Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №2
По дисциплине: «ОМО»
Тема: «Линейные модели для задач регрессии и классификации»

Выполнил:
Студент 3-го курса
Группы АС-66
Ануфриенко М. А.
Проверил:
Крощенко А. А.

Брест 2025

Цель: Изучить применение линейной и логистической регрессии для решения практических задач. Научиться обучать модели, оценивать их качество с помощью соответствующих метрик и интерпретировать результаты.

Общее задание: выполнить задания по варианту (регрессия и классификация), построить все требуемые визуализации и рассчитать метрики, написать отчет, создать пул-реквест в репозиторий с кодом решения и отчетом в формате pdf.

Вариант 1

• Регрессия (Прогнозирование стоимости жилья в Калифорнии)

1. California Housing
2. Предсказать медианную стоимость дома (median_house_value)
3. Задания:
   - загрузите данные и разделите их на обучающую и тестовую выборки;
   - обучите модель линейной регрессии на обучающих данных;
   - сделайте предсказания для тестовой выборки;
   - оцените качество модели, рассчитав метрики MSE (Mean Squared Error) и R2 (Coefficient of Determination);
   - визуализируйте результат: постройте диаграмму рассеяния для признака median_income (медианный доход) и целевой переменной, нанеся на неё линию регрессии.

• Классификация (Прогнозирование выживаемости на "Титанике")

