

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ
НА ТЕМУ:

Студент	<u>ИУ5-63Б</u>	<u>(подпись, дата)</u>	<u>С.В. Лупарев</u>
	(группа)		(И.О. Фамилия)
Руководитель НИР		<u>(подпись, дата)</u>	<u>Ю.Е. Гапанюк</u>
			(И.О. Фамилия)

2025 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой

ИУ5

(индекс)

В.И. Терехов

(И.О. Фамилия)

(подпись)

(дата)

ЗАДАНИЕ
на выполнение научно-исследовательской работы

по теме Обработка набора данных

Студент группы ИУ5-63Б

Беспалова Виктория Андреевна

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

ИССЛЕДОВАТЕЛЬСКАЯ

Источник тематики (кафедра, предприятие, НИР) КАФЕДРА

График выполнения НИР:

25% к _____ нед., 50% к _____ нед., 75% к _____ нед., 75% к _____ нед

Техническое задание: решение задачи машинного обучения на основе материалов

дисциплины. Выбор датасета, первичный анализ, выбор метрик для оценки качества моделей, построение базового решения, оценка качества, подбор гиперпараметров.

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на _____ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

Дата выдачи задания «07» февраля 2025 г.

Руководитель НИР

(подпись, дата)

Ю.Е. Гапанюк

(И.О. Фамилия)

Студент

(подпись, дата)

С.В. Лупарев

(И.О. Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

СОДЕРЖАНИЕ

<i>ВВЕДЕНИЕ.....</i>	<i>4</i>
<i>1 ПОСТАНОВКА ЗАДАЧИ.....</i>	<i>5</i>
<i>2 АНАЛИЗ ДАННЫХ.....</i>	<i>7</i>
<i>3 ВЫБОР МОДЕЛЕЙ И МЕТРИК ДЛЯ ОЦЕНКИ КАЧЕСТВА</i>	<i>20</i>
<i>4 ФОРМИРОВАНИЕ ОБУЧАЮЩЕЙ И ТЕСТОВОЙ ВЫБОРОК.....</i>	<i>22</i>
<i>5 ПОСТРОЕНИЕ БАЗОВОГО РЕШЕНИЯ</i>	<i>23</i>
<i>6 ПОДБОР ГИПЕРПАРАМЕТРОВ.....</i>	<i>25</i>
<i>7 СРАВНЕНИЕ РЕШЕНИЙ</i>	<i>28</i>
<i>8 ВЕБ-ПРИЛОЖЕНИЕ</i>	<i>29</i>
<i>ЗАКЛЮЧЕНИЕ</i>	<i>34</i>
<i>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....</i>	<i>35</i>

ВВЕДЕНИЕ

В глобальном мире остро стоит вопрос зарплаты, поскольку всем людям без исключения хочется содержать себя и свою семью, поэтому знание параметров, которые влияют на зарплату является острой необходимостью для многих.

Целью данного исследования является построение и оценка моделей машинного обучения для классификации людей по их зарплате на тех, кто получает хотя бы 50 000 долларов в год, и на людей, которые получают меньше этой цифры. Исследование основано на анализе выборки данных, включающей различные метрики, оценивающие человека с разных сторон. Важным аспектом работы является применение методов предварительной обработки данных, таких как очистка данных, обработка пропущенных значений и их масштабирование, что позволяет повысить точность и эффективность последующей классификации.

В рамках введения в аналитику мы проводим разведочный анализ данных для идентификации основных статистических характеристик выборки, выявления аномалий и оценки корреляционных связей между переменными. Дальнейший этап включает в себя тщательный отбор и настройку гиперпараметров алгоритмов машинного обучения.

Ожидается, что результаты данной работы окажут значительное влияние на представление людей о их будущих перспективах, предоставят ценную информацию, позволяющую в текущем времени повлиять на уровень будущей зарплаты.

1 ПОСТАНОВКА ЗАДАЧИ

Целью исследования является построение и оценка моделей машинного обучения для прогнозирования зарплаты человека (больше или меньше 50 000 долларов в год) на основании признаков, представленных в Таблице 1. Целевой переменной в решаемой задаче бинарной классификации является больше ли зарплата, чем 50 000 долларов в год.

Таблица 1 – Описание полей датасета

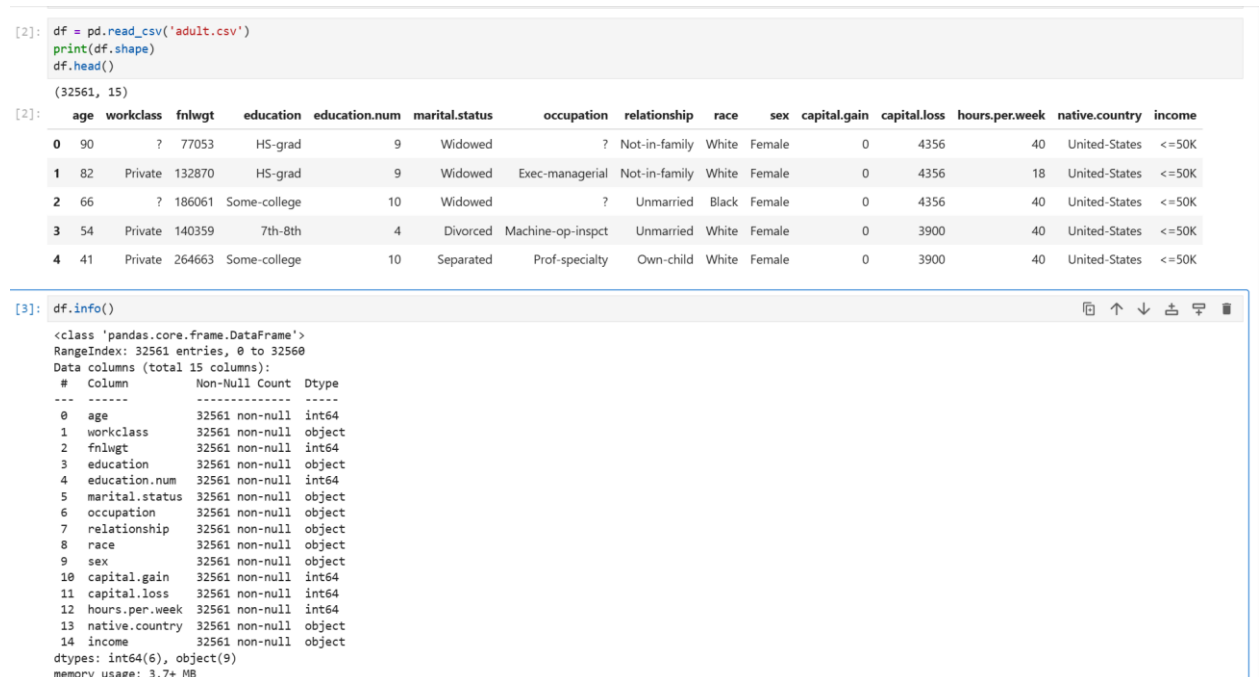
Поле	Описание	Тип данных
age	Возраст человека	object
workclass	Тип занятости (например, Private, Self-emp-not-inc, Local-gov, ?)	object
fnlwgt	"Final weight"; оценка того, сколько людей в популяции представляет данная запись (создается Бюро переписи)	int64
education	Уровень образования в текстовом виде (например, Bachelors, HS-grad, 11th).	object
education-num	Уровень образования в числовом, упорядоченном виде (соответствует education)	int64
marital-status	Семейное положение (например, Married-civ-spouse, Never-married, Divorced).	object
relationship	Родственная связь в семье (например, Husband, Not-in-family, Own-child, Wife).	object
race	Расовая принадлежность (например, White, Black, Asian-Pac-Islander).	object
sex	Пол (Male, Female).	object
capital-gain	Прирост капитала (доход от инвестиций и т.п., не связанный с зарплатой).	int64

Поле	Описание	Тип данных
capital-loss	Убыток капитала.	int64
hours-per-week	Количество рабочих часов в неделю.	int64
native-country	Родная страна (например, United-States, Mexico, Philippines, ?).	object
occupation	Профессиональная сфера деятельности (например, Tech-support, Craft-repair, Other-service, ?).	object
income	Целевая переменная: уровень годового дохода ($\leq 50K$ или $> 50K$).	object

2 АНАЛИЗ ДАННЫХ

Основные характеристики датасета

Основные характеристики датасета, такие как: количество строк и столбцов, типы данных полей, уникальные значения целевого признака – приведены на Рисунке 1. Целевой признак после будущего преобразования является бинарным и содержит только значения 0 и 1. Значит, решается задача бинарной классификации.



```
[2]: df = pd.read_csv('adult.csv')
print(df.shape)
df.head()
```

(32561, 15)

