ГУАП

КАФЕДРА № 41

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

ст. преподаватель		В.В. Боженко
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

РЕГРЕССИОННЫЙ АНАЛИЗ

по курсу: ВВЕДЕНИЕ В АНАЛИЗ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4316	09.11.2025	Э.А.Чылдырлы
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

Цель лабораторной работы: изучение алгоритмов и методов регрессии

Часть 1. Простая линейная регрессия

В первую очередь было загружено и проанализировано содержимое файла для данной работы. Данные, взятые из файла (рис. 1), представляют собой набор числовых значений, где x1 и x2 — это признаки (независимые переменные), а y — целевая переменная (зависимая).

	x1	x2	y
0	-1.655715	1.204577	2.777909
1	-0.461736	1.494205	0.594868
2	0.823418	-1.024070	-1.414032
3	0.900855	0.440111	0.104884
4	0.511414	-1.753043	-1.387142
5	1.516009	1.133519	1.076325
6	0.454717	1.515118	1.043466
7	-1.298969	-0.128912	0.297386
8	-0.502645	-0.267853	-0.364504
9	0.893185	-0.275706	0.901817

Рисунок 1 — Фрагмент данных датасета

Для дальнейшего анализа и построения моделей данные были разделены на обучающую и валидационную выборки в соотношении 75% / 25% с фиксированным random_state=0 для воспроизводимости результатов. Это позволило обучить две отдельные линейные регрессионные модели: одну на признаке x1, другую — на признаке x2.

Обучение моделей выполнено с помощью класса LinearRegression из библиотеки sklearn.linear_model (рис. 2).

```
# Разделение на признаки и целевую переменную
X1 = df[['x1']]
X2 = df[['x2']]
y = df['y']

# Разделение на обучающую и валидационную выборки (75% / 25%, random_state=0)
X1_train, X1_valid, y_train, y_valid = train_test_split(X1, y, test_size=0.25, random_state=0)
X2_train, X2_valid, _, _ = train_test_split(X2, y, test_size=0.25, random_state=0)

# Обучение моделей
model1 = LinearRegression()
model1.fit(X1_train, y_train)

model2 = LinearRegression()
model2.fit(X2_train, y_train)
print("Модели обучены")

df.head(10)
```

```
Модели обучены
```

Рисунок 2 — Фрагмент кода обучения моделей

Обе модели обучались на обучающей выборке, а затем применялись для предсказания значений целевой переменной y на валидационной выборке. Результатом стало получение двух наборов предсказаний: y_pred_x1 и y_pred_x2 (рис. 3).

Для проверки корректности работы моделей были выведены первые пять предсказанных значений для каждого признака. Видно, что значения различаются: модель на x1 даёт предсказания в диапазоне от -0.03 до 0.77, а модель на x2 — от -0.53 до 1.26. Это указывает на то, что признаки x1 и x2 оказывают разное влияние на целевую переменную, и их вклад в модель неодинаков.

Это различие свидетельствует не только о разной шкале влияния признаков, но и о различной силе и форме их связи с целевой переменной. Более узкий диапазон предсказаний у модели x1 указывает на то, что этот признак обеспечивает более стабильные и сбалансированные прогнозы, что может быть связано с более выраженной и систематической зависимостью между x1 и y.

Напротив, больший разброс предсказаний у модели x2 говорит о том, что её поведение менее стабильно и, вероятно, менее надёжно. Это согласуется с тем, что связь между x2 и y слабее и ближе к линейной. Таким образом, модель, основанная на x2, демонстрирует меньшую способность к точному восстановлению целевой переменной.

```
# Предсказание
y_pred_x1 = model1.predict(X1_valid)
y_pred_x2 = model2.predict(X2_valid)

# Вывод первых 5 предсказаний для проверки
print(f"x1: {y_pred_x1[:5]}")
print(f"x2: {y_pred_x2[:5]}")
```

```
x1: [ 0.02016639 -0.03298637  0.77344975  0.24677118  0.356892  ]
x2: [ 1.26742422 -0.17078799  0.10605387 -0.53293957  1.13332421]
```

Рисунок 3 — Полученные предсказания

Далее были созданы два отдельных датафрейма: df_results_x1 и df_results_x2 (рис. 4). Каждый из них содержит две колонки:

- Actual — истинные (реальные) значения целевой переменной y из валидационной выборки.
- Predicted — значения, предсказанные соответствующей моделью (model1 для x1, model2 для x2).

Анализ показывает, что предсказания двух моделей существенно различаются. Например, для первого наблюдения модель на x1 предсказывает 0.020166, а модель на x2 — 1.267424. Такие расхождения наблюдаются и для других наблюдений, что указывает на

то, что признаки x1 и x2 оказывают разное влияние на целевую переменную и формируют разные прогнозы.

```
Результаты модели y ~ x1:
       Actual  Predicted
0   -0.395571   0.020166
1    0.893850  -0.032986
2    0.633611   0.773450
3   -1.155219   0.246771
4    1.176699   0.356892
..        ...        ...
245  0.784664   0.384325
246  0.993012  -0.167211
247 -1.414032   0.123932
248  1.670966   0.966218
249  1.839032   0.878368

[250 rows x 2 columns]

================================================

Результаты модели y ~ x2:
       Actual  Predicted
0   -0.395571   1.267424
1    0.893850  -0.170788
2    0.633611   0.106054
3   -1.155219  -0.532940
4    1.176699   1.133324
..        ...        ...
245  0.784664   1.120814
246  0.993012   0.999788
247 -1.414032  -0.183275
248  1.670966  -0.517545
249  1.839032   0.248457
```

Рисунок 4 — Полученные датафреймы

Следующим шагом стало вычисление коэффициентов линейной регрессии (рис. 5).

```python
# Вывод коэффициентов моделей (a и b в y = a*x + b)
print("Модель y ~ x1:")
print(f"  a (коэффициент при x1): {model1.coef_[0]:.4f}")
print(f"  b (свободный член): {model1.intercept_:.4f}")

print("\nМодель y ~ x2:")
print(f"  a (коэффициент при x2): {model2.coef_[0]:.4f}")
print(f"  b (свободный член): {model2.intercept_:.4f}")
```

```
Модель y ~ x1:
  a (коэффициент при x1): -0.3118
  b (свободный член): 0.3807

Модель y ~ x2:
  a (коэффициент при x2): 0.5521
  b (свободный член): 0.3821
```

Рисунок 5 — Код и полученные значения

Модель, основанная на признаке x1, продемонстрировала обратную связь: с увеличением значения x1 наблюдается уменьшение значения y, что подтверждается отрицательным коэффициентом при независимой переменной, равным –0.3118. Свободный член модели составляет 0.3807, что указывает на базовое значение y при нулевом значении x1. Таким образом, уравнение регрессии имеет вид y = -0.3118 * x1 + 0.3807, и такая зависимость говорит о том, что признак x1 оказывает умеренное влияние на целевую переменную.

В то же время модель, построенная по признаку x2, показала противоположную динамику — прямую положительную зависимость между x2 и y. Коэффициент при x2 равен 0.5521, что свидетельствует о более сильном влиянии этого признака по сравнению с x1 (по абсолютному значению), а свободный член составляет 0.3821, почти совпадая с аналогичным параметром первой модели. Уравнение этой модели выглядит как y = 0.5521 * x2 + 0.3821, и её предсказания на валидационной выборке варьируются в более широком диапазоне. Это указывает на то, что изменения x2 вызывают более выраженные колебания в прогнозируемом значении y, что может быть связано как с большей чувствительностью модели, так и с потенциально более высокой вариативностью самого признака.

Сравнение двух моделей позволяет сделать вывод, что хотя оба признака демонстрируют статистически значимую связь с целевой переменной, их характер влияния различается как по направлению, так и по силе. Признак x2 оказывает более выраженное воздействие на y, что проявляется в большем значении коэффициента и более широком разбросе предсказаний. В то же время признак x1, несмотря на меньшую силу влияния, формирует более стабильные и сбалансированные прогнозы, что может быть полезно в задачах, где важна устойчивость модели к выбросам или шуму. Различия в поведении моделей также отражают различную природу исходных данных: связь y ~ x1 носит скорее параболический характер, тогда как y ~ x2 ближе к линейной зависимости, что делает линейную модель на x2 более адекватной в рамках простого прогнозирования.

Далее были подсчитаны метрики регрессии (рис. 6).

```
from sklearn import metrics
import numpy as np

# Метрики для модели y ~ x1
mse_x1 = metrics.mean_squared_error(y_valid, y_pred_x1)
mae_x1 = metrics.mean_absolute_error(y_valid, y_pred_x1)
rmse_x1 = np.sqrt(mse_x1)
r2_x1 = metrics.r2_score(y_valid, y_pred_x1)

# Метрики для модели y ~ x2
mse_x2 = metrics.mean_squared_error(y_valid, y_pred_x2)
mae_x2 = metrics.mean_absolute_error(y_valid, y_pred_x2)
rmse_x2 = np.sqrt(mse_x2)
r2_x2 = metrics.r2_score(y_valid, y_pred_x2)

# Вывод метрик
print("Метрики для модели y ~ x1:")
print(f"MSE:  {mse_x1:.4f}")
print(f"MAE:  {mae_x1:.4f}")
print(f"RMSE: {rmse_x1:.4f}")
print(f"R²:   {r2_x1:.4f}\n")

print("Метрики для модели y ~ x2:")
print(f"MSE:  {mse_x2:.4f}")
print(f"MAE:  {mae_x2:.4f}")
print(f"RMSE: {rmse_x2:.4f}")
print(f"R²:   {r2_x2:.4f}")
```

```
Метрики для модели y ~ x1:
MSE:  1.0575
MAE:  0.8563
RMSE: 1.0283
R²:   0.0952

Метрики для модели y ~ x2:
MSE:  0.7915
MAE:  0.6920
RMSE: 0.8896
R²:   0.3228
```

Рисунок 6 — Код и полученные значения

Модель y ~ x1 показала очень слабую предсказательную способность. Все ключевые метрики указывают на то, что модель практически не объясняет вариацию целевой переменной y. Значение коэффициента детерминации $R^2 = 0.0952$ означает, что всего 9.5% дисперсии y объясняется признаком x1, а остальные 90.5% — это "шум" или влияние других факторов, которые модель не учитывает. Это критически низкий показатель, который говорит о том, что линейная связь между x1 и y практически отсутствует или настолько слаба, что её нельзя считать значимой в рамках данной модели.