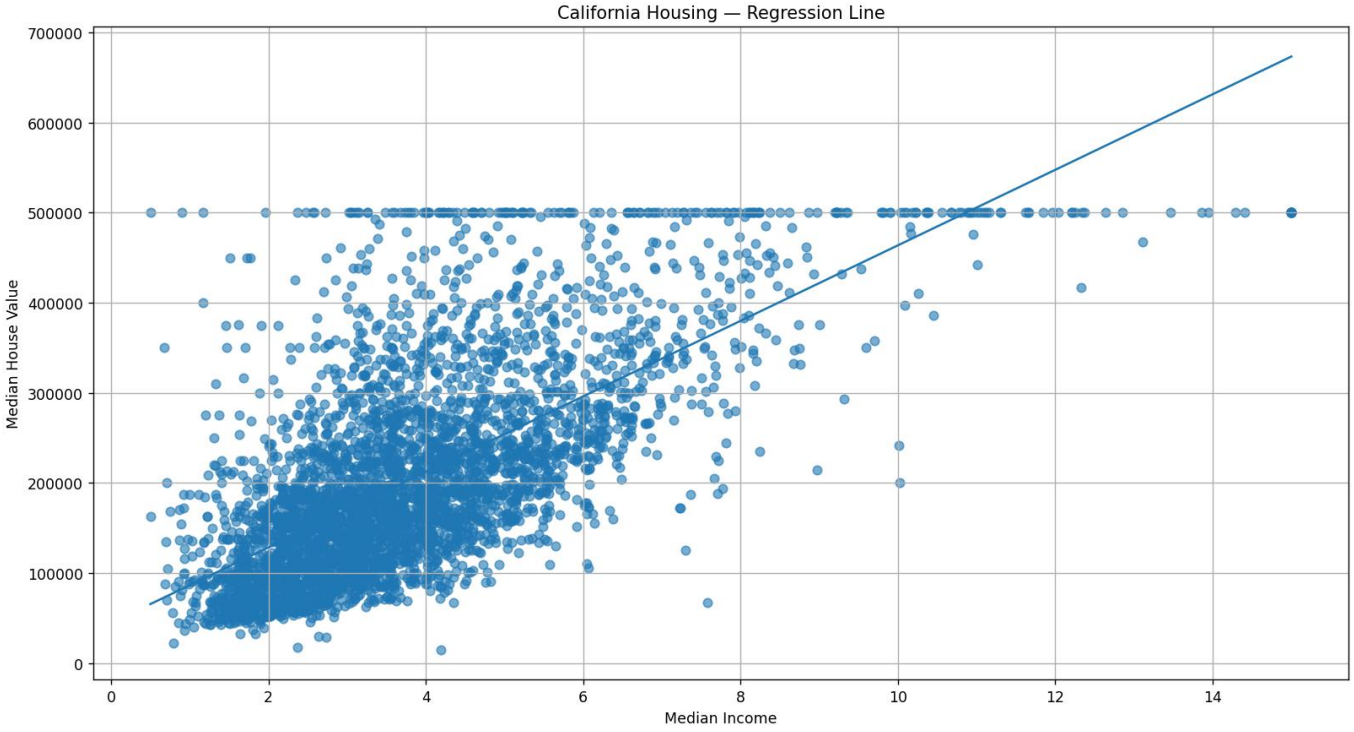
1. Titanic
2. Предсказать, выжил ли пассажир (Survived)
3. Задания:
   - загрузите и предварительно обработайте данные (заполните пропуски, преобразуйте категории в числа);
   - обучите модель логистической регрессии;
   - оцените качество модели, рассчитав Accuracy, Precision и Recall;
   - постройте и проанализируйте матрицу ошибок (confusion matrix).

```
import pandas as pd  # Импортируем библиотеку pandas для работы с таблицами
import numpy as np  # Импортируем numpy для работы с числовыми данными
import matplotlib.pyplot as plt  # Импортируем matplotlib для визуализации графиков
from sklearn.model_selection import train_test_split  # Импортируем для разделения данных на train/test
from sklearn.linear_model import LinearRegression, LogisticRegression  # Импортируем модели линейной и логистической регрессии
from sklearn.metrics import (
    mean_squared_error, r2_score,  # Метрики регрессии (MSE, R2)
    accuracy_score, precision_score, recall_score, confusion_matrix  # Метрики классификации
)
from sklearn.impute import SimpleImputer  # Импортируем инструмент для заполнения пропусков

# ========================================================
#                 ЧАСТЬ 1 — РЕГРЕССИЯ
# ========================================================
```

```
# === Загрузка набора данных California Housing ===
calif_df = pd.read_csv("california_housing.csv")  # Загружаем датасет California Housing из CSV

# Выбираем только числовые признаки
numeric_cols = [  # Список колонок с числовыми признаками
    "housing_median_age",  # Возраст домов
    "total_rooms",  # Общее количество комнат
    "total_bedrooms",  # Число спален
    "population",  # Население района
    "households",  # Количество домохозяйств
    "median_income"  # Средний доход
]

X = calif_df[numeric_cols]  # Формируем матрицу признаков

y = calif_df["median_house_value"]  # Целевая переменная — медианная стоимость жилья

# === Обработка пропусков ===
imputer = SimpleImputer(strategy='median')  # Создаем импьютер для заполнения медианой
X = imputer.fit_transform(X)  # Заполняем пропущенные значения в X

# === Разделение на обучающую и тестовую выборки ===
X_train, X_test, y_train, y_test = train_test_split(  # Делим данные на обучающие и тестовые
    X, y, test_size=0.2, random_state=42  # 20% — тест, фиксируем random_state
)

# === Обучение линейной регрессии ===
lr = LinearRegression()  # Создаем модель линейной регрессии
lr.fit(X_train, y_train)  # Обучаем модель на тренировочных данных

# === Предсказание ===
y_pred = lr.predict(X_test)  # Делаем прогнозы на тестовом наборе

# === Метрики ===
mse = mean_squared_error(y_test, y_pred)  # Вычисляем среднеквадратичную ошибку (MSE)
r2 = r2_score(y_test, y_pred)  # Вычисляем коэффициент детерминации (R2)

print("\n==== California Housing: Linear Regression ====")  # Заголовок
print("MSE:", mse)  # Выводим MSE
print("R2:", r2)  # Выводим R2

# === Визуализация: разброс median_income vs median_house_value ===
# Берём только median_income для визуализации
income = calif_df["median_income"].values.reshape(-1, 1)  # Достаём колонку дохода и преобразуем в 2D
imputer2 = SimpleImputer(strategy='median')  # Создаем новый импьютер для дохода
income = imputer2.fit_transform(income)  # Заполняем пропуски в доходе

X_train_i, X_test_i, y_train_i, y_test_i = train_test_split(  # Делим доход и таргет на train/test
    income, y, test_size=0.2, random_state=42
)
```

```
lr_income = LinearRegression()  # Создаем модель для одной переменной
lr_income.fit(X_train_i, y_train_i)  # Обучаем модель

y_pred_line = lr_income.predict(X_test_i)  # Предсказываем значения

plt.figure(figsize=(8, 6))  # Задаём размер графика
plt.scatter(X_test_i, y_test_i, alpha=0.6)  # Наносим точки данных на график
x_range = np.linspace(X_test_i.min(), X_test_i.max(), 200).reshape(-1, 1)  # Создаём
диапазон значений
plt.plot(x_range, lr_income.predict(x_range))  # Строим линию регрессии
plt.xlabel("Median Income")  # Ось X
plt.ylabel("Median House Value")  # Ось Y
plt.title("California Housing — Regression Line")  # Заголовок графика
plt.grid(True)  # Включаем сетку
plt.show()  # Показываем график
```

```
==== California Housing: Linear Regression ====
MSE: 5968852333.910648
R2: 0.5445046216087996
```