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.country	income
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356	40	United-States	<=50K
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	United-States	<=50K
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356	40	United-States	<=50K
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	40	United-States	<=50K
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	United-States	<=50K

```
[3]: df.info()
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
Column Non-Null Count Dtype

0 age 32561 non-null int64
1 workclass 32561 non-null object
2 fnlwgt 32561 non-null int64
3 education 32561 non-null object
4 education.num 32561 non-null int64
5 marital.status 32561 non-null object
6 occupation 32561 non-null object
7 relationship 32561 non-null object
8 race 32561 non-null object
9 sex 32561 non-null object
10 capital.gain 32561 non-null int64
11 capital.loss 32561 non-null int64
12 hours.per.week 32561 non-null int64
13 native.country 32561 non-null object
14 income 32561 non-null object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB

Рисунок 1. Основные характеристики датасета

В анализируемом наборе данных есть пропуски в колонках «workclass», «occupation» и «native.country» (см. Рисунок 2). Можно отметить, что количество пропусков данных не превышает 6%. Было решено заменить пропуски на самые частые встречающиеся значения (колонки с пропусками являются категориальными).

Пропуски в данном датасете отображаются как '?' => заменим '?' на Nan и проверим еще раз кол-во пропусков

```
[7]: for col in df.columns:
      unique_vals = df[col].unique()
      if '?' in unique_vals or '?' in unique_vals or '?' in unique_vals:
          df[col] = df[col].replace('?', np.nan)
          df[col] = df[col].replace('?', np.nan)
          df[col] = df[col].replace('?', np.nan)

df.isnull().sum()

[7]: age                0
     workclass         1836
     fnlwgt            0
     education         0
     education.num     0
     marital.status    0
     occupation        1843
     relationship      0
     race              0
     sex               0
     capital.gain      0
     capital.loss      0
     hours.per.week    0
     native.country    583
     income            0
     dtype: int64
```

Рисунок 2. Количество пропусков данных по столбцам

На Рисунке 3 представлены основные характеристики числовых столбцов датасета.

```
[4]: df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	32561.0	38.581647	13.640433	17.0	28.0	37.0	48.0	90.0
fnlwgt	32561.0	189778.366512	105549.977697	12285.0	117827.0	178356.0	237051.0	1484705.0
education.num	32561.0	10.080679	2.572720	1.0	9.0	10.0	12.0	16.0
capital.gain	32561.0	1077.648844	7385.292085	0.0	0.0	0.0	0.0	99999.0
capital.loss	32561.0	87.303830	402.960219	0.0	0.0	0.0	0.0	4356.0
hours.per.week	32561.0	40.437456	12.347429	1.0	40.0	40.0	45.0	99.0

Рисунок 3. Основные характеристики числовых столбцов датасета

Баланс классов

Из описания набора данных мы знаем, что присутствует дисбаланс классов, в наборе данных 76% принадлежат к классу « $\leq 50K$ » и 24% – к классу « $> 50K$ ».

Дисбаланс действительно ожидаем в данных о зарплате в реальном мире, поскольку мой личный опыт и многочисленные исследования показывают в среднем низкий уровень зарплат.

Визуальный анализ датасета

Распределение значений целевой переменной проиллюстрировано на Рисунке 4. Распределение значений числовых переменных представлено на Рисунке 5. Распределение значений категориальных переменных представлено на Рисунке 6. Стоит отметить, что распределения значений числовой переменной близки к нормальному.