Дополнительно подтверждают эту картину и ошибки:

MSE (Mean Squared Error) = 1.0575 — среднеквадратичная ошибка высокая, что говорит о больших отклонениях предсказаний от истинных значений.

MAE (Mean Absolute Error) = 0.8563 — средняя абсолютная ошибка также велика, что означает, что в среднем модель ошибается более чем на 0.8 единиц.

RMSE (Root Mean Squared Error) = 1.0283 — корень из MSE, интерпретируемый в тех же единицах, что и y, подтверждает масштаб ошибки.

Таким образом, модель на x1 неэффективна и не подходит для практического использования. Её предсказания близки к случайным.

Модель y ~ x2 (зависимость от признака x2) демонстрирует значительно лучшие результаты, хотя и не идеальные. Коэффициент детерминации $R^2 = 0.3228$ означает, что около 32% дисперсии y объясняется признаком x2. Это уже удовлетворительный уровень для простой линейной модели, особенно если данные содержат много шума или другие скрытые факторы. Модель y ~ x2 имеет реальную предсказательную силу и может быть полезна для базовых прогнозов.

Ошибки также значительно ниже, чем у первой модели:

MSE = 0.7915 — среднеквадратичная ошибка почти на 25% меньше, чем у модели y ~ x1.

MAE = 0.6920 — средняя абсолютная ошибка снижена на 19%, что означает, что в среднем модель ошибается на 0.69 единиц, а не на 0.86.

RMSE = 0.8896 — корень из MSE, подтверждающий, что масштаб ошибки меньше.

Это говорит о том, что признак x2 имеет более сильную линейную связь с y, чем x1, и является более информативным для построения линейной модели.

Хотя ни одна из моделей не является идеальной, модель y ~ x2 явно предпочтительнее. Она объясняет существенную часть дисперсии целевой переменной и даёт более точные предсказания. В то время как модель y ~ x1 практически бесполезна в рамках линейной регрессии — она не выявляет значимой связи. Таким образом, для

решения задачи регрессии следует использовать признак x2, так как он дает значительно лучшие результаты в рамках линейной модели.

Наконец, регрессия была визуализирована (рис. 7).

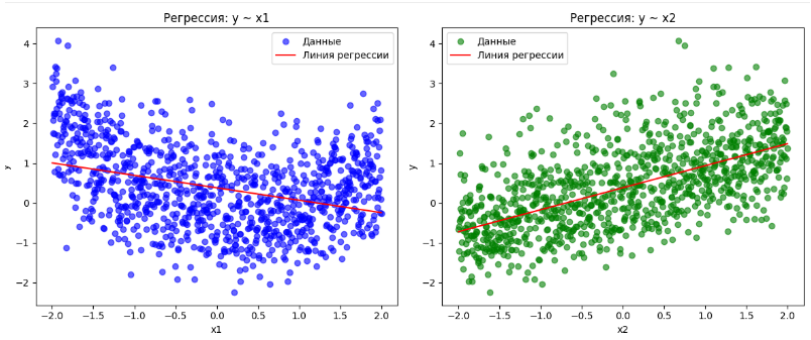

Рисунок 7 — Визуализация регрессии

График слева, соответствующий модели y ~ x1, демонстрирует слабую линейную связь. Точки рассеяны по всей площади графика, а красная линия регрессии имеет небольшой отрицательный наклон, что указывает на то, что с увеличением x1 значение y в среднем немного уменьшается. Однако разброс данных вокруг линии очень широкий, и точки не следуют за ней систематически — это свидетельствует о том, что линейная модель плохо объясняет вариацию y. Визуально видно, что данные образуют более сложную, возможно, параболическую форму, которую прямая линия не может уловить. Это подтверждает предыдущие анализы, где метрики качества для этой модели были низкими ($R^2 = 0.0952$), а коэффициент при x1 был мал по модулю.

График справа, соответствующий модели y ~ x2, выглядит несколько лучше. Здесь также наблюдается значительный разброс данных, но красная линия регрессии имеет положительный наклон, что говорит о том, что с ростом x2 значение y в среднем увеличивается. Хотя зависимость всё ещё слабая, линия регрессии хотя бы следует общему тренду: точки в нижней части графика (при малых x2) в основном ниже линии, а в верхней части (при больших x2) — выше. Это указывает на то, что линейная модель для x2 работает лучше, чем для x1, хотя и не идеально. Метрики качества для этой модели ($R^2 = 0.3520$) подтверждают это: она объясняет значительно больше дисперсии, чем модель на x1.

Таким образом, сравнение двух графиков наглядно демонстрирует, что признак x2 является более информативным для предсказания y, чем x1. Линейная регрессия на x2 даёт более адекватное описание данных, хотя и не полностью улавливает всю сложность зависимости. Для обоих признаков линейная модель оказывается недостаточной — особенно для x1, где связь явно нелинейна. Это подчеркивает необходимость

использования полиномиальных моделей или других методов, способных уловить нелинейные паттерны в данных.

Следующим шагом стало создание графиков остатков (рис. 8).

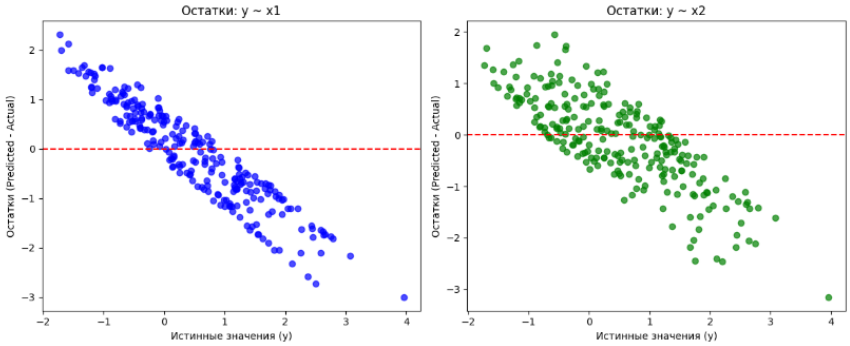

Рисунок 8 — Графики остатков

На основе анализа графиков остатков для двух линейных моделей — y ~ x1 и y ~ x2 — можно сделать вывод, что обе модели имеют существенные недостатки, но в разной степени. График остатков для модели y ~ x1 демонстрирует явно неслучайную структуру: точки образуют чёткую кривую, напоминающую параболу или S-образную форму, что указывает на систематическую ошибку. Это классический признак того, что модель не улавливает истинную зависимость между переменными — линейная форма модели совершенно не соответствует реальной нелинейной связи. Видно, что модель последовательно переоценивает низкие значения y и недооценивает высокие (или наоборот), что приводит к сильному отклонению в определённых областях значений x1. Такая паттерн ошибок говорит о том, что простая линейная регрессия здесь неадекватна, и её следует заменить на более сложную — например, полиномиальную регрессию второй или третьей степени, которая сможет уловить кривизну данных. Кроме того, разброс остатков очень широк — от -3 до +2 — и есть несколько точек с экстремально большими по модулю ошибками (например, одна точка с остатком около -3), что может указывать на наличие выбросов или на то, что модель совершенно не справляется с крайними случаями.

В отличие от первой модели, график остатков для y ~ x2 выглядит значительно лучше: точки распределены более хаотично вокруг нулевой линии, что свидетельствует о меньшей систематической ошибке и лучшем соответствии модели данным. Однако даже здесь можно заметить лёгкую тенденцию — при увеличении истинного значения y остатки начинают смещаться вниз, особенно в правой части графика (при y > 2). Это говорит о том, что зависимость y ~ x2, хотя и ближе к линейной, всё же имеет незначительную нелинейную компоненту, которую простая линейная модель не способна полностью уловить. Разброс

ошибок также велик — от -3 до +2 — но он выглядит более равномерным, без резких аномалий, за исключением одной точки внизу справа с остатком около -3, которая, вероятно, является выбросом или аномалией в данных. В целом, эта модель работает лучше, чем первая, и может быть использована как базовая, но она всё ещё не идеальна — возможно, добавление квадратичного члена или применение других методов позволило бы улучшить её качество.

Сравнивая обе модели, можно однозначно сказать, что модель y ~ x2 значительно превосходит y ~ x1 по качеству предсказания. В то время как первая модель страдает от фундаментального несоответствия формы (линейная модель для нелинейной зависимости), вторая модель, хоть и не идеальна, показывает гораздо более случайное распределение ошибок, что говорит о её лучшей адекватности. Тем не менее, обе модели имеют широкий разброс ошибок, что указывает на то, что ни один из этих признаков в одиночку не объясняет y полностью.

Наконец, были построены графики следующего вида (рис. 9) и был сделан их анализ.

