номер.

Для анализа загрузим данные в DataFrame с именем df и выведем первые несколько строк с помощью следующего программного кода:

df = pd.read\_csv('/kaggle/input/data-science-salaries-2023/ds\_salaries.csv', sep=',')

df.head()

Результат выполнения кода представлен на рисунке 1.



Рисунок 1 – Результат выполнение кода

Используя полученную структуру данных, перейдём   
к непосредственному их анализу.

# **3. Предварительная обработка данных**

Определим, какой тип имеют столбцы датафрейма df. Соответствующий программный код и результат его выполнения представлены на рисунке 2.

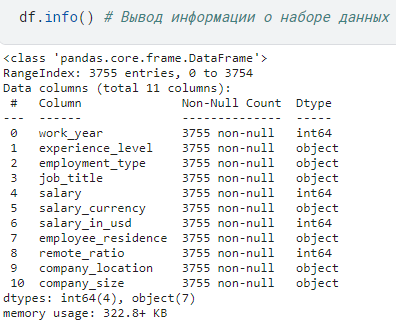


Рисунок 2 – Результат выполнения кода

Можно видеть, что анализируемый набор данных включает в себя 11 столбцов и 3755 строк. Количество непустых (Non-Null) значений для всех столбцов равно 3755, что означает высокое качество имеющихся данных, а значит, не потребуется их коррекция.

В последнем столбце Dtype приводятся типы данных хранимых значений: целочисленными являются work\_year, salary, salary\_in\_usd, remote\_ratio, все остальные столбцы хранят данные типа object.

Нужно определить, как поступить с данными столбцов, имеющими тип object. Возможны два варианта действий: исключить эти столбцы или преобразовать их к числовому типу, более пригодному для автоматической обработки.

Для понимания смысла данных, содержащихся в столбцах, в блокноте в текстовой ячейке приведём описание набора.

Для корректного сравнения заработной платы все значения должны быть в единой валюте. В наборе данных имеется соответствующий столбец salary\_in\_usd, поэтому столбцы salary и salary\_currency, отвечающие за размер и указание национальной валюты, логически дублируют смысл нашей целевой переменной, поэтому нам не потребуются. Удалим их из датафрейма:

df = df.drop("salary", axis=1)

df = df.drop("salary\_currency", axis=1)

df.head()

В результате мы немного сократили размерность данных (рисунок 3).

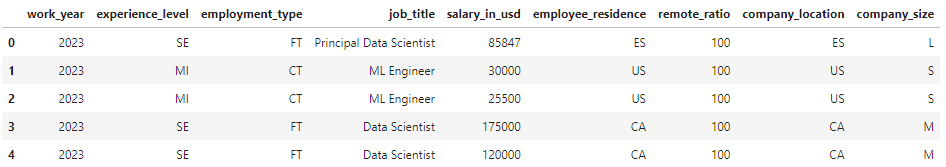


Рисунок 3 – Сокращение размерности

Сформируем списки уникальных значений и подсчитаем их количество для столбцов experience\_level, employment\_type, job\_title, employee\_residence и company\_location, company\_size.

Соответствующий код для столбца experience\_level имеет вид:

# Опыт работы

experience = df.experience\_level.unique().tolist()

print(experience)

len(experience)

Для остальных столбцов код аналогичен. Получили следующие результаты (таблица 1).

Таблица 1 – Описание столбцов таблицы

|  |  |  |
| --- | --- | --- |
| **Столбец** | **Смысл** | **Количество значений** |
| work\_year | год (2023, 2022, 2020, 2021) | 4 |
| experience\_level | уровень опыта работы | 4 |
| employment\_type | тип найма | 4 |
| job\_title | должность | 93 |
| employee\_residence | страна проживания сотрудника | 78 |
| company\_location | страна главного офиса или филиала работодателя | 72 |
| company\_size | размер компании | 3 |

Уникальные значения в столбцах job\_title, employee\_residence и company\_location при выводе были отсортированы, чтобы убедиться, что не было случаев, когда одни и те же значения записаны различным способом.