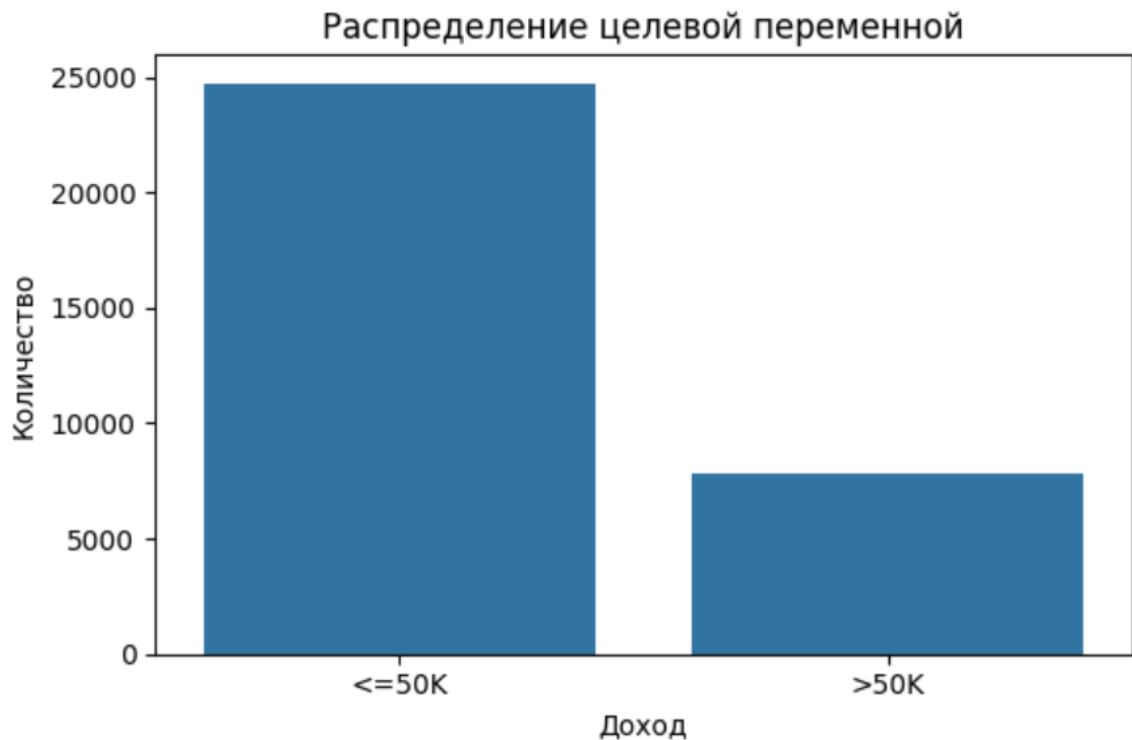


Рисунок 4. Распределение значений целевой переменной

Гистограммы числовых признаков

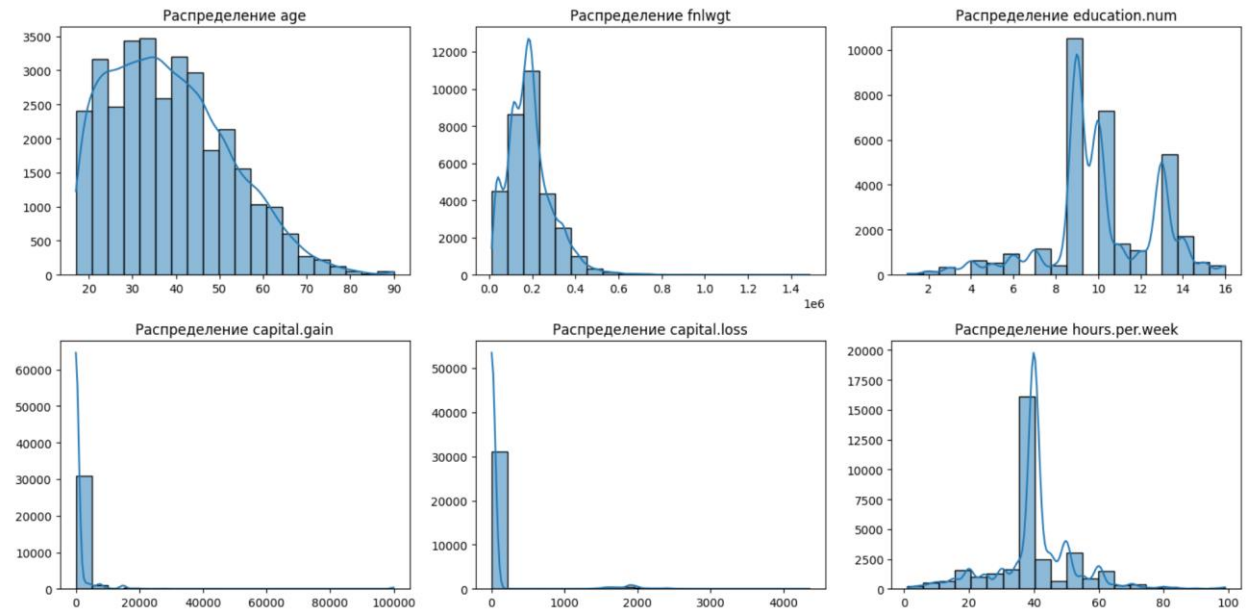


Рисунок 5. Распределение значений числовых переменных

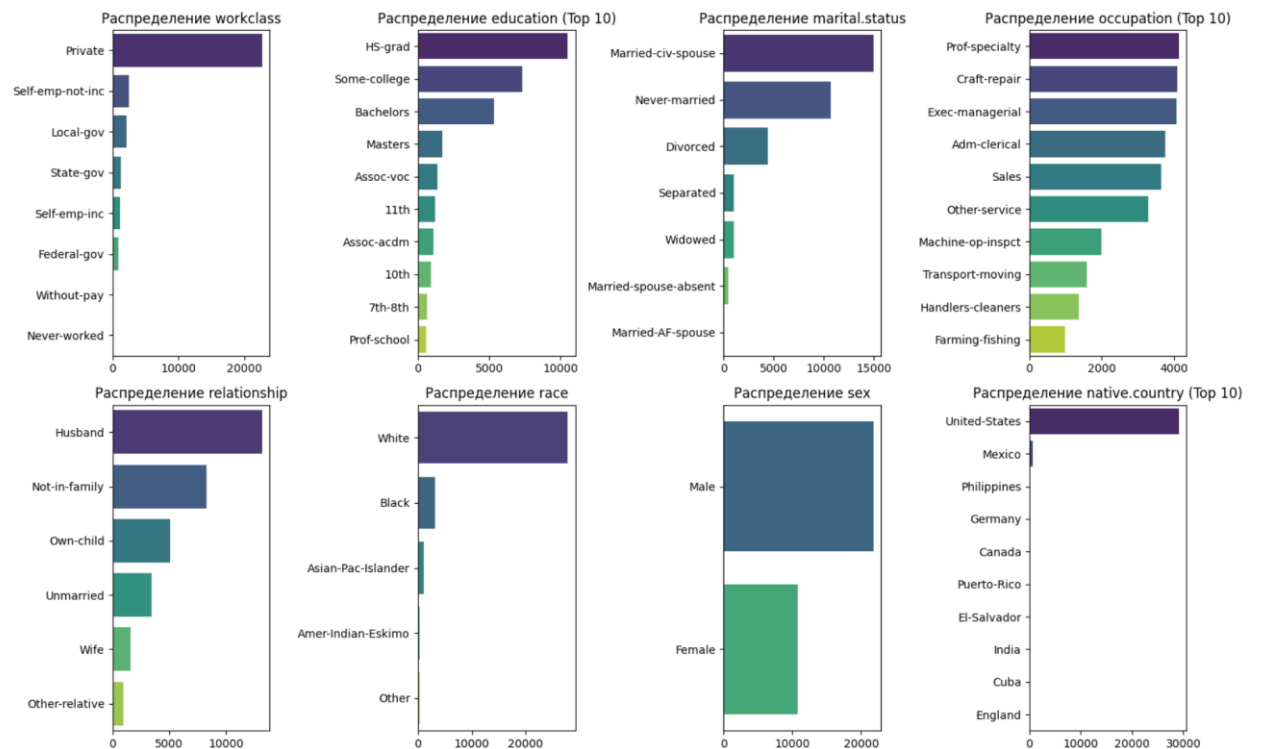


Рисунок 6. Распределение значений категориальных переменных

Заполнение пропусков

Заполнение пропусков проводится с заполнением самого частого значения, поскольку как было отмечено ранее, количество пропусков не превышает 6% и пропуски содержатся только в категориальных переменных. Код заполнения пропусков представлен на Рисунке 7.

```
# 1. Заполнение пропусков
cols_to_fill = ['workclass', 'occupation', 'native.country']
for col in cols_to_fill:
    mode = df[col].mode()[0]
    df[col].fillna(mode, inplace=True)
    print(f"Пропуски в '{col}' заполнены модой: '{mode}'")
```

Пропуски в 'workclass' заполнены модой: 'Private'

Пропуски в 'occupation' заполнены модой: 'Prof-specialty'

Пропуски в 'native.country' заполнены модой: 'United-States'

Рисунок 3. Код заполнения пропусков

Преобразования категориальных признаков

Поскольку большинство признаков являются категориальными и содержат множество значений, было решено произвести WoE-трансформацию для всех категориальных признаков (см. Рисунок 9), кроме целевой переменной, для ее преобразования был использован LabelEncoder (см. Рисунок 8). После преобразования классу «<=50К» соответствует значение 0, а классу «>50К» - значение 1.

Также был удален признак «education», так как признак «education.num» уже соответствует уровню образования человека, но уже является числовым (также см. Рисунок 8).

```
# Подготовка признаков
from sklearn.preprocessing import StandardScaler, LabelEncoder
# 2. Выбор признаков (удаление 'education' в пользу 'education.num')
df = df.drop('education', axis=1)

# 3. Кодирование целевой переменной 'income'
le = LabelEncoder()
df['income'] = le.fit_transform(df['income'])
print(f"\nЦелевая переменная 'income' закодирована. Классы: {le.classes_} -> {le.transform(le.classes_)}")
```

Целевая переменная 'income' закодирована. Классы: ['<=50K' '>50K'] -> [0 1]

Рисунок 8. Кодирование целевой переменной

```

# 4. Расчет WoE и трансформация категориальных признаков
woe_dfs = []
woe_mappings = {}

def calculate_woe(df, feature_name, target_name):
    # Константа для сглаживания
    smoothing = 0.5
    df_temp = pd.DataFrame({'feature': df[feature_name], 'target': df[target_name]})

    # Количество событий (target=1) и не-событий (target=0)
    total_events = df_temp['target'].sum()
    total_non_events = len(df_temp) - total_events

    # Группируем по значениям признака
    grouped = df_temp.groupby('feature')['target'].agg(['sum', 'count'])
    grouped.rename(columns={'sum': 'events', 'count': 'total_in_category'}, inplace=True)
    grouped['non_events'] = grouped['total_in_category'] - grouped['events']

    # Применяем сглаживание
    grouped['events'] = grouped['events'] + smoothing
    grouped['non_events'] = grouped['non_events'] + smoothing

    dist_events = grouped['events'] / (total_events + smoothing * len(grouped)) # len(grouped) - число категорий
    dist_non_events = grouped['non_events'] / (total_non_events + smoothing * len(grouped))

    # На всякий случай
    dist_events = np.where(dist_events == 0, 0.00001, dist_events)
    dist_non_events = np.where(dist_non_events == 0, 0.00001, dist_non_events)

    grouped['woe'] = np.log(dist_non_events / dist_events)
    return grouped['woe'].to_dict()

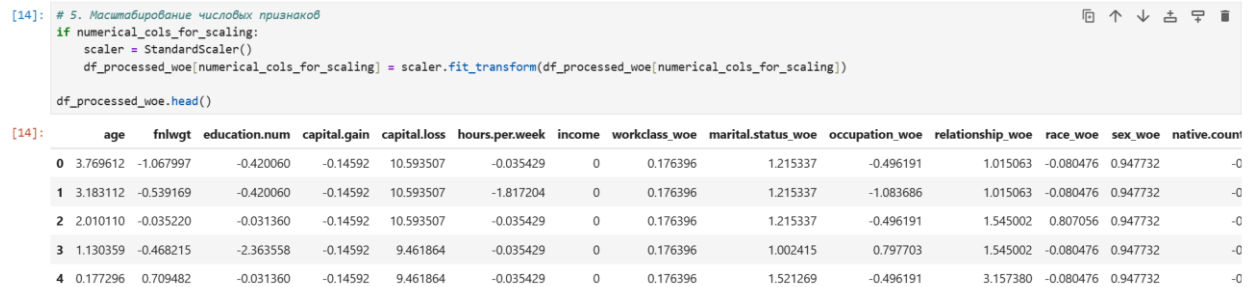
df_processed_woe = df.copy()
for col in categorical_cols_for_woe:
    woe_map = calculate_woe(df_processed_woe, col, 'income')
    woe_mappings[col] = woe_map
    df_processed_woe[col + '_woe'] = df_processed_woe[col].map(woe_map)
    df_processed_woe = df_processed_woe.drop(col, axis=1)
    print(f"Столбец {col} заменен на {col}_woe.")

```

Рисунок 9. Код WoE-трансформации

Масштабирование данных

Для числовых данных было произведено масштабирование (см. Рисунок 10).