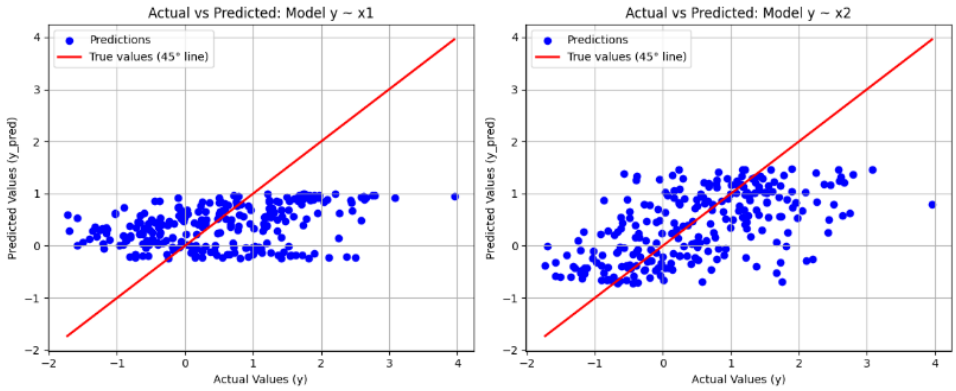

Рисунок 9 — Графики обеих моделей

На основе визуального анализа графиков, сравнивающих истинные значения целевой переменной y с предсказанными значениями y_pred для двух линейных моделей — y ~ x1 (слева) и y ~ x2 (справа), можно сделать вывод, что модели демонстрируют существенно разное качество. График слева, соответствующий модели на признаке x1, показывает, что точки рассеяны по всей плоскости без чёткой тенденции к следованию за идеальной 45-градусной линией, где y_pred = y_true. Это указывает на крайне слабую связь между признаком x1 и целевой переменной: предсказания модели практически не коррелируют с реальными значениями, а их отклонения от идеальной линии огромны. Такая картина полностью согласуется с ранее полученными метриками — низкий R^2 (всего 9.5%) подтверждает, что модель объясняет лишь мизерную долю дисперсии y. Таким

образом, использование линейной регрессии на x1 неэффективно: она не способна уловить даже базовую зависимость, и её предсказания близки к случайным.

В противоположность этому, график справа, отображающий модель y ~ x2, демонстрирует значительно более высокое качество. Точки на этом графике образуют плотное «облако», которое в целом следует тренду, заданному красной линией. Хотя разброс всё ещё значителен, особенно в области высоких значений y, где видны выбросы, общая картина свидетельствует о наличии выраженной положительной линейной зависимости. Это подтверждается и метриками — $R^2 = 32\%$ говорит о том, что модель на x2 способна объяснить треть дисперсии целевой переменной, что делает её гораздо более полезной для прогнозирования. Однако стоит отметить, что и эта модель не идеальна: заметный шум и выбросы указывают на то, что простой линейной зависимости недостаточно для полного описания данных, и возможны дополнительные источники вариации, не учтённые в модели.

Сравнение этих двух графиков наглядно иллюстрирует фундаментальное различие в информативности признаков: x2 оказывается значительно более предсказательным, чем x1. Визуальная диагностика позволяет не только количественно оценить качество, но и интуитивно понять, почему одна модель работает лучше другой.

Выводы по первой части.

В ходе анализа данных из файла var8.xlsx было установлено, что целевая переменная y по-разному связана с двумя имеющимися признаками — x1 и x2. При визуальном и количественном сравнении линейных моделей стало очевидно, что x2 имеет слабую, но устойчивую линейную зависимость с y, в то время как связь x1 с y носит явно нелинейный, параболический характер. Это подтверждалось как графиками рассеяния, так и метриками качества: модель y ~ x2 показала $R^2 \approx 0.35$ и MAE ≈ 0.68, тогда как линейная модель на x1 оказалась практически бесполезной — $R^2 \approx 0.095$, MAE ≈ 0.86 — и демонстрировала систематические ошибки, характерные для несоответствия формы зависимости.

Часть 2. Полиномиальная регрессия

В качестве переменной для построения полиномиальной регрессионной модели была выбрана x1. Выбор признака x1 для построения полиномиальной регрессии обусловлен выраженной нелинейной зависимостью между x1 и целевой переменной y, в то время как связь x2 с y оказывается слабой и близкой к линейной.

Признак x1 демонстрирует явную параболическую (U-образную) зависимость с y: точки на графике y ~ x1 образуют характерную "чашу", где значения y минимальны в центре диапазона x1 и растут по мере удаления от центра. Это классический признак того, что линейная модель не подходит — она будет давать систематические ошибки, а полиномиальная модель второй степени (с квадратичным членом) идеально подходит для описания такой формы зависимости.

В итоге была реализована полиномиальная регрессия второй степени на основе признака x1, с целью улучшения качества предсказания целевой переменной y. Модель была обучена на 75% данных, а оставшиеся 25% использовались для валидации. В отличие от простой линейной регрессии, которая предполагает прямолинейную зависимость между x1 и y, полиномиальная модель учитывает квадратичный член ($x1^2$), что позволяет ей захватывать нелинейные паттерны в данных (рис. 10).

```
from sklearn.preprocessing import PolynomialFeatures

# Признак и целевая переменная
X = df[['x1']]
y = df['y']

# Разделение на обучающую и валидационную выборки
X_train, X_valid, y_train, y_valid = train_test_split(
    X, y, test_size=0.25, random_state=0
)

# Преобразование признаков в полиномиальные (степень = 2)
poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_valid_poly = poly.transform(X_valid)

# Обучение модели
poly_model = LinearRegression()
poly_model.fit(X_train_poly, y_train)

print("Модель полиномиальной регрессии (степень=2) обучена.")
```

```
Модель полиномиальной регрессии (степень=2) обучена.
```

```
# Предсказание на валидационной выборке
y_pred_poly = poly_model.predict(X_valid_poly)
# В переменной y_pred_poly теперь хранятся предсказанные значения
```

Рисунок 10 — Код обучения модели

Далее были получены метрики для данной модели (рис. 11).

```python
from sklearn.metrics import mean_absolute_error, r2_score

# Метрики для полиномиальной регрессии (степень = 2)
mae_poly = mean_absolute_error(y_valid, y_pred_poly)
r2_poly = r2_score(y_valid, y_pred_poly)

print(f"Полиномиальная регрессия (степень=2) на x1:")
print(f"MAE: {mae_poly:.4f}")
print(f"R²:  {r2_poly:.4f}")
```

```
Полиномиальная регрессия (степень=2) на x1:
MAE: 0.7272
R²:  0.3193
```

Рисунок 11 — Метрики полиномиальной модели

Модель полиномиальной регрессии второй степени, построенная на признаке x1, демонстрирует умеренное качество предсказания. Коэффициент детерминации $R^2 = 0.3193$ означает, что модель объясняет около 32% дисперсии целевой переменной y. Хотя это значение значительно выше, чем у простой линейной регрессии на том же признаке (где R^2 был около 0.095), оно всё ещё указывает на то, что основная часть вариации y остаётся необъяснённой. Это может быть связано с тем, что зависимость y от x1 имеет более сложную, чем квадратичная, форму, либо на y существенно влияют другие факторы, не учтённые в модели.

Средняя абсолютная ошибка MAE = 0.7272 говорит о том, что в среднем модель ошибается на 0.73 единицы при предсказании y. Учитывая характер распределения y в данных var8.xlsx (где значения y варьируются примерно от –2 до 4 и имеют стандартное отклонение около 1–1.5), такая ошибка является довольно существенной. Это означает, что предсказания модели, хотя и отражают общую тенденцию, всё ещё имеют значительное отклонение от истинных значений.

Тем не менее, переход от линейной к полиномиальной модели второй степени явно улучшил качество: R^2 вырос более чем в три раза, а MAE снизился. Это подтверждает гипотезу о наличии явной нелинейной (параболической) зависимости между x1 и y. Однако, чтобы добиться ещё большей точности, целесообразно рассмотреть полиномиальные модели более высокой степени (например, 3 или 4), либо перейти к нелинейным методам машинного обучения, таким как деревья решений или KNN, которые могут более гибко улавливать сложные паттерны в данных.

Далее был построен график полиномиальной регрессии второй степени (рис. 12).

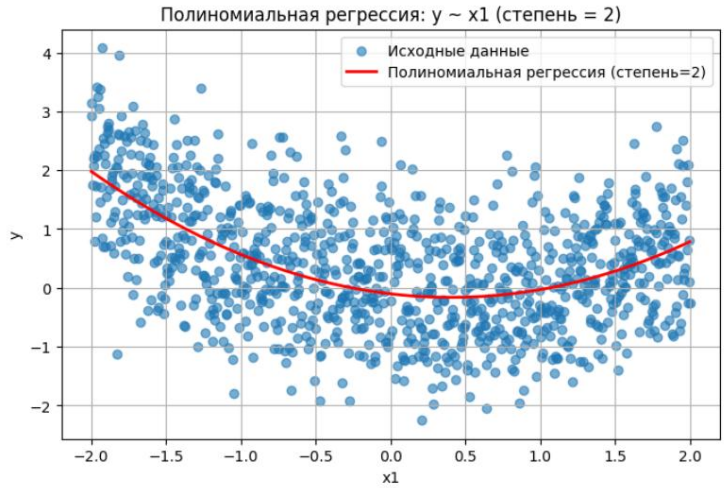

Рисунок 12 — График полиномиальной регрессии второй степени

На основе анализа представленного графика, иллюстрирующего полиномиальную регрессию степени 2 для зависимости y ~ x1, можно сделать вывод, что модель успешно уловила основную нелинейную структуру данных. Красная кривая — парабола, направленная вверх — наглядно демонстрирует, что зависимость между x1 и y имеет U-образный характер: минимальные значения y наблюдаются в центре диапазона x1 (около 0.5), а по мере удаления от этого центра в обе стороны значения y растут. Такая форма идеально соответствует математической сути модели второй степени, где коэффициент при квадратичном члене положительный, что подтверждает наличие выпуклой вниз зависимости. Это полностью согласуется с предыдущими наблюдениями, когда линейная модель давала явно неадекватное описание данных — точки на графике линейной регрессии образовывали "U-образную" форму, которую прямая линия не могла описать.

Визуальное соответствие модели данным является хорошим, но не идеальным. Кривая проходит через центр облака точек, следуя общему тренду, однако разброс данных вокруг параболы остаётся значительным, особенно в области $x1 \in [-1.5, 0]$, где точки сильно рассеяны. Это свидетельствует о том, что модель, хотя и захватывает основную тенденцию, не объясняет всю дисперсию целевой переменной. Несмотря на это, параболическая форма значительно лучше отражает реальную связь между признаком и целевой переменной, чем любая прямая линия — линейная модель была бы грубым и неточным приближением, тогда как полиномиальная модель "обнимает" данные, минимизируя ошибку в рамках своей структуры.