Список уникальных значений в столбце job\_title:

['3D Computer Vision Researcher', 'AI Developer', 'AI Programmer', 'AI Scientist', 'Analytics Engineer', 'Applied Data Scientist', 'Applied Machine Learning Engineer', 'Applied Machine Learning Scientist', 'Applied Scientist', 'Autonomous Vehicle Technician', 'Azure Data Engineer', 'BI Analyst', 'BI Data Analyst', 'BI Data Engineer', 'BI Developer', 'Big Data Architect', 'Big Data Engineer', 'Business Data Analyst', 'Business Intelligence Engineer', 'Cloud Data Architect', 'Cloud Data Engineer', 'Cloud Database Engineer', 'Compliance Data Analyst', 'Computer Vision Engineer', 'Computer Vision Software Engineer', 'Data Analyst', 'Data Analytics Consultant', 'Data Analytics Engineer', 'Data Analytics Lead', 'Data Analytics Manager', 'Data Analytics Specialist', 'Data Architect', 'Data DevOps Engineer', 'Data Engineer', 'Data Infrastructure Engineer', 'Data Lead', 'Data Management Specialist', 'Data Manager', 'Data Modeler', 'Data Operations Analyst', 'Data Operations Engineer', 'Data Quality Analyst', 'Data Science Consultant', 'Data Science Engineer', 'Data Science Lead', 'Data Science Manager', 'Data Science Tech Lead', 'Data Scientist', 'Data Scientist Lead', 'Data Specialist', 'Data Strategist', 'Deep Learning Engineer', 'Deep Learning Researcher', 'Director of Data Science', 'ETL Developer', 'ETL Engineer', 'Finance Data Analyst', 'Financial Data Analyst', 'Head of Data', 'Head of Data Science', 'Head of Machine Learning', 'Insight Analyst', 'Lead Data Analyst', 'Lead Data Engineer', 'Lead Data Scientist', 'Lead Machine Learning Engineer', 'ML Engineer', 'MLOps Engineer', 'Machine Learning Developer', 'Machine Learning Engineer', 'Machine Learning Infrastructure Engineer', 'Machine Learning Manager', 'Machine Learning Research Engineer', 'Machine Learning Researcher', 'Machine Learning Scientist', 'Machine Learning Software Engineer', 'Manager Data Management', 'Marketing Data Analyst', 'Marketing Data Engineer', 'NLP Engineer', 'Power BI Developer', 'Principal Data Analyst', 'Principal Data Architect', 'Principal Data Engineer', 'Principal Data Scientist', 'Principal Machine Learning Engineer', 'Product Data Analyst', 'Product Data Scientist', 'Research Engineer', 'Research Scientist', 'Software Data Engineer', 'Staff Data Analyst', 'Staff Data Scientist']

Список уникальных значений в столбце employee\_residence:

['AE', 'AM', 'AR', 'AS', 'AT', 'AU', 'BA', 'BE', 'BG', 'BO', 'BR', 'CA', 'CF', 'CH', 'CL', 'CN', 'CO', 'CR', 'CY', 'CZ', 'DE', 'DK', 'DO', 'DZ', 'EE', 'EG', 'ES', 'FI', 'FR', 'GB', 'GH', 'GR', 'HK', 'HN', 'HR', 'HU', 'ID', 'IE', 'IL', 'IN', 'IQ', 'IR', 'IT', 'JE', 'JP', 'KE', 'KW', 'LT', 'LU', 'LV', 'MA', 'MD', 'MK', 'MT', 'MX', 'MY', 'NG', 'NL', 'NZ', 'PH', 'PK', 'PL', 'PR', 'PT', 'RO', 'RS', 'RU', 'SE', 'SG', 'SI', 'SK', 'TH', 'TN', 'TR', 'UA', 'US', 'UZ', 'VN']

Список кодов стран в столбце company\_location представлен в Приложении.

В столбцах experience\_level, employment\_type и company\_size разнообразие значений невелико, поэтому можно перейти от типа object к int64 с помощью вспомогательных словарей:

experience\_dict = {'SE': 1, 'MI': 2, 'EN': 3, 'EX': 4}

employment\_dict = {'FT': 1, 'CT': 2, 'FL': 3, 'PT': 4}

csize\_dict = {'S': 1, 'M': 2, 'L': 3}

Замена значений в столбцах выполняется с помощью следующего программного кода:

df.experience\_level = df.experience\_level.apply(lambda x : experience\_dict[x])

df.employment\_type = df.employment\_type.apply(lambda x : employment\_dict[x])

df.company\_size = df.company\_size.apply(lambda x : csize\_dict[x])

В результате датафрейм принял следующий вид (рисунок 4).