The screenshot shows a Jupyter Notebook cell with the following code:

```
[14]: # 5. Масштабирование числовых признаков
if numerical_cols_for_scaling:
    scaler = StandardScaler()
    df_processed_woe[numerical_cols_for_scaling] = scaler.fit_transform(df_processed_woe[numerical_cols_for_scaling])
df_processed_woe.head()
```

The output of the code is a table showing the first five rows of the scaled data:

	age	fnlwtg	education.num	capital.gain	capital.loss	hours.per.week	income	workclass_woe	marital.status_woe	occupation_woe	relationship_woe	race_woe	sex_woe	native.count
0	3.769612	-1.067997	-0.420060	-0.14592	10.593507	-0.035429	0	0.176396	1.215337	-0.496191	1.015063	-0.080476	0.947732	-0
1	3.183112	-0.539169	-0.420060	-0.14592	10.593507	-1.817204	0	0.176396	1.215337	-1.083686	1.015063	-0.080476	0.947732	-0
2	2.010110	-0.035220	-0.031360	-0.14592	10.593507	-0.035429	0	0.176396	1.215337	-0.496191	1.545002	0.807056	0.947732	-0
3	1.130359	-0.468215	-2.363558	-0.14592	9.461864	-0.035429	0	0.176396	1.002415	0.797703	1.545002	-0.080476	0.947732	-0
4	0.177296	0.709482	-0.031360	-0.14592	9.461864	-0.035429	0	0.176396	1.521269	-0.496191	3.157380	-0.080476	0.947732	-0

Рисунок 10. Масштабирования числовых признаков

Однофакторный анализ

Был произведен однофакторный анализ между всеми признаками и целевой переменной. В качестве метрики отбора был выбран коэффициент Somers'D, при значениях модуля которого $< 0,2$ признак признавался недостаточно информативным и отбрасывался, и не был использован в качестве финального признака (см. Рисунок 11, Рисунок 12).

```
# 6. Расчет Somers' D
from sklearn.metrics import roc_auc_score
target_col_name = 'income'
features_for_somers_d_woe = [col for col in df_processed_woe.columns if col != target_col_name]
somers_d_results_woe = {}

for feature in features_for_somers_d_woe:
    auc = roc_auc_score(df_processed_woe[target_col_name], df_processed_woe[feature])
    somers_d = 2 * (auc - 0.5)
    somers_d_results_woe[feature] = somers_d

somers_d_df_woe = pd.DataFrame(list(somers_d_results_woe.items()), columns=['Признак', 'Somers' D'])
somers_d_df_woe.dropna(subset=["Somers' D"], inplace=True)
somers_d_df_woe['Abs Somers\'' D'] = somers_d_df_woe["Somers' D"].abs()
somers_d_df_woe = somers_d_df_woe.sort_values(by='Abs Somers\'' D', ascending=False).reset_index(drop=True)

print("Somers' D для признаков (с WoE) по отношению к 'income':")
pd.set_option('display.max_rows', None)
print(somers_d_df_woe[['Признак', 'Somers' D']])
pd.reset_option('display.max_rows')

# 7. Отбор признаков по Somers' D > 0.2
threshold_somers_d = 0.2
selected_features_somers_d = somers_d_df_woe[somers_d_df_woe['Abs Somers\'' D'] > threshold_somers_d]['Признак'].tolist()

print(f"Признаки, отобранные по |Somers' D| > 0.2")
print(f"Количество отобранных признаков: {len(selected_features_somers_d)}")
print("Список отобранных признаков:")
for f in selected_features_somers_d:
    print(f"- {f} (Somers' D: {somers_d_df_woe[somers_d_df_woe['Признак'] == f]['Somers\'' D'].values[0]:.4f})")
```

Рисунок 11. Расчет Somers'D


```

Somers' D для признаков (с WoE) по отношению к 'income':
    Признак    Somers' D
0  relationship_woe  -0.558728
1  marital.status_woe  -0.539146
2  education.num    0.433993
3  occupation_woe   -0.414669
4  age              0.368486
5  hours.per.week   0.344075
6  sex_woe          -0.237662
7  capital.gain     0.179986
8  workclass_woe    -0.141996
9  race_woe         -0.076308
10 capital.loss     0.069592
11 native.country_woe  -0.059207
12 fnlwgt          -0.014499
Признаки, отобранные по |Somers' D| > 0.2
Количество отобранных признаков: 7
Список отобранных признаков:
- relationship_woe (Somers' D: -0.5587)
- marital.status_woe (Somers' D: -0.5391)
- education.num (Somers' D: 0.4340)
- occupation_woe (Somers' D: -0.4147)
- age (Somers' D: 0.3685)
- hours.per.week (Somers' D: 0.3441)
- sex_woe (Somers' D: -0.2377)

|: # Создаем итоговый DataFrame
df_final_selected = df_processed_woe[selected_features_somers_d + [target_col_name]].copy()
print(df_final_selected.shape)
df_final_selected.head()

(32561, 8)

|:
   relationship_woe  marital.status_woe  education.num  occupation_woe    age  hours.per.week  sex_woe  income
0         1.015063         1.215337        -0.420060        -0.496191  3.769612        -0.035429  0.947732      0
1         1.015063         1.215337        -0.420060        -1.083686  3.183112        -1.817204  0.947732      0
2         1.545002         1.215337        -0.031360        -0.496191  2.010110        -0.035429  0.947732      0
3         1.545002         1.002415        -2.363558         0.797703  1.130359        -0.035429  0.947732      0
4         3.157380         1.521269        -0.031360        -0.496191  0.177296        -0.035429  0.947732      0

```

Рисунок 12. Отбор предфинального набора признаков

Многофакторный анализ

Для оставшихся признаков был подсчитан коэффициент корреляции Пирсона и была построена heat-map (см. Рисунок 13). Оказалось, что между признаками «relationship_woe» и «marital.status_woe» существует сильная корреляция, поэтому было принято решение исключить признак «marital.status_woe» из списка финальных признаков (см. Рисунок 14).

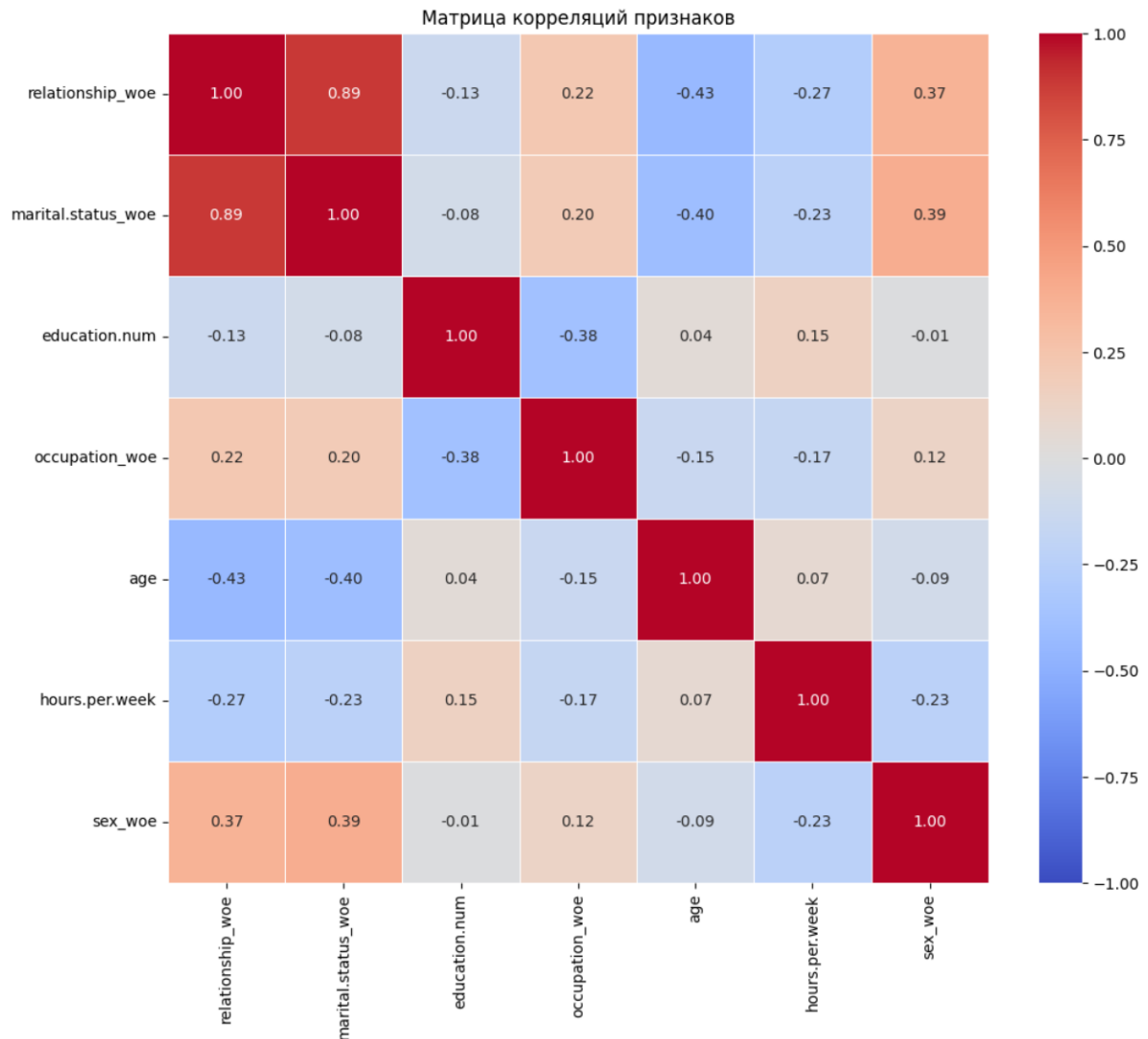


Рисунок 13. Корреляционная матрица

```
df_final_selected = df_final_selected.drop('marital.status_woe', axis=1)  
df = df_final_selected.copy()
```

Рисунок 14. Удаление последнего лишнего признака.