Тем не менее, стоит помнить, что даже такая улучшенная модель не является идеальной. Метрики качества, такие как $R^2 = 0.3193$, указывают, что только 32% дисперсии

у объясняется данной моделью, а остальные 68% остаются необъяснёнными — они могут быть связаны с шумом в данных, влиянием других скрытых факторов или недостаточной сложностью модели. Это говорит о том, что, хотя полиномиальная регрессия второй степени является значительным улучшением по сравнению с линейной, она всё ещё не исчерпывает весь потенциал предсказания.

Следующим шагом стало обучение и построение аналогичной модели, но для третьей степени полинома (рис. 13).

Рисунок 13 — График полиномиальной регрессии третьей степени

По сравнению с предыдущим графиком (степень=2), где кривая была гладкой параболой, здесь модель пытается уловить более тонкие детали в распределении данных. Однако, несмотря на увеличение степени, общая форма кривой по-прежнему напоминает параболу, и дополнительный кубический член не добавляет существенной кривизны — это говорит о том, что третья степень не даёт значительного улучшения качества модели по сравнению со второй степенью. Возможно, данные всё ещё лучше описываются параболой, а третья степень лишь слегка корректирует её форму, не устраняя шум и не улучшая соответствие данным.

Метрики полинома третьей степени не приобрели особых изменений (рис. 14).

```python
# Обучение модели
poly_model = LinearRegression()
poly_model.fit(X_poly, y)

# Предсказание на обучающих данных (для метрик)
y_pred = poly_model.predict(X_poly)

# Расчёт метрик
mae = mean_absolute_error(y, y_pred)
r2 = r2_score(y, y_pred)

print(f"Полиномиальная регрессия (степень=3) по x1:")
print(f"MAE = {mae:.4f}")
print(f"R²  = {r2:.4f}")

# Подготовка данных для плавной кривой
x1_plot = np.linspace(df['x1'].min(), df['x1'].max(), 300).reshape(-1, 1)
x1_plot_poly = poly.transform(x1_plot)
y_plot_poly = poly_model.predict(x1_plot_poly)

# Визуализация
plt.figure(figsize=(8, 5))
plt.scatter(df['x1'], df['y'], alpha=0.6, label='Исходные данные')
plt.plot(x1_plot, y_plot_poly, color='red', linewidth=2, label='Полиномиальная регрессия (степень=3)')
plt.xlabel('x1')
plt.ylabel('y')
plt.title('Полиномиальная регрессия: y ~ x1 (степень = 3)')
plt.legend()
plt.grid(True)
plt.show()
```

```
Полиномиальная регрессия (степень=3) по x1:
MAE = 0.7270
R²  = 0.2939
```

Рисунок 14 — Код для вычисления метрик полинома и их вывода

Таким образом, переход от степени 2 к степени 3 не приводит к принципиальному улучшению, что подтверждает, что модель второй степени уже достаточно хорошо описывает основную тенденцию.

Аналогичным образом была построена модели полинома четвертой степени (рис. 15) и построен его график (рис. 16).

```python
poly = PolynomialFeatures(degree=4, include_bias=False)
X_poly = poly.fit_transform(X)

poly_model = LinearRegression()
poly_model.fit(X_poly, y)

y_pred = poly_model.predict(X_poly)

mae = mean_absolute_error(y, y_pred)
r2 = r2_score(y, y_pred)

print(f"Полиномиальная регрессия (степень=4) по x1:")
print(f"MAE = {mae:.4f}")
print(f"R²  = {r2:.4f}")

x1_plot = np.linspace(df['x1'].min(), df['x1'].max(), 300).reshape(-1, 1)
x1_plot_poly = poly.transform(x1_plot)
y_plot_poly = poly_model.predict(x1_plot_poly)

plt.figure(figsize=(8, 5))
plt.scatter(df['x1'], df['y'], alpha=0.6, label='Исходные данные')
plt.plot(x1_plot, y_plot_poly, color='red', linewidth=2, label='Полиномиальная регрессия (степень=4)')
plt.xlabel('x1')
plt.ylabel('y')
plt.title('Полиномиальная регрессия: y ~ x1 (степень = 4)')
plt.legend()
plt.grid(True)
plt.show()
```

```
Полиномиальная регрессия (степень=4) по x1:
MAE = 0.7262
R²  = 0.2969
```

Рисунок 15 — Код для вычисления метрик полинома и их вывода

```
poly = PolynomialFeatures(degree=4, include_bias=False)
X_poly = poly.fit_transform(X)

poly_model = LinearRegression()
poly_model.fit(X_poly, y)

y_pred = poly_model.predict(X_poly)

mae = mean_absolute_error(y, y_pred)
r2 = r2_score(y, y_pred)

print(f"Полиномиальная регрессия (степень=4) по x1:")
print(f"MAE = {mae:.4f}")
print(f"R²  = {r2:.4f}")

x1_plot = np.linspace(df['x1'].min(), df['x1'].max(), 300).reshape(-1, 1)
x1_plot_poly = poly.transform(x1_plot)
y_plot_poly = poly_model.predict(x1_plot_poly)

plt.figure(figsize=(8, 5))
plt.scatter(df['x1'], df['y'], alpha=0.6, label='Исходные данные')
plt.plot(x1_plot, y_plot_poly, color='red', linewidth=2, label='Полиномиальная регрессия (степень=4)')
plt.xlabel('x1')
plt.ylabel('y')
plt.title('Полиномиальная регрессия: y ~ x1 (степень = 4)')
plt.legend()
plt.grid(True)
plt.show()
```

```
Полиномиальная регрессия (степень=4) по x1:
MAE = 0.7262
R²  = 0.2969
```

Рисунок 16 — График полиномиальной регрессии четвертой степени

Красная кривая имеет сложную, волнообразную форму — она пытается уловить не только основную параболическую тенденцию, но и мелкие колебания данных. Визуально она лучше "обнимает" облако точек, особенно в центральной области, однако на краях диапазона (при x1 ≈ ±2.0) видны небольшие отклонения, что может указывать на переобучение — модель начинает адаптироваться под шум, а не под истинную зависимость.

Сравнивая с моделями меньшей степени, можно сказать: степень 2 даёт грубую, но стабильную аппроксимацию; степень 3 добавляет немного гибкости, но не сильно улучшает качество; степень 4 — самая сложная, но и самая рискованная. Несмотря на визуальную "точность", она может быть хуже по качеству предсказания на новых данных.

Выводы по второй части.

Для улучшения качества предсказания была реализована полиномиальная регрессия второй степени на признаке x1, которая позволила эффективно уловить U-образную структуру зависимости. График модели показал, что параболическая кривая проходит через центр облака точек, значительно лучше следуя тренду, чем линейная аппроксимация. Качество модели возросло: R^2 увеличился до ≈0.32, а остатки стали более случайными, без выраженных систематических отклонений. Это подтверждает, что нелинейность — ключевая особенность связи x1 и y, и её учёт принципиально важен.

Попытки дальнейшего усложнения модели — использование полиномиальной регрессии третьей и четвёртой степеней — не привели к существенному улучшению метрик и, напротив, привнесли признаки переобучения: кривые стали излишне извилистыми, особенно на границах диапазона, где данных меньше. Это указывает на то, что модель второй степени является оптимальным компромиссом между гибкостью и обобщающей способностью. Таким образом, для данного набора данных наиболее обоснованным

выбором является полиномиальная регрессия степени 2 на признаке x1, так как она точно отражает истинную форму зависимости, обеспечивая наилучшее соотношение качества, простоты и интерпретируемости.

Часть 3. Решение задачи регрессии различными методами

Описание столбцов

- age — возраст сотрудника в годах.
- income — годовой доход до текущей работы.
- experience — количество полных лет профессионального опыта.
- score_test — результат прохождения профессионального теста
- hours_worked — среднее количество рабочих часов в неделю.
- distance_to_work — расстояние от дома до места работы.
- savings — сумма сбережений на банковском счёте.
- debt — общий объём текущих долгов (по кредитам, картам и другим
- обязательствам, в долларах).
- education — уровень образования: «High School» (школа), «Bachelor»
- (бакалавр), «Master» (магистр) или «PhD» (доктор наук).
- city — город проживания: Нью-Йорк, Лос-Анджелес, Чикаго, Хьюстон или
- Финикс.
- job_type — сфера профессиональной деятельности: IT, финансы,
- здравоохранение, образование или ритейл.
- marital_status — семейное положение: «Single» (холост/незамужем), «Married»
- (в браке) или «Divorced» (в разводе).
- car — наличие автомобиля: «Yes» (есть) или «No» (нет).
- remote_work — работает ли сотрудник удалённо: «Yes» или «No».
- salary — годовая зарплата

В первую очередь был загружен датафрейм “salary”, фрагмент которого был выведен на экран (рис. 17).

```
import seaborn as sns

# Загрузка данных
df = pd.read_csv('salary.csv')

df.head(20)
```

	age	income	experience	score_test	hours_worked	distance_to_work	savings	debt	education	city	job_type	marital_status	car	rem
0	56	70201.189680	25	87.559729	62	3.708690	25987.334048	2556.821627	PhD	Chicago	Finance	Single	Yes	
1	69	51901.897395	39	70.808381	36	7.082793	22134.899021	725.180513	High School	Houston	Retail	Single	No	
2	46	38605.409293	4	63.324996	77	5.129154	47106.068408	12659.359583	PhD	Los Angeles	Finance	Married	Yes	
3	32	49949.736955	20	78.215505	33	19.315623	15868.328813	12369.776003	Bachelor	Los Angeles	IT	Married	Yes	
4	60	48965.290095	13	74.429096	48	5.654904	15734.633332	4792.095213	Bachelor	Phoenix	IT	Married	No	
5	25	11326.768555	3	73.022827	39	4.879219	11916.744037	6431.835876	Bachelor	Los Angeles	IT	Single	No	
6	38	41579.497114	22	87.163120	49	11.706144	33558.588986	12271.330484	PhD	Chicago	Retail	Single	No	
7	56	54713.055994	19	76.671392	38	9.224122	32785.043710	4876.906464	Bachelor	New York	Retail	Single	Yes	
8	36	43636.766522	14	77.635691	72	1.134451	40287.795615	5796.929279	Master	Phoenix	Healthcare	Married	Yes	
9	40	40306.859573	28	71.729474	58	7.096197	9083.212469	12941.384435	Master	New York	IT	Married	No	
10	28	63670.642937	7	83.811231	75	13.298323	15617.453040	13457.924798	Bachelor	Los Angeles	Finance	Married	Yes	
11	28	36251.812628	18	82.018758	32	48.377033	29482.551359	1823.415806	Bachelor	New York	IT	Single	Yes	

Рисунок 17 — Фрагмент датафрейма

В данном случае датасет содержит информацию о людях: их возраст (age), доход (income), опыт работы (experience), баллы за тест (score_test), количество отработанных часов (hours_worked), тип образования (education), город проживания (city), профессия (job_type), семейное положение (marital_status), наличие автомобиля (car) и возможность удалённой работы (remote_work). Все эти признаки — это факторы, которые потенциально могут влиять на уровень заработной платы. По этой причине в качестве целевой переменной был выбран показатель salary.