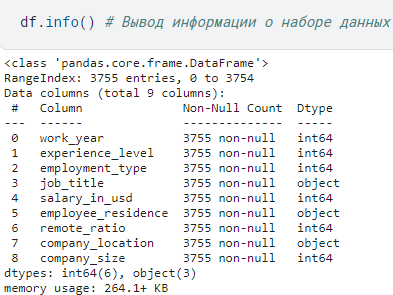


Рисунок 4 – Датафрейм

После выполненных преобразований можно переходить к следующему этапу решения поставленной задачи – исследовательскому анализу данных.

# **4. Исследовательский анализ данных**

Исследовательский анализ данных начнём с вычисления матрицы корреляции (рис. 5). Данная матрица состоит из коэффициентов корреляции Пирсона, т. е. содержит безразмерные индексы в диапазоне от –1,0 до 1,0 включительно, которые отражают степень линейной зависимости между двумя множествами данных.

corr\_matrix = df.corr(numeric\_only=True).round(2)

corr\_matrix

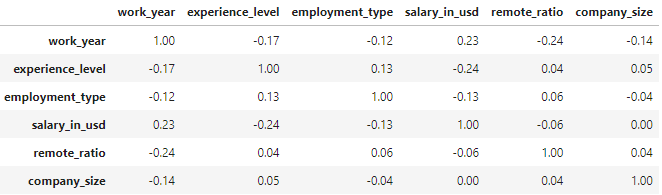


Рисунок 5 – Корреляционная матрица

Визуализируем корреляционную матрицу с помощью библиотеки seaborn, которая позволяет строить тепловую карту. Результат построения тепловой карты представлен на рисунке 6, программный код приведён в Приложении.

{{IMAGE1}}

Рисунок 6 – Тепловая карта

На главной диагонали отображается самый насыщенный цвет – он означает, что параметр сам с собой коррелирует с максимальным коэффициентом 1. Рядом с картой изображена шкала-подсказка.

По тепловой карте видно, что все коэффициенты корреляции имеют достаточно малые значения. Обратим внимание на коэффициент зависимости зарплаты от размера компании – он равен 0, а это значит, что при построении модели анализа данных столбец company\_size можно не учитывать.

Интересно оценить зависимость уровня заработной платы от должностей.

Список должностей jobs мы уже получили, теперь подсчитаем среднюю заработную плату для каждой должности и сохраним в массив с именем mean\_salaries. Изначально массив заполнен нулями. Код вычисления средней зарплаты представлен ниже:

n = len(jobs)

mean\_salaries = np.zeros(n)

for i in range(0, n):

position = jobs[i] #текущая должность

df\_tmp = df[df['job\_title'] == position] # выбираем соотвествующие записи

mean\_salaries[i] = df\_tmp['salary\_in\_usd'].mean()

Результат визуализации зависимости уровня средней зарплаты от должности job\_title в виде гистограммы представлен на рисунке 7. Соответствующий код на языке Python представлен в Приложении.