3 ВЫБОР МОДЕЛЕЙ И МЕТРИК ДЛЯ ОЦЕНКИ КАЧЕСТВА

Выбор метрик для оценки качества моделей

1) *Accuracy*

Метрика вычисляет процент (долю в диапазоне от 0 до 1) правильно определенных классов.

2) *ROC AUC*

Основана на вычислении следующих характеристик:

True Positive Rate, откладывается по оси ординат. Совпадает с recall.

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate, откладывается по оси абсцисс. Показывает какую долю из объектов отрицательного класса алгоритм предсказал неверно.

$$FPR = \frac{FP}{FP + TP}$$

TPR содержит в знаменателе количество истинных 1.

FPR содержит в знаменателе количество истинных 0.

Идеальная ROC-кривая проходит через точки (0,0)-(0,1)-(1,1), то есть через верхний левый угол графика. Чем сильнее отклоняется кривая от верхнего левого угла графика, тем хуже качество классификации.

В качестве количественной метрики используется площадь под кривой - ROC AUC (Area Under the Receiver Operating Characteristic Curve). Чем ниже проходит кривая тем меньше ее площадь и тем хуже качество классификатора.

3) *Recall*

Доля верно предсказанных классификатором положительных объектов, из всех объектов, которые классификатор верно или неверно определил как положительные.

4) *F1-мера*

Объединяет precision и recall в единую метрику.

Precision – доля верно предсказанных классификатором положительных объектов, из всех объектов, которые классификатор верно или неверно определил как положительные.

Recall – доля верно предсказанных классификатором положительных объектов, из всех объектов, которые классификатор верно или неверно определил как положительные.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

В качестве основной метрики оценки качества была выбрана метрика ROC AUC.

Выбор моделей для решения задачи бинарной классификации

С учетом следующих условий:

- Требование задания НИРС: не менее 5 моделей, не менее 2 ансамблевых моделей;
- Решение задачи бинарной классификации;
- Датасет после всех преобразований содержит только числовые признаки;

выберем модели:

- Логистическая регрессия;
- Метод опорных векторов;
- Алгоритм KNN для классификации;
- Случайный лес;
- XGBoost(один из представителей GBM).

4 ФОРМИРОВАНИЕ ОБУЧАЮЩЕЙ И ТЕСТОВОЙ ВЫБОРОК

Разделим выборку на обучающую и тестовую с помощью `train_test_split`, учтем дисбаланс классов (см. Рисунок 15).

```
df_final_selected = df_final_selected.drop('marital.status_woe', axis=1)
df = df_final_selected.copy()
from sklearn.model_selection import train_test_split

X = df.drop('income', axis=1)
y = df['income']

# Разделение данных на обучающую и тестовую выборки(сохраняем баланс классов)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=52, stratify=y)

print(f"\nРазмер обучающей выборки (X_train): {X_train.shape}")
print(f"Размер тестовой выборки (X_test): {X_test.shape}")
print(f"Распределение 'income' в y_train: \n{y_train.value_counts(normalize=True)}")
print(f"Распределение 'income' в y_test: \n{y_test.value_counts(normalize=True)}")
```

```
Размер обучающей выборки (X_train): (22792, 6)
Размер тестовой выборки (X_test): (9769, 6)
Распределение 'income' в y_train:
income
0    0.75917
1    0.24083
Name: proportion, dtype: float64
Распределение 'income' в y_test:
income
0    0.759238
1    0.240762
Name: proportion, dtype: float64
```

Рисунок 15. Разделение выборки

5 ПОСТРОЕНИЕ БАЗОВОГО РЕШЕНИЯ

Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров. Производится обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки.

Обучение моделей

Обучим 5 выбранных моделей без подбора гиперпараметров (см. Рисунок 16). Используем модели из библиотек scikit-learn и xgboost.

```
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score, f1_score, recall_score, accuracy_score, confusion_matrix

# Инициализация моделей с параметрами по умолчанию
models = {
    "Logistic Regression": LogisticRegression(random_state=52, max_iter=1000, n_jobs=-1),
    "K-Nearest Neighbors": KNeighborsClassifier(n_jobs=-1),
    "Support Vector Machine": SVC(random_state=52, probability=True),
    "Random Forest": RandomForestClassifier(random_state=52, n_jobs=-1),
    "XGBoost": XGBClassifier(random_state=52, use_label_encoder=False, eval_metric='logloss', n_jobs=-1)
}

baseline_results = []

for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred_proba = model.predict_proba(X_test)[: , 1]
    y_pred = model.predict(X_test)

    roc_auc = roc_auc_score(y_test, y_pred_proba)
    f1 = f1_score(y_test, y_pred, zero_division=0)
    recall = recall_score(y_test, y_pred, zero_division=0)
    accuracy = accuracy_score(y_test, y_pred)

    baseline_results.append({
        "Модель": model_name,
        "ROC AUC": roc_auc,
        "F1": f1,
        "Recall": recall,
        "Accuracy": accuracy
    })
    cm = confusion_matrix(y_test, y_pred)
    print(cm)

# Преобразование результатов в DataFrame для удобного просмотра
baseline_results_df = pd.DataFrame(baseline_results)
baseline_results_df = baseline_results_df.sort_values(by="ROC AUC", ascending=False).reset_index(drop=True)
baseline_results_df
```

Рисунок 16. Обучение моделей без подбора гиперпараметров

Оценка качества моделей

Оценим качество работы моделей. На Рисунке 17 приведены 4 метрики: ROC-AUC, Accuracy, Recall и F1-мера.

	Модель	ROC AUC	F1	Recall	Accuracy
0	XGBoost	0.888155	0.633922	0.587160	0.836728
1	Logistic Regression	0.882622	0.603673	0.531037	0.832122
2	Random Forest	0.855675	0.598570	0.569303	0.816153
3	Support Vector Machine	0.844653	0.606941	0.524235	0.836524
4	K-Nearest Neighbors	0.840356	0.615419	0.592262	0.821783

Рисунок 17. Оценка качества работы моделей baseline-решения

Лучшие результаты по основной метрике показала модель XGBoost. Худшие результаты у модели KNN для классификации.

6 ПОДБОР ГИПЕРПАРАМЕТРОВ

Производится подбор гиперпараметров для выбранных моделей.

Обучение моделей

Обучим выбранные модели, подбирая гиперпараметры с помощью GridSearchCV и кросс-валидации по методу StratifiedK-Fold (см. Рисунок 18, Рисунок 19).

```
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import GridSearchCV
counts = y_train.value_counts()
scale_pos_weight_val = counts[0] / counts[1] if counts[1] != 0 else 1
class_weight_val = 'balanced'

# Определение моделей и сеток гиперпараметров для GridSearchCV
param_grids = {
    "Logistic Regression": [
        { # Сетка для solver='saga'
            'solver': ['saga'],
            'penalty': ['l1', 'l2'],
            'C': [0.01, 0.1, 1, 10],
            'class_weight': [None, class_weight_val],
            'max_iter': [1000, 2000, 3000]
        },
        { # Сетка для solver='liblinear'
            'solver': ['liblinear'],
            'penalty': ['l1', 'l2'],
            'C': [0.01, 0.1, 1, 10],
            'class_weight': [None, class_weight_val],
            'max_iter': [1000]
        }
    ],
    "K-Nearest Neighbors": {
        'n_neighbors': [3, 5, 7, 9],
        'weights': ['uniform', 'distance'],
        'metric': ['minkowski', 'manhattan']
    },
    "Support Vector Machine": {
        'C': [0.1, 1, 10],
        'kernel': ['rbf', 'linear'],
        'gamma': ['scale', 'auto'],
        'class_weight': [None, class_weight_val]
    },
}
```