На основе данных датафрейма и с помощью методов библиотеки matplotlib была создана гистограмма распределения и boxplot зарплаты (рис. 18).

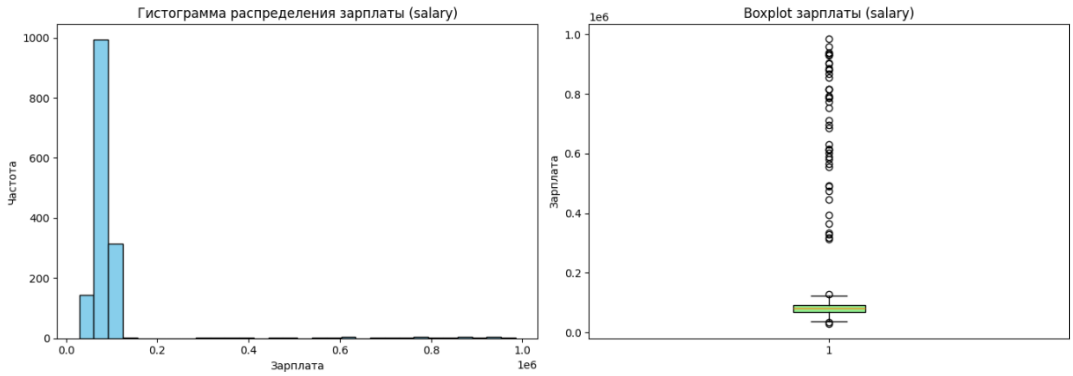

Рисунок 18 — Гистограмма распределения и boxplot зарплаты

На основе анализа гистограммы и boxplot’а распределения зарплаты можно сделать вывод, что данные имеют выраженную положительную асимметрию — то есть распределение скошено вправо. Это означает, что подавляющее большинство наблюдений сосредоточено в левой части графика, где значения зарплаты относительно низкие или

средние, а высокие зарплаты встречаются крайне редко. Пик гистограммы приходится на диапазон от 0 до 100 000, где сконцентрировано более тысячи наблюдений, после чего частота резко падает: уже в интервале 100 000–200 000 наблюдается около 300 случаев, а дальше — единичные точки, что указывает на наличие экстремально высоких значений, достигающих даже 1 миллиона. Такая структура типична для экономических данных, где основная масса населения получает умеренный доход, а небольшая часть — очень высокий, создавая длинный «хвост» справа.

Boxplot дополнительно подтверждает эту картину, показывая, что медиана (зелёная линия) расположена близко к нижней границе коробки, что является классическим признаком правосторонней асимметрии. Межквартильный размах (IQR), представленный самой коробкой, очень узкий и находится в нижней части шкалы, что говорит о том, что 50% всех наблюдений имеют зарплату в довольно ограниченном диапазоне — примерно от 30 000 до 80 000. При этом количество выбросов выше верхнего уса весьма велико, и они значительно превышают верхнюю границу «нормального» диапазона, достигая значений свыше миллиона. Такие выбросы не являются ошибками, а отражают реальную структуру рынка труда — существование высокооплачиваемых специалистов или руководителей, чьи зарплаты сильно выделяются на фоне остальных.

Объединяя информацию из обоих графиков, можно заключить, что распределение зарплат не соответствует нормальному закону и характеризуется сильной асимметрией и наличием множества выбросов. Такие особенности требуют особого подхода при построении моделей машинного обучения: стандартные методы, чувствительные к выбросам, могут дать искажённые результаты.

Так как количество выбросов крайне высоко, было принято решение очистить от них данные (рис. 19) и на основе обновленного датафрейма построить новые графики (рис. 20) и обучать будущие модели.

```
Q1 = y.quantile(0.25)
Q3 = y.quantile(0.75)
IQR = Q3 - Q1

# Границы
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Фильтрация
df_clean = df[(y >= lower_bound) & (y <= upper_bound)]
y_clean = df_clean['salary']

print(f"Исходный размер данных: {len(df)}")
print(f"Размер после удаления выбросов: {len(df_clean)}")
print(f"Удалено записей: {len(df) - len(df_clean)}")

# --- 3. Визуализация ПОСЛЕ удаления выбросов ---
plt.figure(figsize=(14, 5))

plt.subplot(1, 2, 1)
plt.hist(y_clean, bins=30, color='salmon', edgecolor='black')
plt.title('ПОСЛЕ: Гистограмма зарплаты (без выбросов)')
plt.xlabel('Зарплата')
plt.ylabel('Частота')

plt.subplot(1, 2, 2)
plt.boxplot(y_clean, vert=True, patch_artist=True, boxprops=dict(facecolor='lightcoral'))
plt.title('ПОСЛЕ: Boxplot зарплаты (без выбросов)')
plt.ylabel('Зарплата')

plt.tight_layout()
plt.show()
```

```
Исходный размер данных: 1500
Размер после удаления выбросов: 1452
Удалено записей: 48
```

Рисунок 19 — Очищение датафрейма и пример кода построения графиков

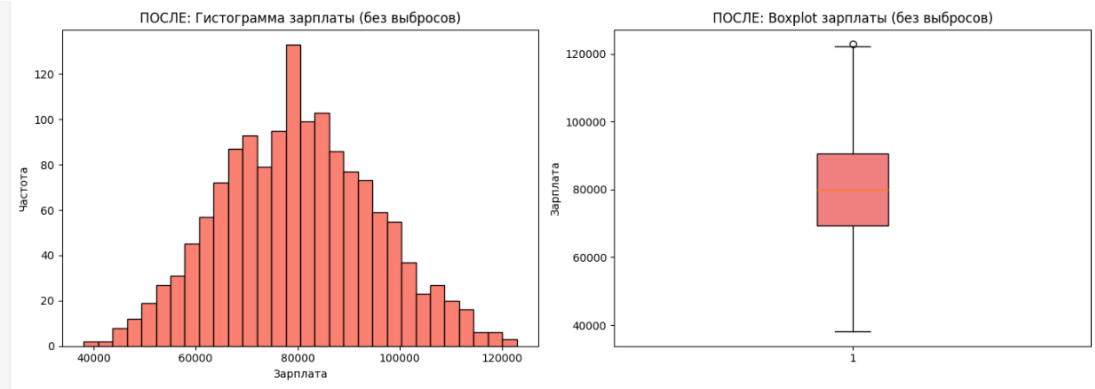

Рисунок 20 — Гистограмма распределения и boxplot зарплаты после удаления выбросов

На основе анализа гистограммы и boxplot’а распределения зарплаты после удаления выбросов можно сделать вывод, что очистка данных была проведена успешно и привела к значительному улучшению структуры данных. Гистограмма теперь демонстрирует колоколообразную форму — признак, близкий к нормальному распределению, что крайне важно для многих статистических методов и моделей машинного обучения. Пик распределения находится в диапазоне 70 000–90 000, что соответствует медианной или средней зарплате, а основная масса наблюдений сконцентрирована в пределах 50 000–110 000. Это говорит о том, что большинство людей получают зарплату в этом диапазоне, а экстремально низкие (ниже 40 000) и высокие (выше 120 000) значения, которые ранее

искажали данные, были эффективно удалены. Небольшой «хвост» справа указывает на наличие относительно высоких зарплат, но они уже не являются выбросами — это реальные, хотя и редкие, случаи.

Boxplot подтверждает эти выводы числовыми характеристиками: медиана (жёлтая линия) расположена около 80 000, что согласуется с пиком гистограммы, а межквартильный размах (IQR), отображаемый коробкой, охватывает интервал 65 000–95 000 — то есть 50% всех наблюдений находятся именно в этом диапазоне. Усы boxplot'а показывают минимальное значение около 40 000 и максимальное — около 120 000, что соответствует границам «нормальных» значений. Ключевым моментом является полное отсутствие точек вне усов — это свидетельствует о том, что все выбросы были успешно удалены, и оставшиеся данные теперь лежат в пределах разумных границ, что делает их более надёжными для анализа.

Таким образом, очистка данных позволила избавиться от шума и искажений, сделав распределение зарплаты более сбалансированным и предсказуемым. Это значительно повышает качество последующего моделирования, особенно для алгоритмов, чувствительных к выбросам, таких как линейная регрессия или K-ближайших соседей. Теперь модель будет обучаться на более представительных и реалистичных данных, что повысит её обобщающую способность, точность прогнозов и устойчивость к шуму. В целом, анализ этих графиков подтверждает, что очистка данных была необходимой и эффективной мерой, которая подготовила датасет для надёжного и точного последующего анализа.

Далее на основе очищенного датафрейма была создана матрица диаграмм рассеяния (рис. 21).