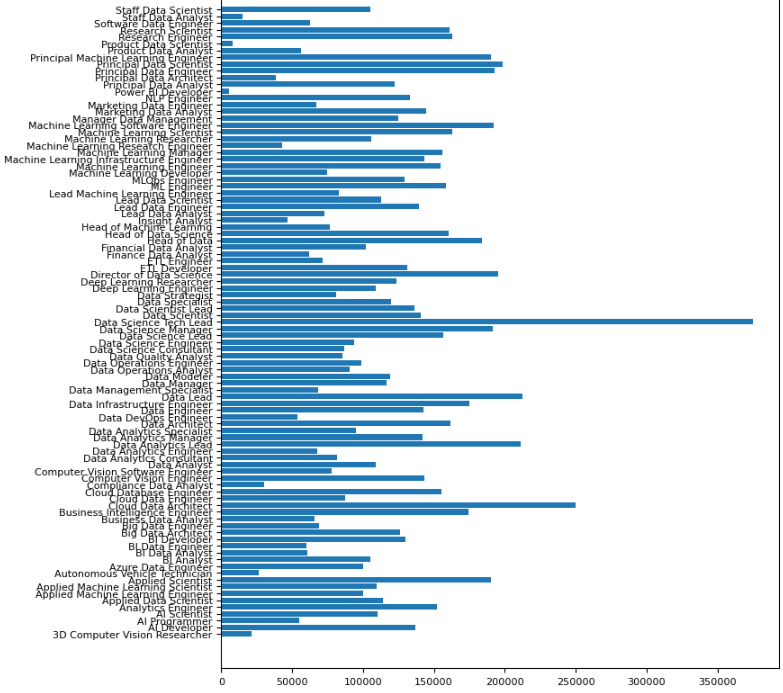


Рисунок 7 – Гистограмма

Найдём программно должности, на которых среднее значение зарплаты было максимальным и минимальным. В результате получили, что максимальный уровень зарплат наблюдается на должности Data Science Tech Lead, а минимальный – на должности Power BI Developer (рисунок 8).

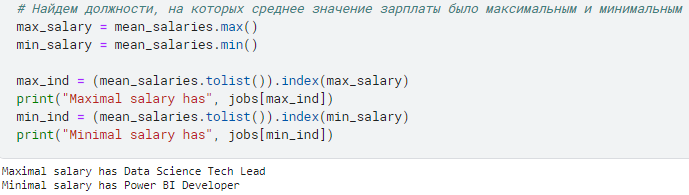


Рисунок 8 – Программный код и результат его выполнения

С помощью гистограммы можно определить, что относительно низкий уровень средней зарплаты имеют также Product Data Scientist и Staff Data Analyst.

Как уже обозначено ранее, на уровень заработной платы совершенно не влияет размер компании company\_size (соответствующий коэффициент корреляции Пирсона равен 0), а также можно пренебречь влиянием доли удалённой работы remote\_ratio.

Таким образом, в целевую переменную y сохраним значения заработной платы, а в датафрейм X запишем признаки с наиболее высокой корреляцией с целевой переменной:

X = df[['work\_year', 'experience\_level', 'employment\_type']]

y = df['salary\_in\_usd']

Перейдём к построению моделей для прогнозирования уровня заработной платы.

# **5. Построение модели линейной регрессии и её исследование**

**5.1. Инициализация модели**

Начнём с того, что разобьём имеющийся набор данных на обучающую и тестовую выборки с помощью стандартных функций библиотеки. Размер тестовой выборки установим в {{PROCENT}}%, также зададим точку отсчёта random\_state = {{RANDOM\_STATE}} для воспроизводимости результатов. Для контроля выведем размеры полученных выборок:

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,

test\_size = 0.{{PROCENT}},

random\_state = {{RANDOM\_STATE}})

# размерность обучающей выборки

print(X\_train.shape, y\_train.shape)

# размерность тестовой выборки

print(X\_test.shape, y\_test.shape)

Получили, что в обучающую выборку вошло {{LEANING}} значения, а в тестовую – {{TEST}}.

Модель линейной регрессии представляет собой линейную комбинацию векторов признаков и весов. Будем использовать готовую реализацию линейной регрессии из библиотеки sklearn.

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

**5.2. Обучение и тестирование модели**

После того, как объект модели создан, необходимо провести обучение, которое заключается в подборе весов для признаков. Используем для обучения стандартную функцию.:

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

Мы получили обученную модель (рисунок 9).

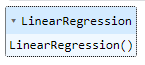


Рисунок 9 – Обученная модель в среде Kaggle

Когда алгоритм обучен, ещё один важный шаг – понять, насколько качественно он работает, иными словами, оценить качество выполняемых алгоритмом предсказаний. В качестве метрик рассмотрим корень среднеквадратической ошибки (Root Mean Squared Error (RMSE)) и R2. Значение RMSE показывает, насколько хорошо модель регрессии может предсказать значение целевой переменной в абсолютном выражении, в то время как R2 отражает, насколько хорошо модель может предсказать значение переменной отклика в процентном выражении. Соответственно, чем ближе значение R2 к 1, тем выше качество предсказаний.