Рисунок 18. Обучение моделей с подбором гиперпараметров

```

        "XGBoost": {
            'n_estimators': [100, 200],
            'learning_rate': [0.01, 0.1],
            'max_depth': [3, 5, 7],
            'subsample': [0.7, 0.9],
            'colsample_bytree': [0.7, 0.9],
            'gamma': [0, 0.1],
            'scale_pos_weight': [1, scale_pos_weight_val]
        }
    }

base_models_for_grid = {
    "Logistic Regression": LogisticRegression(random_state=52, n_jobs=-1), # max_iter будет из сетки
    "K-Nearest Neighbors": KNeighborsClassifier(n_jobs=-1),
    "Support Vector Machine": SVC(random_state=52, probability=True),
    "Random Forest": RandomForestClassifier(random_state=52, n_jobs=-1),
    "XGBoost": XGBClassifier(random_state=52, use_label_encoder=False, eval_metric='logloss', n_jobs=-1)
}

best_estimators = {}
tuned_model_results = []

for model_name in base_models_for_grid:
    print(f"\nПодбор гиперпараметров для: {model_name}")
    model = base_models_for_grid[model_name]
    grid = param_grids[model_name]
    stratified_kfold_splitter = StratifiedKFold(n_splits=3, shuffle=True, random_state=52)
    grid_search = GridSearchCV(estimator=model, param_grid=grid, cv=stratified_kfold_splitter, scoring='roc_auc', n_jobs=-1, verbose=1)
    grid_search.fit(X_train, y_train)
    print(f"Лучшие параметры для {model_name}: {grid_search.best_params_}")
    print(f"Лучший ROC AUC на CV для {model_name}: {grid_search.best_score_:.4f}")
    best_estimators[model_name] = grid_search.best_estimator_

```

Рисунок 19. Обучение моделей с подбором гиперпараметров

Оценка качества моделей

Оценим качество работы моделей с подбором гиперпараметров. На Рисунке 20 приведены 4 метрики: ROC-AUC, Accuracy, Recall и F1-мера.

	Модель	ROC AUC	F1	Recall	Accuracy
0	XGBoost (Tuned)	0.895514	0.637524	0.573554	0.842973
1	Random Forest (Tuned)	0.893857	0.639375	0.574405	0.843996
2	Logistic Regression (Tuned)	0.882639	0.601263	0.526361	0.831917
3	Support Vector Machine (Tuned)	0.881629	0.592795	0.507228	0.832224
4	K-Nearest Neighbors (Tuned)	0.869245	0.623260	0.590136	0.828232

Рисунок 20. Оценка качества работы моделей оптимального решения

Мы видим, что лучшие результаты относительно основной метрики показывает классификатор XGBoost. Подбор гиперпараметров не позволил значительно улучшить качество работы моделей.

Стоит отметить, что после подбора гиперпараметров некоторые метрики упали, пусть и незначительно.

7 СРАВНЕНИЕ РЕШЕНИЙ

На Рисунке 21 и Рисунке 22 представлены столбчатые диаграммы позволяющие сравнить AUC ROC и F1-меру моделей базового и оптимального решения.

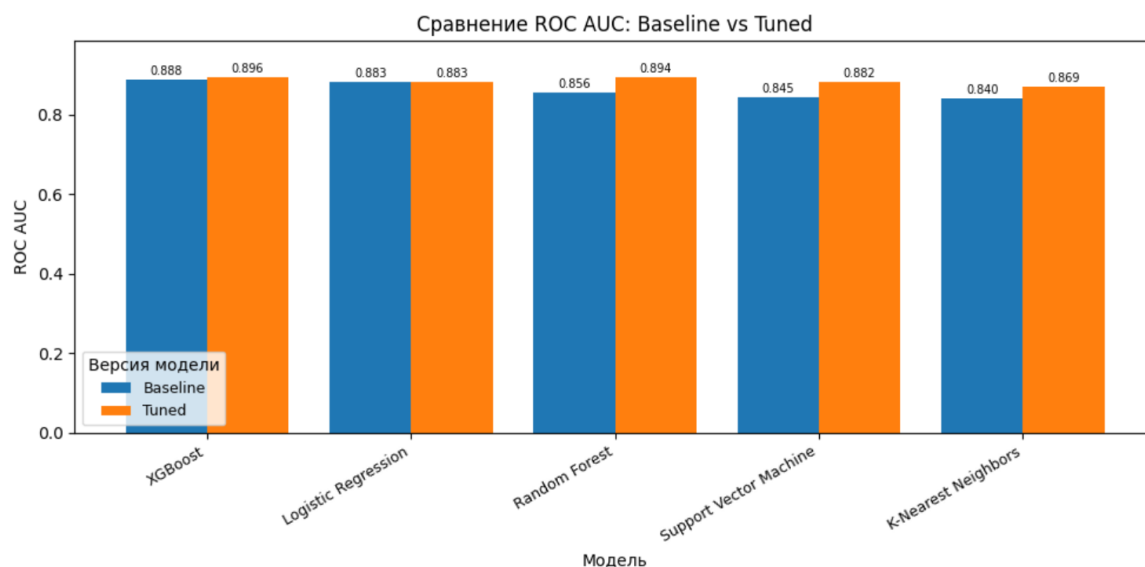


Рисунок 21. Сравнение решений по ROC AUC

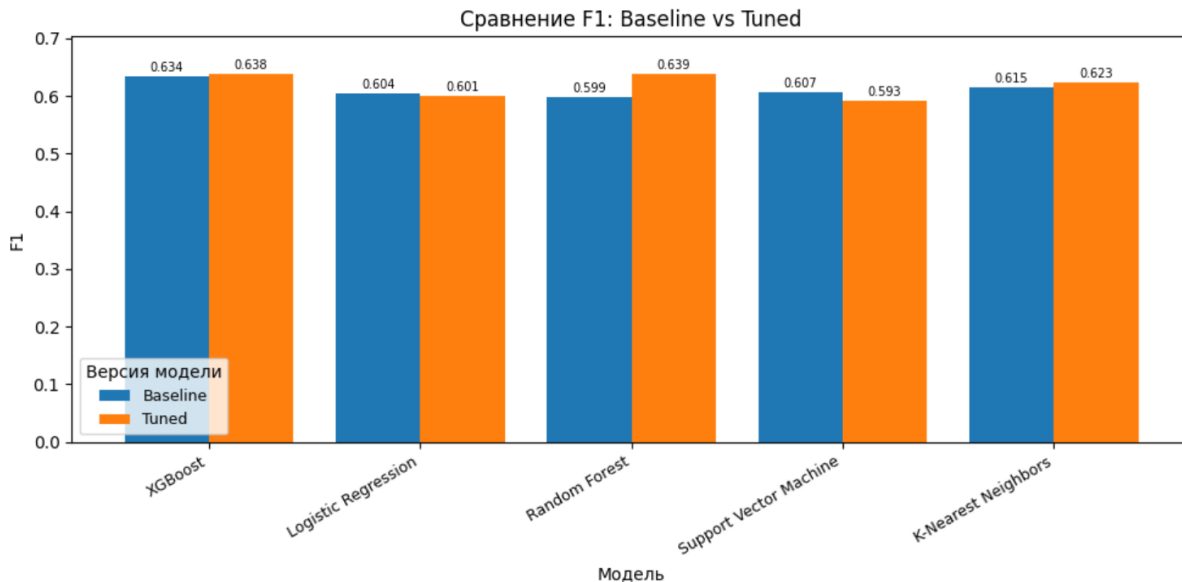


Рисунок 22. Сравнение решений по F1-мере

8 ВЕБ-ПРИЛОЖЕНИЕ

Создадим веб-приложение для демонстрации модели XGBoost с помощью фреймворка streamlit. Приложение позволяет производить подбор гиперпараметров модели (см. Рисунок 23), изменять размер тестовой выборки(см. Рисунок 24), смотреть информацию по входным данным (см. Рисунок 25.1, Рисунок 25.2, Рисунок 25.3, Рисунок 25.4), где в качестве данных представляется финальная версия датасета после всех обработок данных, производить обучение с введенными гиперпараметрами, а также просматривать информацию по обученной модели, а именно: оценку выбранных метрик (см. Рисунок 26), матрицу ошибок (см. Рисунок 27), ROC-кривую (см. Рисунок 28) и важность признаков (см. Рисунок 29).