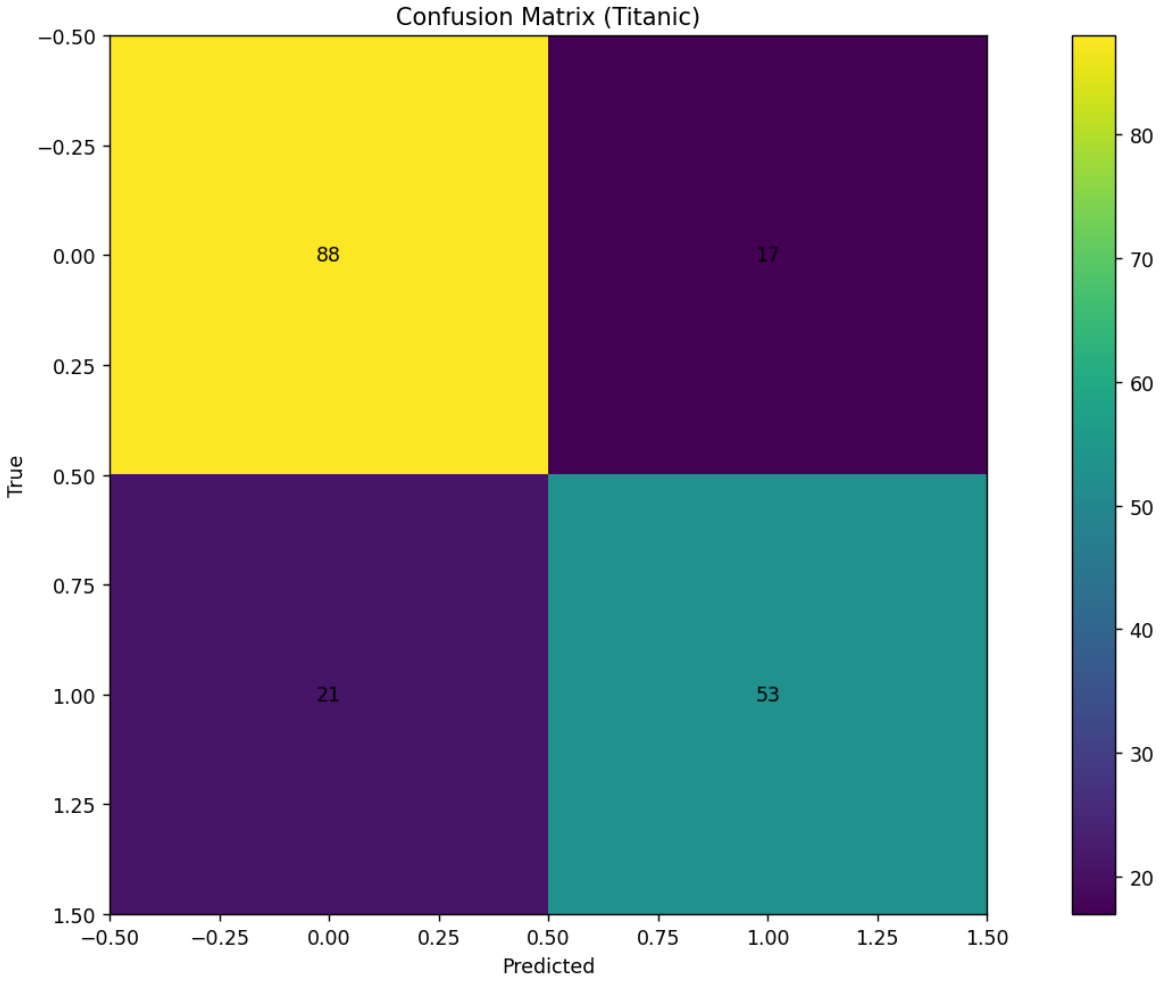



```
# ============================================================
#                  ЧАСТЬ 2 — КЛАССИФИКАЦИЯ
# ============================================================

# === Загрузка набора данных Titanic ===
df = pd.read_csv("Titanic-Dataset.csv")  # Загружаем датасет Titanic из CSV

# === Предобработка данных ===

# Заполняем пропуски в Age медианой, если колонка есть
if "Age" in df.columns:
    df["Age"] = df["Age"].fillna(df["Age"].median())  # Заполняем NaN медианой
```

```
# Embarked заполним модой (наиболее частым значением), если колонка есть
if "Embarked" in df.columns:
    df["Embarked"] = df["Embarked"].fillna(df["Embarked"].mode()[0])  # Заполняем модой

# Удалим ненужные столбцы, если они существуют
drop_cols = ["PassengerId", "Name", "Ticket", "Cabin"]  # Список столбцов для удаления
for c in drop_cols:
    if c in df.columns:  # Проверяем, есть ли столбец
        df.drop(columns=c, inplace=True)  # Удаляем столбец

# Преобразуем признак пола в числовой формат (0/1)
# Преобразуем пол в числовой формат

# Кодируем категориальный признак Embarked с помощью one-hot
if "Embarked" in df.columns:  # Если колонка осталась
    embarked_dummies = pd.get_dummies(df["Embarked"], prefix="Emb", drop_first=True)  #
Создаем дамми-переменные
    df = pd.concat([df, embarked_dummies], axis=1)  # Добавляем дамми-признаки
    df.drop(columns=["Embarked"], inplace=True)  # Удаляем исходную колонку

# === Целевая переменная и признаки ===
y_t = df["Survived"]  # Целевая переменная — выжил или нет
X_t = df.drop(columns=["Survived"])  # Признаки — всё, кроме таргета

# Удалим object-колонки, если они остались
for col in X_t.columns:
    if X_t[col].dtype == "object":  # Если тип object
        X_t.drop(columns=[col], inplace=True)  # Удаляем колонку