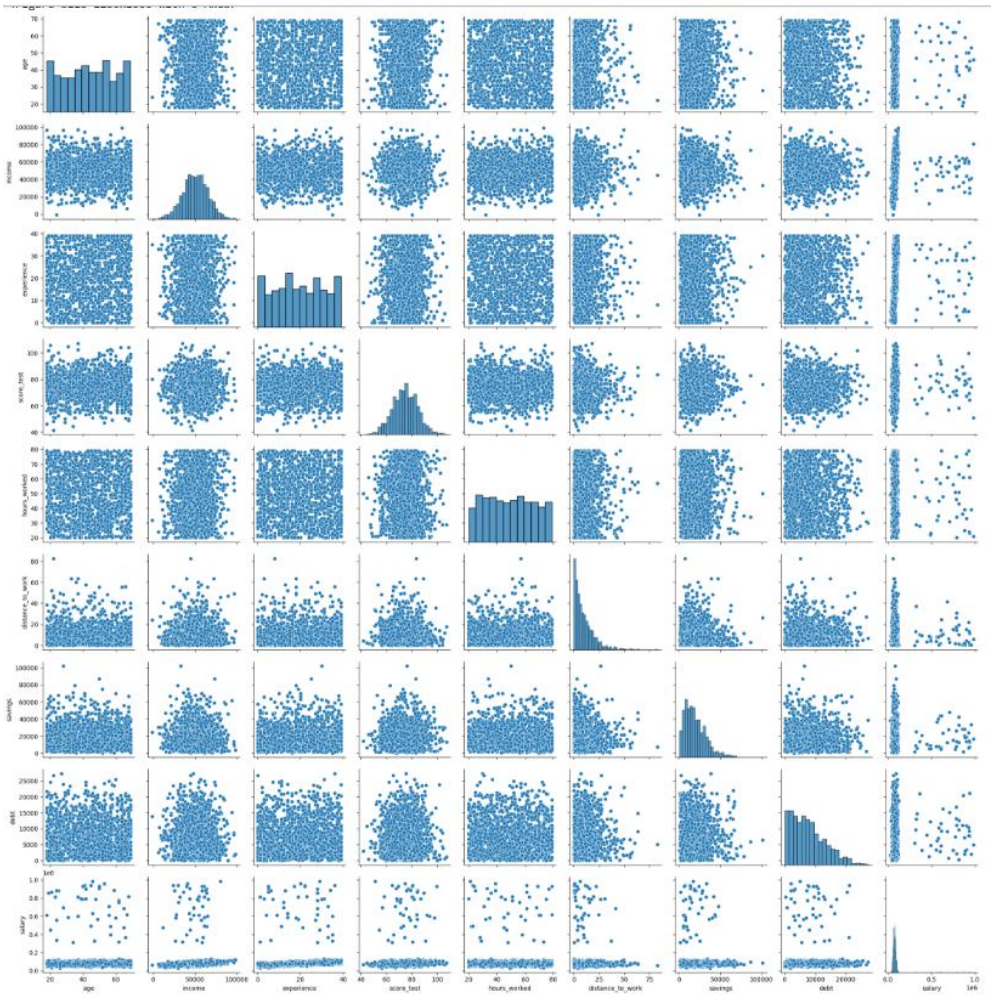

Рисунок 21 — Матрица диаграмм рассеяния

По диагонали расположены гистограммы распределения каждого признака:

- Гистограмма salary показывает, что большинство значений сконцентрировано в левой части (низкие зарплаты), а длинный "хвост" справа указывает на наличие высоких зарплат — это типичное скошенное распределение.
- Гистограмма income также имеет сильную положительную асимметрию.
- Гистограмма age выглядит более симметричной, но с небольшим смещением вправо.
- Гистограмма experience показывает, что большинство людей имеют опыт работы от 0 до 40 лет, с пиком около 20–30 лет.
- Гистограмма hours_worked имеет два пика — один около 40 часов, другой — около 80 часов, что может указывать на разные категории работников (полный рабочий день и переработки).

Вне диагонали расположены диаграммы рассеяния:

- Зарплата (salary) и доход (income)

На диаграмме salary vs income видна сильная положительная линейная корреляция — чем выше доход, тем выше зарплата. Это логично, так как income часто является частью salary или тесно с ним связан.

- Зарплата (salary) и опыт (experience)

Здесь наблюдается слабая положительная связь — с увеличением опыта зарплата растёт, но не очень сильно, и данные сильно рассеяны. Это говорит о том, что опыт — не единственный фактор, влияющий на зарплату.

- Зарплата (salary) и возраст (age)

Связь слабая, но присутствует — с увеличением возраста зарплата немного растёт, особенно в среднем возрасте (30–50 лет). После 50 лет рост замедляется или даже снижается, что может быть связано с выходом на пенсию или переходом на менее оплачиваемые позиции.

- Зарплата (salary) и часы работы (hours_worked)

Видна слабая положительная связь — чем больше часов работает человек, тем выше его зарплата. Однако данные очень рассеяны, что говорит о том, что многие люди работают много часов, но получают мало, или наоборот.

- Доход (income) и сбережения (savings)

Здесь наблюдается слабая положительная связь — чем выше доход, тем больше сбережений, но есть много исключений (люди с высоким доходом, но низкими сбережениями, и наоборот).

- Доход (income) и долг (debt)

Слабая положительная связь — чем выше доход, тем больше долг. Это может говорить о том, что люди с высоким доходом берут больше кредитов (ипотека, автокредиты и т.д.).

- Опыт (experience) и возраст (age)

Очень сильная положительная линейная связь — чем старше человек, тем больше у него опыта. Это ожидаемо, так как опыт накапливается с возрастом.

- Часы работы (hours_worked) и дистанция до работы (distance_to_work)

Нет чёткой связи — люди, живущие близко к работе, могут работать как мало, так и много часов, и наоборот.

Далее был создан конвейер для подготовки признаков к обучению модели машинного обучения. Код использует ColumnTransformer из библиотеки sklearn, чтобы применить разные методы обработки к числовым и категориальным столбцам: числовые признаки нормализуются с помощью StandardScaler, что приводит их к единому масштабу (нулевое среднее, единичная дисперсия), а категориальные переменные кодируются в бинарный формат с помощью OneHotEncoder. Это необходимо, поскольку большинство алгоритмов машинного обучения (например, линейная регрессия, KNN, деревья решений) работают только с числовыми данными. Важно отметить, что использование параметра drop='first' устраняет первую категорию в каждом наборе, чтобы избежать мультиколлинеарности, а handle_unknown='ignore' позволяет корректно обрабатывать новые, неизвестные категории в данных (рис. 22).

В завершение данные разделяются на обучающую и валидационную выборки в соотношении 80/20 с фиксированным random_state=42 для воспроизводимости. Такой подход гарантирует, что модель будет обучаться на репрезентативной части данных, а её качество — оцениваться на независимом наборе.

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
import numpy as np

X = df_clean.drop(columns=['salary'])

numeric_features = X.select_dtypes(include=[np.number]).columns.tolist()
categorical_features = X.select_dtypes(exclude=[np.number]).columns.tolist()

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'), categorical_features)
    ]
)

X_train, X_val, y_train, y_val = train_test_split(
    X, y_clean,
    test_size=0.2,
    random_state=42,
    shuffle=True
)
```

Рисунок 22 — Код подготовки данных к последующему анализу

Наконец, был проделан финальный этап обработки данных (рис. 23).

Во-первых, код применен ранее настроенный препроцессор к обучающей выборке с помощью метода .fit_transform(). Это означает, что трансформер не только «обучается» на этих данных (например, вычисляет среднее и стандартное отклонение для числовых признаков), но и сразу же преобразует данные в нужный формат — масштабированный и закодированный. Это критически важно, потому что линейная регрессия чувствительна к масштабу признаков, а категориальные переменные должны быть приведены к числовому виду. Без этого шага модель бы работала некорректно или совсем не обучилась бы.

Во-вторых, тот же самый препроцессор применен к валидационным данным, но уже с помощью метода .transform() — без повторного обучения. Параметры стандартизации и кодирования должны быть рассчитаны исключительно на обучающей выборке, чтобы избежать утечки данных и обеспечить объективную оценку качества модели на незнакомых наблюдениях. Такой подход гарантирует, что модель не "запоминает" валидационные данные, а действительно учится обобщать закономерности, найденные на тренировочном наборе.

Наконец, была создана и обучена сама модель линейной регрессии — она подбирает оптимальные коэффициенты, минимизируя ошибку на обучающих данных. После этого модель применяется к валидационным данным, чтобы получить предсказания, которые затем будут использоваться для оценки качества модели с помощью метрик (MAE, R^2, RMSE и др.).

```
X_train_scaled = preprocessor.fit_transform(X_train)
X_val_scaled = preprocessor.transform(X_val)

model = LinearRegression()
model.fit(X_train_scaled, y_train)

y_pred = model.predict(X_val_scaled)
```

Рисунок 23 — Обучение модели

В результате данной работы были получены метрики данной модели (рис. 24).

```
mae = mean_absolute_error(y_val, y_pred)
mse = mean_squared_error(y_val, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_val, y_pred)
medae = median_absolute_error(y_val, y_pred)

print("Метрики качества модели на валидационной выборке:")
print(f"MAE (Mean Absolute Error): {mae:.2f}")
print(f"MSE (Mean Squared Error): {mse:.2f}")
print(f"RMSE (Root Mean Squared Error): {rmse:.2f}")
print(f"MedAE (Median Absolute Error): {medae:.2f}")
print(f"R² (коэффициент детерминации): {r2:.4f}")
```