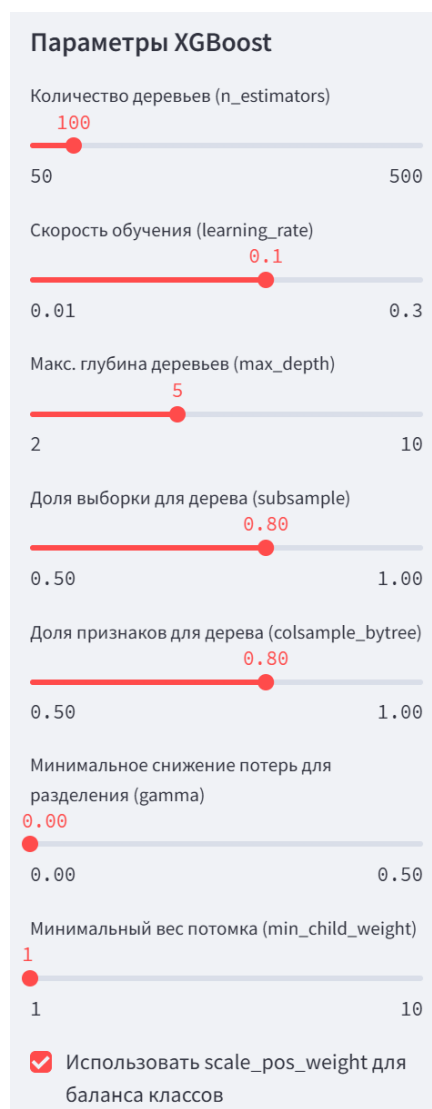


Рисунок 23. Подбор гиперпараметров

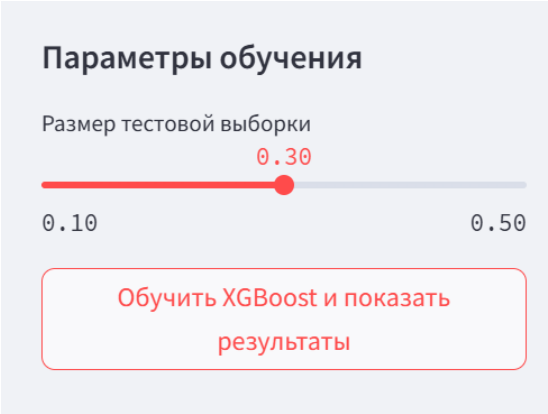


Рисунок 24. Изменение размера тестовой выборки

Обзор предоставленных данных

Признаки для модели (6): relationship_woe, education.num, occupation_woe, age, hours.per.week, sex_woe

Целевая переменная: income (0: <=50K, 1: >50K)

Первые 5 строк

	relationship_woe	education.num	occupation_woe	age	hours.per.week	sex_woe	income
0	1.0151	-0.4201	-0.4962	3.7696	-0.0354	0.9477	0
1	1.0151	-0.4201	-1.0837	3.1831	-1.8172	0.9477	0
2	1.545	-0.0314	-0.4962	2.0101	-0.0354	0.9477	0
3	1.545	-2.3636	0.7977	1.1304	-0.0354	0.9477	0
4	3.1574	-0.0314	-0.4962	0.1773	-0.0354	0.9477	0

Рисунок 25.1. Вершина датасета

Описательная статистика признаков

	relationship_woe	education.num	occupation_woe	age	hours.per.week	sex_woe
count	32561	32561	32561	32561	32561	32561
mean	0.5445	0.0000000000000001	0.2014	-0.0000000000000002	-0.0000000000000002	0.094
std	1.5208	1	0.9169	1	1	0.6003
min	-1.0485	-3.5297	-1.0837	-1.5822	-3.194	-0.3281
25%	-0.9416	-0.4201	-0.4962	-0.7758	-0.0354	-0.3281
50%	1.0151	-0.0314	0.0794	-0.116	-0.0354	-0.3281
75%	1.545	0.746	0.7134	0.6905	0.3695	0.9477
max	3.1574	2.3008	3.4475	3.7696	4.743	0.9477

Рисунок 25.2. Описательная статистика датасета

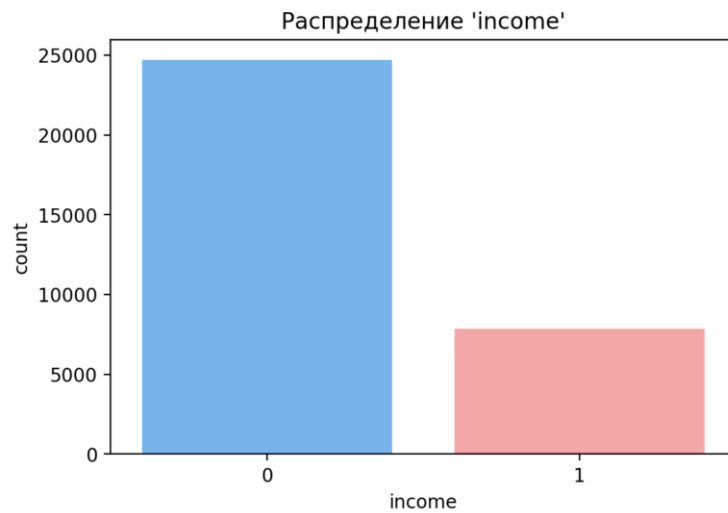


Рисунок 25.3. Распределение целевой переменной

Распределения признаков

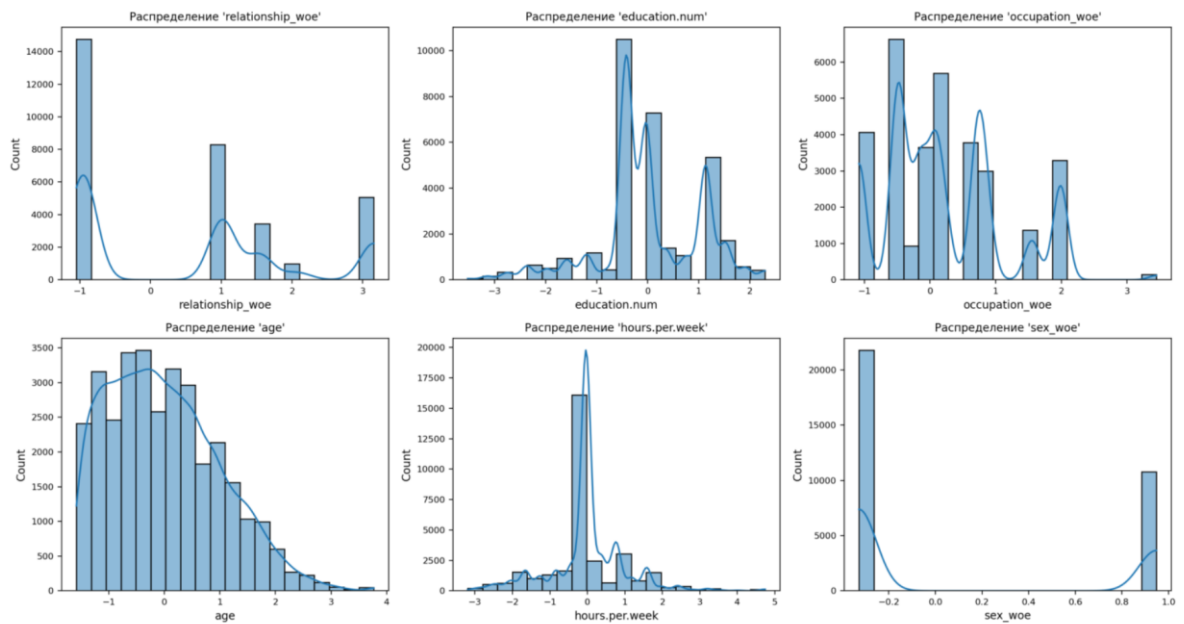


Рисунок 25.4. Распределение описательных признаков

Модель: XGBoost Classifier

Обзор данных Результаты моделирования XGBoost

Начинаем обучение XGBoost...

Размер обучающей выборки: 22792, Тестовой: 9769

Обучение XGBoost завершено за 0.08 сек.

Метрики качества XGBoost на тестовой выборке

ROC AUC	Accuracy
0.8947	0.7908
F1-score (для >50K)	Recall (для >50K)
0.6626	0.8533