В массиве y\_test хранятся значения целевой переменной из тестовой выборки, а в y\_pred запишем предсказанные значения, полученные с помощью модели линейной регрессии:

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

Импортируем модуль метрик из библиотеки sklearn:

from sklearn import metrics

Выведем значения корня среднеквадратической ошибки и R2:

print('Root Mean Squared Error (RMSE):', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

print('R2:', np.round(metrics.r2\_score(y\_test, y\_pred), 2))

Получили следующие значения:

{{ROOT\_MEAN1}}

{{R1}}

Визуализируем тестовые значения и прогнозируемые с помощью модели линейной регрессии (рисунок 10). Соответствующий программный код представлен в приложении.

{{IMAGE2}}

Рисунок 10 - Визуализация

Как метрики качества, так и визуализация предсказаний показали их невысокую точность.

Следует построить другую модель, чтобы попробовать улучшить качество прогноза, например, метод k-ближайших соседей.

# **6. Использование метода k-ближайших соседей**

Метод k-ближайших соседей (kNN – k Nearest Neighbours) – метод решения задач классификации и задач регрессии, основанный на поиске ближайших объектов с известными значения целевой переменной.

Предсказания модели выглядят следующим образом:



где x – новый объект, для которого требуется построить прогноз, K – число ближайших соседей, которых нужно рассматривать.

Из библиотеки sklearn импортируем регрессию по методу   
k-ближайших соседей, создадим и обучим модель (рисунок 11).

from sklearn.neighbors import KNeighborsRegressor

model2 = KNeighborsRegressor()

model2.fit(X\_train, y\_train)



Рисунок 11 – Обученная модель k-ближайших соседей

Найдём прогнозируемые значения и вычислим метрики RMSE и R2:

pred\_knn = model2.predict(X\_test)

print('Root Mean Squared Error (RMSE):', np.round(np.sqrt(metrics.mean\_squared\_error(y\_test, pred\_knn)), 2))

print('R2:', np.round(metrics.r2\_score(y\_test, pred\_knn), 2))

Получены следующие значения:

{{ROOT\_MEAN2}}

{{R2}}

Визуализируем тестовые значения и прогнозируемые с помощью модели k-ближайших соседей, аналогично тому, как это сделано ранее (рисунок 12). Код на Python также представлен в Приложении.

{{IMAGE3}}

Рисунок 12 - Визуализация

На рисунке 12 прогнозируемые значения несколько лучше, чем на рисунке 10, однако качество предсказаний всё ещё низкое.

# **7. Выводы по полученным результатам анализа данных**

Для прогнозирования размера ожидаемой оплаты труда в области Data Science был использован набор данных «Data Science Salaries 2023» с ресурса Kaggle.

Был проведён исследовательский анализ данных, который позволил определить, что максимальный уровень зарплат наблюдается на должности Data Science Tech Lead, а минимальный – на должностях Power BI Developer, Product Data Scientist и Staff Data Analyst.

Прогнозирование проводилось с использованием двух моделей – линейной регрессии и k-ближайших соседей.

Для обеих моделей метрики оценки качества прогнозирования имеют неудовлетворительные значения, что свидетельствует о том, что для более точно прогнозирования размера ожидаемой оплаты труда требуется анализ иных параметров, чем те, что представлены в наборе данных «Data Science Salaries 2023».

# **Список литературы**

1. Берман К. Основы Python для Data Science. – СПб.: Питер, 2023. – 272 с.
2. Брюс П., Брюс Э. Разведочный анализ данных // Практическая статистика для специалистов Data Science. – СПб.: БХВ-Петербург, 2018. – 304 с.
3. Дюк В., Самойленко А. Data Mining: учебный курс (+CD). – СПб.: Изд. Питер, 2001. – 368 с.
4. Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными = Introduction to Machine Learning with Python: A Guide for Data Scientists. – Вильямс, 2017. – 480 с.
5. Sandro Tosi Matplotlib for Python Developers. – Packt Publishing, 2009. – 308 с.