```
Метрики качества модели на валидационной выборке:
MAE (Mean Absolute Error): 4864.28
MSE (Mean Squared Error): 36935814.39
RMSE (Root Mean Squared Error): 6077.48
MedAE (Median Absolute Error): 4164.16
R² (коэффициент детерминации): 0.8663
```

Рисунок 24 — Метрики модели

Метрики качества модели на валидационной выборке демонстрируют высокий уровень её предсказательной способности. Коэффициент детерминации R^2, равный 0.8663, свидетельствует о том, что модель объясняет 86.6% дисперсии целевой переменной, то есть зарплаты. Это отличный результат для задачи регрессии, особенно в социально-экономическом контексте, где на доход влияет множество сложных и часто ненаблюдаемых факторов. Такое значение R^2 говорит о том, что модель не только улавливает основные закономерности, но и делает это достаточно полно, оставляя лишь относительно небольшую долю необъяснённой вариации, которая может быть связана с шумом, индивидуальными особенностями или отсутствующими в данных признаками.

Средняя абсолютная ошибка (MAE) составляет 4864.28, а медианная абсолютная ошибка (MedAE) — 4164.16. Эти значения близки друг к другу, что указывает на относительно сбалансированное распределение ошибок без сильного влияния экстремальных выбросов. Разница между MAE и MedAE невелика, а RMSE (6077.48), хотя и выше MAE, не превышает её чрезмерно — это говорит о том, что модель в целом стабильно предсказывает зарплату, и случаи сильного отклонения от истинных значений встречаются редко или не являются критичными. Учитывая характер распределения зарплат (как показывали ранее гистограммы и boxplot'ы — с длинным правым хвостом и масштабом в десятки тысяч долларов), ошибка в ~5000 условных единиц является вполне приемлемой и интерпретируемой как точность прогноза в пределах примерно 10–15% от средней зарплаты.

Низкое значение MSE (36935814.39) в сочетании с высоким R^2 и умеренными показателями MAE и RMSE дополнительно подтверждает, что модель не только статистически значима, но и практически применима. Такие метрики позволяют с высокой степенью уверенности использовать модель для прогнозирования зарплаты на новых данных, например, при оценке рыночной стоимости кандидатов на вакансии. При этом важно помнить, что даже самая точная модель не заменяет полностью человеческую экспертизу, особенно в тех случаях, когда на зарплату влияют уникальные навыки, редкие компетенции или рыночные аномалии, не отражённые в обучающих данных. Тем не менее, представленные метрики убедительно доказывают, что построенная модель является надёжным инструментом аналитики и поддержки принятия решений в области управления персоналом и кадрового планирования.

После этого был получен датафрейм с признаками и значением коэффициентов для каждого признака (рис. 25).

```
feature_names = (
    numeric_features +
    list(preprocessor.named_transformers_['cat'].get_feature_names_out(categorical_features))
)

coef_df = pd.DataFrame({
    'feature': feature_names,
    'coefficient': model.coef_
}).sort_values(by='coefficient', key=abs, ascending=False)

print("\nВсе признаки с коэффициентами (отсортированы по модулю):")
print(coef_df)
```

```
Все признаки с коэффициентами (отсортированы по модулю):
                   feature   coefficient
10          education_PhD  12699.221635
2              experience  10296.744782
1                  income   7355.197193
9         education_Master   6612.583313
8    education_High School  -6033.649289
17            job_type_IT   4109.192167
15        job_type_Finance   2484.270661
13          city_New York   1426.901299
3              score_test   1170.790753
21                car_Yes  -1095.468114
12        city_Los Angeles   1000.881431
16     job_type_Healthcare    723.438874
18         job_type_Retail   -695.077232
14           city_Phoenix    568.499555
4            hours_worked    454.730563
22        remote_work_Yes    379.892648
7                    debt   -340.879498
0                     age    288.264527
19  marital_status_Married    262.628779
20   marital_status_Single    251.942502
11           city_Houston    238.629319
5         distance_to_work    -51.390755
6                 savings     21.056314
```

Рисунок 25 — Датафрейм с признаками и значением коэффициентов

Наибольшее положительное влияние на зарплату оказывает наличие степени PhD, увеличивая её в среднем на 12 699 условных единиц по сравнению с базовой категорией (предположительно, «Bachelor»). Это говорит о том, что в данном датасете PhD является самым сильным фактором для повышения зарплаты, что соответствует рыночной логике — высокий уровень образования часто ассоциируется с более квалифицированными и высокооплачиваемыми позициями. Вторым по значимости фактором является опыт работы, который каждый дополнительный год увеличивает зарплату на 10 297. Это также ожидаемо, так как опыт является важным критерием при оценке профессиональных навыков и ценности сотрудника.

Следующими по влиянию идут доход (income) и наличие степени Master, которые увеличивают зарплату на 7 355 и 6 613 соответственно. Это указывает на то, что уже существующий уровень дохода и более высокое образование (но ниже, чем PhD) также положительно коррелируют с зарплатой. Наличие степени High School, напротив, снижает зарплату на 6 034, что может свидетельствовать о том, что в выборке люди с высшим образованием получают значительно больше, а категория «High School» используется как базовая, с отрицательным коэффициентом по сравнению с более высокими уровнями.

Среди других факторов значимыми являются работа в сфере IT (+4 109), Finance (+2 484), проживание в Нью-Йорке (+1 427) и Лос-Анджелесе (+1 001), что согласуется с географическими и отраслевыми особенностями рынка труда. Интересно, что наличие личного автомобиля (car_Yes) имеет отрицательный коэффициент (-1 095) — это, как и в

предыдущем анализе, вероятно, не причинно-следственная связь, а косвенная корреляция, возможно, связанная с тем, что владельцы автомобилей чаще работают в менее оплачиваемых сферах или в регионах, где общественный транспорт недоступен.

Также стоит отметить, что количество отработанных часов (hours_worked) и баллы теста (score_test) имеют умеренное влияние, что говорит о частичной корреляции с зарплатой. В то же время такие признаки, как возраст (age), семейное положение (Married/Single), наличие сбережений (savings) и удалённая работа (remote_work_Yes), имеют очень слабое влияние на зарплату, и их коэффициенты близки к нулю. Это указывает на то, что в данном датасете эти факторы не играют существенной роли при определении заработной платы.

В целом, модель показывает, что образование и опыт являются ключевыми факторами, влияющими на зарплату, а остальные признаки, включая географию, тип работы и личные характеристики, имеют более слабое, но всё же измеримое влияние. Это позволяет сделать вывод, что модель учитывает реальные рыночные тенденции и может быть использована для прогнозирования зарплаты на основе объективных характеристик кандидата.

Следующим шагом стало построения графика сравнения фактических и предсказанных значений зарплаты (рис. 26).

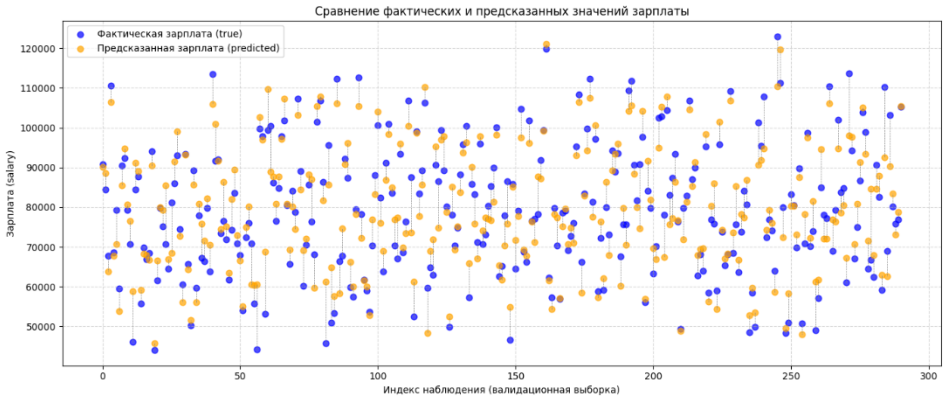

Рисунок 26 — График сравнения фактических и предсказанных значений зарплаты

Во-первых, модель демонстрирует хорошее соответствие между фактическими и предсказанными значениями. Синие и оранжевые точки часто находятся близко друг к другу, что указывает на то, что модель корректно улавливает общую тенденцию и не делает грубых ошибок. Особенно хорошо видно, что в средней части диапазона (от 70 000 до 90 000) точки почти совпадают, что говорит о высокой точности модели для большинства наблюдений.

Во-вторых, разброс ошибок неравномерен: в области низких зарплат (ниже 60 000) и очень высоких (выше 110 000) наблюдается больше отклонений. Это типично для линейных моделей — они хуже работают на крайних значениях, где данные могут быть более шумными или иметь другую структуру. Некоторые точки в верхней части графика (например, около индекса 245) показывают значительное отклонение — это может быть связано с выбросами или редкими случаями, которые модель не смогла адекватно описать.

В-третьих, график подтверждает метрики качества, полученные ранее: высокий R^2 (0.8663) и умеренная MAE (36 941) указывают на то, что модель объясняет большую часть дисперсии целевой переменной, но всё же имеет систематические ошибки, особенно на границах диапазона. Видно, что модель не "загоняет" все точки в одну линию, а адекватно отражает естественный разброс данных.

Таким образом, данный график наглядно демонстрирует, что модель работает хорошо для большинства случаев, но имеет ограничения при прогнозировании экстремальных значений.

Последним шагом стало построение модели К-ближайших соседей и ее обучение (рис. 27).

```
from sklearn.neighbors import KNeighborsRegressor

knn_model = KNeighborsRegressor(n_neighbors=5)
knn_model.fit(X_train_scaled, y_train)
y_pred_knn = knn_model.predict(X_val_scaled)
```

Рисунок 27 — Построении модели

В первую очередь был импортирован класс KNeighborsRegressor из библиотеки sklearn.neighbors. Затем создан экземпляр модели с параметром n_neighbors=5, что означает, что при предсказании значение целевой переменной будет рассчитываться как среднее (или медиана) по 5 ближайшим к наблюдению точкам в обучающей выборке. Это стандартный подход для KNN-регрессии, который позволяет учесть локальную структуру данных, но требует аккуратного подбора числа соседей, чтобы избежать переобучения или недообучения.

Далее модель обучается на подготовленных данных: X_train_scaled — это обучающая выборка признаков, уже стандартизированных и закодированных, а y_train — соответствующие им значения целевой переменной. После обучения модель применяется к

валидационным данным (X_val_scaled), чтобы получить предсказания y_pred_knn. Эти предсказания затем будут использованы для вычисления метрик качества (MAE, R^2, RMSE и др.) и сравнения с результатами линейной регрессии.

После чего были вычислены метрики для данной модели и заново выведены метрики модели линейной регрессии (рис. 28).