Рисунок 26. Метрики обученной модели

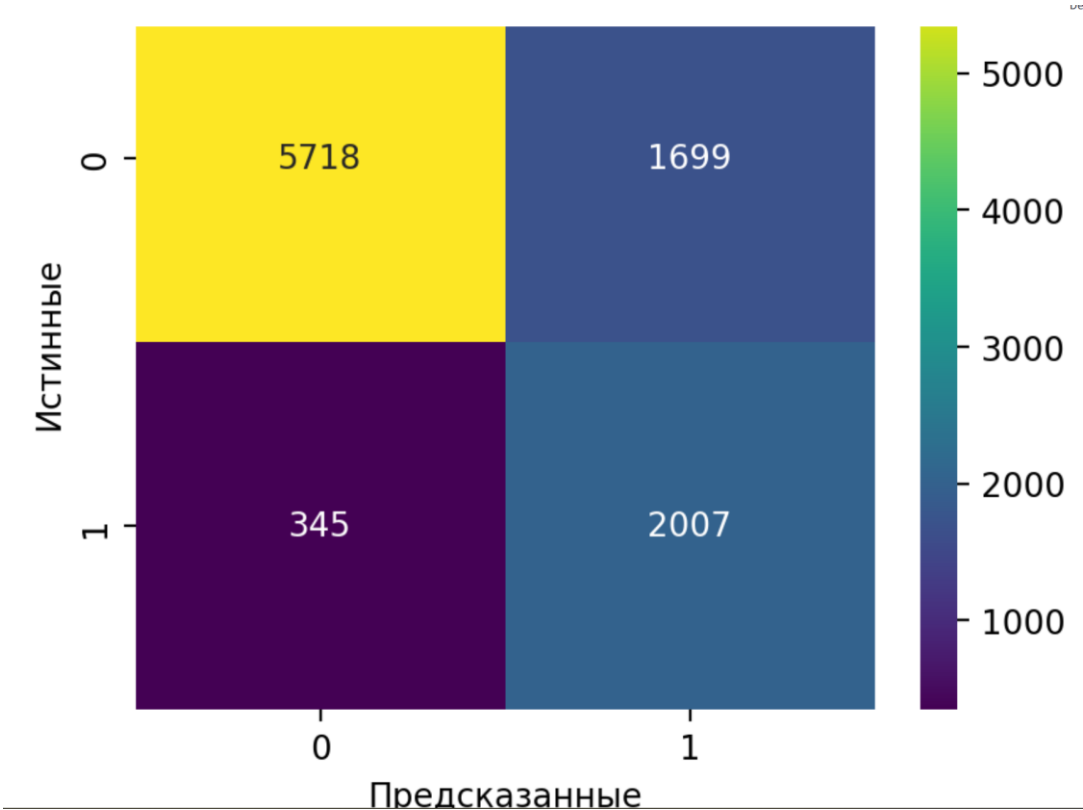


Рисунок 27. Матрица ошибок обученной модели

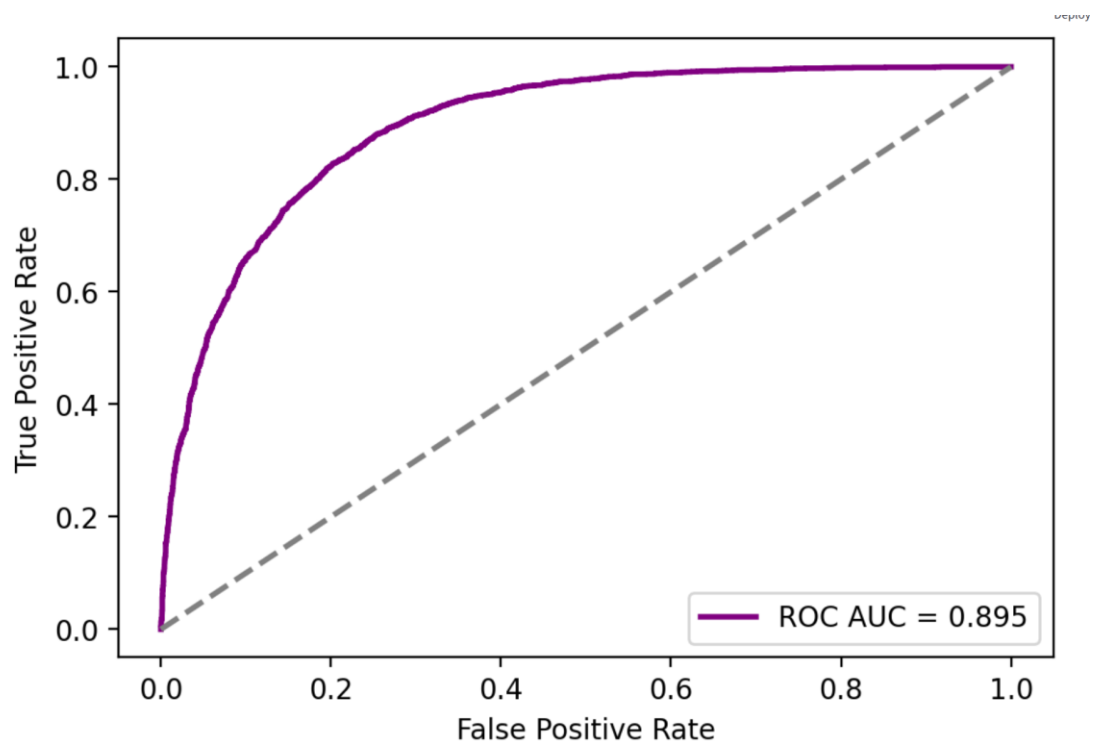


Рисунок 28. ROC-кривая обученной модели

Важность признаков (XGBoost)

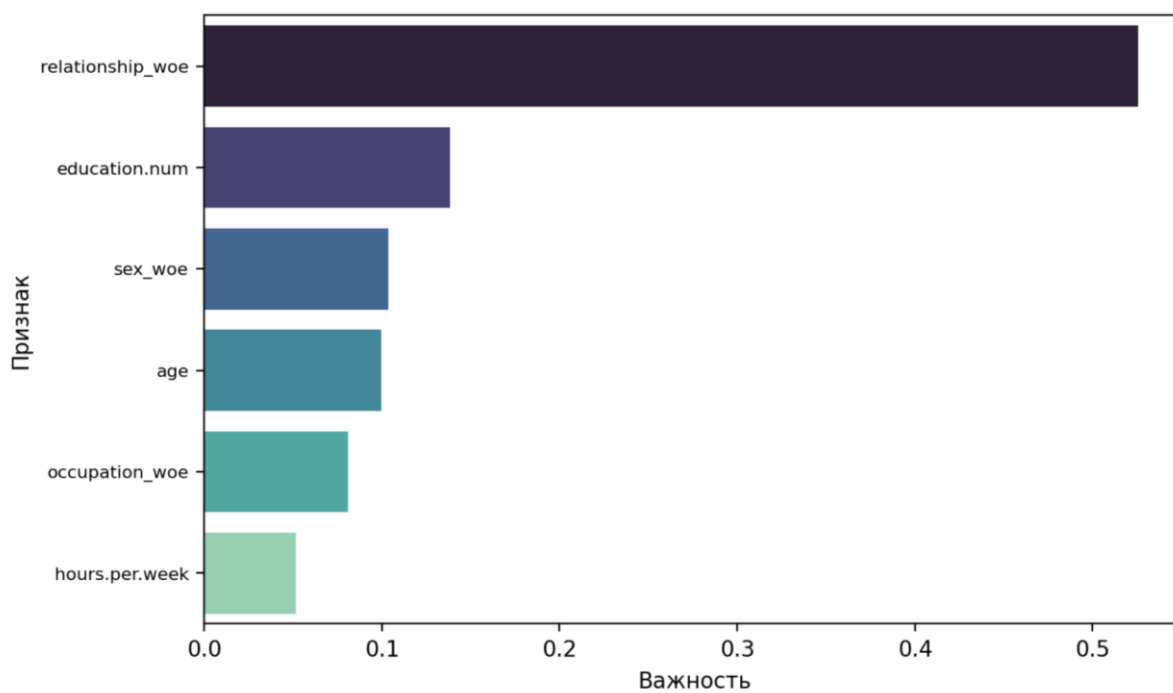


Рисунок 29. Важность признаков обученной модели

ЗАКЛЮЧЕНИЕ

В рамках научно-исследовательской работы была решена задача бинарной классификации – определение величины зарплаты человека. В ходе исследования были получены два решения: базовое и оптимальное за счет подбора гиперпараметров.

Наилучшего качества классификации по выбранным метрикам достигла модель XGBoost после подбора гиперпараметров.

Для демонстрации влияния гиперпараметров на результаты работы модели было разработано веб-приложение.

Построенные модели показывают достаточно хорошие результаты работы, позволяющие с достаточной уверенностью разделять людей на уровень дохода по их данным. Существует некоторое пространство для улучшения, связанное с использованием большего количества признаков и более тщательной настройки гиперпараметров.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дьяконов А. Блог Александра Дьяконова [Электронный ресурс]. URL: <https://alexanderdyakonov.wordpress.com/> (дата обращения: 03.05.2025).
2. Streamlit. Документация Streamlit [Электронный ресурс]. URL: <https://streamlit.io/> (дата обращения: 03.05.2025).
3. Открытый курс машинного обучения. Тема 10: Градиентный бустинг [Электронный ресурс]. URL: <https://habr.com/ru/companies/ods/articles/327250/> (дата обращения: 03.05.2025).
4. scikit-learn. Supervised learning [Электронный ресурс]. URL: https://scikit-learn.org/stable/supervised_learning.html (дата обращения: 03.05.2025).
5. Гапанюк Ю. Е. Репозиторий курса "Технологии машинного обучения", бакалавриат, 6 семестр [Электронный ресурс]. URL: https://github.com/ugapanyuk/courses_current/wiki/COURSE_TMO_SPRING_2025/ (дата обращения: 03.05.2025).
6. Метрики в задачах машинного обучения [Электронный ресурс]. URL: <https://habr.com/ru/companies/ods/articles/328372/> (дата обращения: 03.05.2025).