# === Разделение на обучающую и тестовую выборки ===
X_train_t, X_test_t, y_train_t, y_test_t = train_test_split(  # Разделяем данные на
train/test
    X_t, y_t, test_size=0.2, random_state=42
)

# === Логистическая регрессия ===
logreg = LogisticRegression(max_iter=2000, solver="liblinear")  # Создаем модель
логистической регрессии
logreg.fit(X_train_t, y_train_t)  # Обучаем модель

# === Предсказание ===
y_pred_t = logreg.predict(X_test_t)  # Предсказываем на тесте

# === Метрики ===
acc = accuracy_score(y_test_t, y_pred_t)  # Вычисляем Accuracy
prec = precision_score(y_test_t, y_pred_t, zero_division=0)  # Вычисляем Precision
rec = recall_score(y_test_t, y_pred_t, zero_division=0)  # Вычисляем Recall

print("\n==== Titanic: Logistic Regression ====")  # Заголовок
print("Accuracy:", acc)  # Вывод accuracy
print("Precision:", prec)  # Вывод precision
print("Recall:", rec)  # Вывод recall

# === Матрица ошибок ===
```

```python
cm = confusion_matrix(y_test_t, y_pred_t)  # Вычисляем confusion matrix
print("\nConfusion Matrix:\n", cm)  # Печатаем матрицу

plt.figure(figsize=(5, 4))  # Размер графика
plt.imshow(cm)  # Показываем матрицу путаницы
plt.colorbar()  # Добавляем цветовую шкалу
plt.title("Confusion Matrix (Titanic)")  # Заголовок
plt.xlabel("Predicted")  # Подпись оси X
plt.ylabel("True")  # Подпись оси Y

for i in range(cm.shape[0]):  # Перебор строк
    for j in range(cm.shape[1]):  # Перебор столбцов
        plt.text(j, i, str(cm[i, j]), ha="center", va="center")  # Пишем число внутри клетки

plt.show()  # Показываем матрицу
```

```
==== Titanic: Logistic Regression ====
Accuracy: 0.7877094972067039
Precision: 0.7571428571428571
Recall: 0.7162162162162162

Confusion Matrix:
 [[88 17]
 [21 53]]
```



Вывод: я изучил применение линейной и логистической регрессии для решения практических задач и научился обучать модели, оценивать их качество с помощью соответствующих метрик и интерпретировать результаты.