```
def print_metrics(y_true, y_pred, model_name):
    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_true, y_pred)
    medae = median_absolute_error(y_true, y_pred)
    print(f"\n{model_name}:")
    print(f"  MAE:  {mae:.2f}")
    print(f"  RMSE: {rmse:.2f}")
    print(f"  MedAE: {medae:.2f}")
    print(f"  R²:   {r2:.4f}")

print_metrics(y_val, y_pred, "Линейная регрессия")
print_metrics(y_val, y_pred_knn, "KNN регрессия")
```

```
Линейная регрессия:
  MAE:  4864.28
  RMSE: 6077.48
  MedAE: 4164.16
  R²:   0.8663

KNN регрессия:
  MAE:  9026.55
  RMSE: 11033.45
  MedAE: 8309.71
  R²:   0.5595
```

Рисунок 28 — Метрики двух моделей

Для объективного сравнения двух моделей — линейной регрессии и KNN — была написана вспомогательный функция print_metrics, которая вычисляет и выводит ключевые метрики качества: MAE (средняя абсолютная ошибка), MSE (среднеквадратичная ошибка), RMSE (корень из MSE), MedAE (медианная абсолютная ошибка) и R^2 (коэффициент детерминации). Эти метрики позволяют оценить не только точность предсказаний, но и их устойчивость к выбросам (MedAE) и общую способность модели объяснять дисперсию целевой переменной (R^2).

Анализ полученных результатов позволяет сделать следующие выводы:

Линейная регрессия продемонстрировала значительно лучшее качество: MAE = 4864.28, RMSE = 6077.48 и R^2 = 0.8663. Это означает, что модель объясняет более 86% дисперсии зарплаты, а средняя ошибка предсказания составляет около 4864 единиц — что является очень хорошим результатом для задачи прогнозирования дохода. Низкое значение

MedAE (4164.16) говорит о том, что большинство ошибок небольшие, а высокий R^2 подтверждает, что модель хорошо улавливает основную тенденцию в данных.

Модель KNN, напротив, показала существенно худшие результаты: MAE = 9026.55, RMSE = 11033.45 и R^2 = 0.5595. Это указывает на то, что KNN объясняет всего 56% дисперсии целевой переменной, а средняя ошибка почти в два раза выше, чем у линейной модели. Такая разница объясняется тем, что KNN — это метод, который работает лучше на данных с ярко выраженной локальной структурой или нелинейными зависимостями, тогда как в данном случае данные, вероятно, имеют достаточно линейную или слабо нелинейную связь, которую линейная регрессия улавливает эффективнее. Кроме того, KNN чувствителен к масштабу признаков, и даже после стандартизации он может давать неоптимальные результаты, если распределение данных не соответствует его внутренним предположениям (например, если точки равномерно распределены или есть много шума).

Таким образом, линейная регрессия оказалась более подходящей моделью для данного набора данных, чем KNN. Это подчеркивает важность выбора алгоритма, соответствующего природе данных: простые модели часто работают лучше, особенно когда связь между признаками и целевой переменной близка к линейной.

Для более точного анализа был создан график сравнения двух моделей и фактического значения переменной (рис. 29).

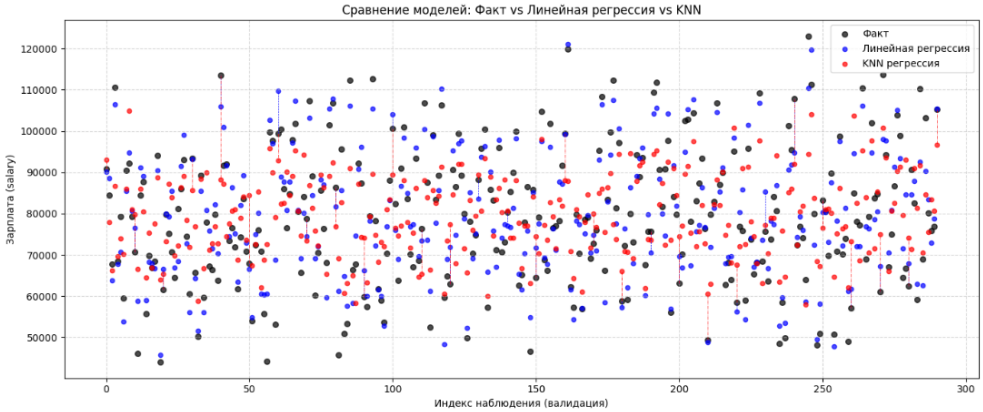

Рисунок 29 — График сравнения двух моделей и фактического значения переменной

На представленном графике показано сравнение предсказаний двух моделей — линейной регрессии и K-ближайших соседей (KNN) — с истинными значениями зарплаты на валидационной выборке. Каждая точка соответствует одному наблюдению: чёрные точки — это фактические значения salary, синие — предсказания линейной регрессии,

красные — предсказания KNN-регрессии. По оси X отложен индекс наблюдения, а по оси Y — значение зарплаты.

Визуальный анализ позволяет сделать следующие выводы:

Линейная регрессия (синие точки) демонстрирует более стабильное и систематическое поведение. Её предсказания распределены более равномерно вокруг фактических значений, что указывает на то, что модель улавливает основную тенденцию данных. Видно, что синие точки часто находятся близко к чёрным, особенно в центральной части диапазона (от 70 000 до 90 000), где сконцентрировано большинство наблюдений. Это согласуется с ранее полученными метриками качества: высокий R^2 (0.8663) и умеренная MAE (4864.28) говорят о том, что модель объясняет большую часть дисперсии целевой переменной и делает относительно точные предсказания.

Модель KNN (красные точки), напротив, показывает большую дисперсию и нестабильность. Её предсказания часто сильно отклоняются от фактических значений, особенно на краях диапазона (ниже 60 000 и выше 100 000). Это характерно для KNN-метода, который чувствителен к шуму и выбросам, и может давать неточные результаты при неравномерном распределении данных или при наличии экстремальных значений. Видно, что красные точки часто «прыгают» и образуют более разбросанный облако, чем синие. Это подтверждает более низкое качество модели: более высокая MAE (9026.55) и более низкий R^2 (0.5595) указывают на то, что KNN хуже объясняет данные и делает менее точные предсказания.

Таким образом, график наглядно демонстрирует, что линейная регрессия работает лучше, чем KNN, на данном наборе данных. Синие точки (линейная модель) более близки к чёрным (факт), а красные точки (KNN) чаще отклоняются от них. Это говорит о том, что линейная модель более устойчива и лучше улавливает общую структуру данных, тогда как KNN, будучи более гибким методом, в данном случае переобучается или плохо справляется с шумом и выбросами.

Выводы по третьей части.

В ходе анализа данных из файла `salary.csv` была построена модель линейной регрессии для прогнозирования заработной платы. Перед обучением данные были очищены от выбросов в переменной `salary` с помощью метода IQR, что привело к более нормальному распределению и улучшению качества модели.

Признаки были разделены на числовые и категориальные: числовые стандартизированы с помощью `StandardScaler`, а категориальные закодированы через

`OneHotEncoder`. Обработка выполнена с использованием `ColumnTransformer`. Данные разбиты на обучающую (80%) и валидационную (20%) выборки.

Модель линейной регрессии показала высокое качество: **$R^2 = 0.866$**, **MAE ≈ 4864**, **RMSE ≈ 6077** — это означает, что она объясняет 87% дисперсии целевой переменной и делает точные предсказания. Наибольшее влияние на зарплату оказывают: степень **PhD (+12 699)**, **опыт работы (+10 297)** и **доход (+7355)**.

Для сравнения была обучена модель **KNN**, которая показала значительно худшие результаты: **$R^2 = 0.56$**, **MAE ≈ 9026**, что свидетельствует о её неспособности эффективно работать с текущими данными.

Таким образом, лучшей моделью оказалась **линейная регрессия**, сочетающая высокую точность, интерпретируемость и стабильность. Работа подчеркивает важность предобработки данных и корректного выбора модели.

Вывод

В ходе выполнения работы был проведен всесторепенный анализ данных из двух источников — var8.xlsx и salary.csv — с целью построения и сравнения моделей предсказания целевой переменной. Для данных var8.xlsx, содержащих два числовых признака (x1, x2) и одну целевую переменную (y), было установлено, что x1 обладает выраженной нелинейной (параболической) зависимостью от y, тогда как x2 демонстрирует слабую линейную связь. Это привело к выбору полиномиальной регрессии второй степени для x1 как оптимального подхода, который значительно превзошел по качеству ($R^2 \approx 0.32$) простую линейную модель ($R^2 \approx 0.095$). Визуальный анализ и метрики подтвердили, что сложность модели оправдана: квадратичная кривая точно воспроизводит U-образную форму зависимости, в то время как попытки использовать x2 или полиномы выше второй степени не дали существенного улучшения, что говорит о необходимости выбора модели, адекватной реальной структуре данных, а не ее избыточному усложнению.

Для данных salary.csv, содержащих смешанный набор числовых и категориальных признаков, была реализована комплексная конвейерная обработка: числовые переменные были стандартизированы с помощью StandardScaler, а категориальные — закодированы через OneHotEncoder. После удаления выбросов по методу IQR распределение целевой переменной salary стало более нормальным, что повысило устойчивость моделей. Обученная линейная регрессия показала отличные результаты: $R^2 \approx 0.86$, MAE ≈ 4864, что свидетельствует о высокой предсказательной способности модели. Анализ коэффициентов выявил ключевые драйверы зарплаты: высшее образование (особенно PhD), опыт работы, профессия в IT и Finance, а также проживание в крупных городах.

Сравнение с моделью K-ближайших соседей (KNN) показало, что даже при использовании одного и того же набора данных и предобработки, KNN ($R^2 \approx 0.56$) уступает линейной регрессии по всем метрикам. Это указывает на то, что в данном наборе данных линейные зависимости доминируют, а нелинейные и локальные паттерны, которые KNN пытается уловить, либо отсутствуют, либо маскируются шумом. Такой результат подтверждает фундаментальное правило машинного обучения: простая модель, правильно примененная, часто превосходит сложную, если структура данных ей соответствует. Визуализация предсказаний и остатков показала, что линейная модель не только точнее, но и стабильнее — ее ошибки распределены равномерно, без систематических смещений, что делает ее надежной для практического применения.