# **Приложение**

Листинг программы на Python

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

for filename in filenames:

print(os.path.join(dirname, filename))

df = pd.read\_csv('/kaggle/input/data-science-salaries-2023/ds\_salaries.csv', sep=',')

df.head()

df.info()

df = df.drop("salary", axis=1)

df = df.drop("salary\_currency", axis=1)

df.head()

year = df.work\_year.unique().tolist()

print(year)

len(year)

experience = df.experience\_level.unique().tolist()

print(experience)

len(experience)

employment = df["employment\_type"].unique().tolist()

print(employment)

len(employment)

jobs = df.job\_title.unique().tolist()

jobs.sort()

print(jobs)

len(jobs)

residence = df["employee\_residence"].unique().tolist()

residence.sort()

print(residence)

len(residence)

location = df["company\_location"].unique().tolist()

location.sort()

print(location)

len(location)

csize = df["company\_size"].unique().tolist()

print(csize)

len(csize)

experience\_dict = {'SE': 1, 'MI': 2, 'EN': 3, 'EX': 4}

employment\_dict = {'FT': 1, 'CT': 2, 'FL': 3, 'PT': 4}

csize\_dict = {'S': 1, 'M': 2, 'L': 3}

df.experience\_level = df.experience\_level.apply(lambda x : experience\_dict[x])

df.employment\_type = df.employment\_type.apply(lambda x : employment\_dict[x])

df.company\_size = df.company\_size.apply(lambda x : csize\_dict[x])

df.info()

corr\_matrix = df.corr(numeric\_only=True).round(2)

import seaborn as sns

fig, ax = plt.subplots(figsize=(9, 6))

sns.heatmap(corr\_matrix, cmap="{{COLOR}}", annot=True)

plt.show()

n = len(jobs)

mean\_salaries = np.zeros(n)

for i in range(0,n):

position = jobs[i]

df\_tmp = df[df['job\_title'] == position] mean\_salaries[i] = df\_tmp['salary\_in\_usd'].mean()

fig = plt.figure(figsize=(8, 10))

plt.rc('font', size = 8 )

plt.barh(jobs, mean\_salaries)

plt.show()

max\_salary = mean\_salaries.max()

min\_salary = mean\_salaries.min()

max\_ind = (mean\_salaries.tolist()).index(max\_salary)

print("Maximal salary has", jobs[max\_ind])

min\_ind = (mean\_salaries.tolist()).index(min\_salary)

print("Minimal salary has", jobs[min\_ind])

X = df[['work\_year', 'experience\_level', 'employment\_type']]

y = df['salary\_in\_usd']

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,

test\_size = 0.{{PROCENT}},

random\_state = {{RANDOM\_STATE}})

print(X\_train.shape, y\_train.shape)

print(X\_test.shape, y\_test.shape)

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

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

y\_pred = model.predict(X\_test) print(np.round(y\_pred[:5],2))

from sklearn import metrics

print('Root Mean Squared Error (RMSE):', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

print('R2:', np.round(metrics.r2\_score(y\_test, y\_pred), 2))

order = np.argsort(y\_test.values) y\_test\_ordered = y\_test.values[order] y\_pred\_ordered = y\_pred[order]

plt.figure(figsize=(10, 8))

plt.scatter(y\_test\_ordered, y\_pred\_ordered, label="Linear Regression") plt.plot(y\_test\_ordered, y\_test\_ordered, label="True values", color="red")

plt.legend()

plt.xlabel("True values")

plt.ylabel("Predicted values")

from sklearn.neighbors import KNeighborsRegressor

model2 = KNeighborsRegressor()

model2.fit(X\_train, y\_train)

pred\_knn = model2.predict(X\_test)

print('Root Mean Squared Error (RMSE):', np.round(np.sqrt(metrics.mean\_squared\_error(y\_test, pred\_knn)), 2))

print('R2:', np.round(metrics.r2\_score(y\_test, pred\_knn), 2))

plt.figure(figsize=(10, 8))

plt.scatter(y\_test\_ordered, pred\_knn[order], label="kNN")

plt.plot(y\_test\_ordered, y\_test\_ordered, label="True values", color="red") plt.legend();

plt.xlabel("True values");

plt.ylabel("Predicted values");