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")

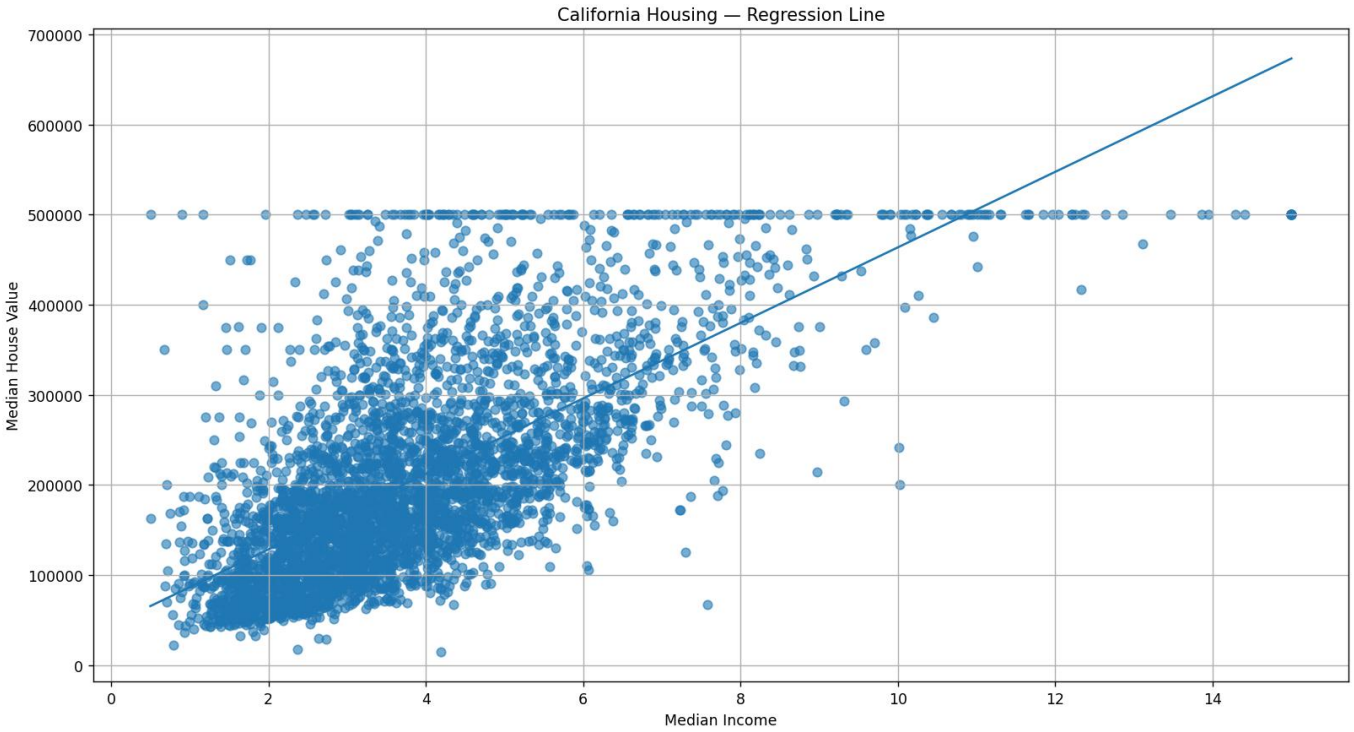
1. Titanic
2. Предсказать, выжил ли пассажир (Survived)
3. Задания:
   - загрузите и предварительно обработайте данные (заполните пропуски, преобразуйте категории в числа);
   - обучите модель логистической регрессии;
   - оцените качество модели, рассчитав Accuracy, Precision и Recall;
   - постройте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

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import (
    mean_squared_error, r2_score,
    accuracy_score, precision_score, recall_score, confusion_matrix
)
from sklearn.impute import SimpleImputer


# ======================================================
#                 PART 1 — REGRESSION
# ======================================================

# === Load California Housing dataset ===
calif_df = pd.read_csv("california_housing.csv")

# выберем числовые признаки
```

```python
numeric_cols = [
    "housing_median_age",
    "total_rooms",
    "total_bedrooms",
    "population",
    "households",
    "median_income"
]

X = calif_df[numeric_cols]
y = calif_df["median_house_value"]

# === Обработка пропусков ===
imputer = SimpleImputer(strategy='median')
X = imputer.fit_transform(X)

# === Train/test split ===
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# === Train Linear Regression ===
lr = LinearRegression()
lr.fit(X_train, y_train)

# === Predict ===
y_pred = lr.predict(X_test)

# === Metrics ===
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

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


# === Visualization: scatter median_income vs median_house_value ===
# Возьмем только median_income для визуализации
income = calif_df["median_income"].values.reshape(-1, 1)
imputer2 = SimpleImputer(strategy='median')
income = imputer2.fit_transform(income)

X_train_i, X_test_i, y_train_i, y_test_i = train_test_split(
    income, y, test_size=0.2, random_state=42
)

lr_income = LinearRegression()
lr_income.fit(X_train_i, y_train_i)
y_pred_line = lr_income.predict(X_test_i)

plt.figure(figsize=(8, 6))
plt.scatter(X_test_i, y_test_i, alpha=0.6)
x_range = np.linspace(X_test_i.min(), X_test_i.max(), 200).reshape(-1, 1)
plt.plot(x_range, lr_income.predict(x_range))
```

```
plt.xlabel("Median Income")
plt.ylabel("Median House Value")
plt.title("California Housing — Regression Line")
plt.grid(True)
plt.show()
```

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

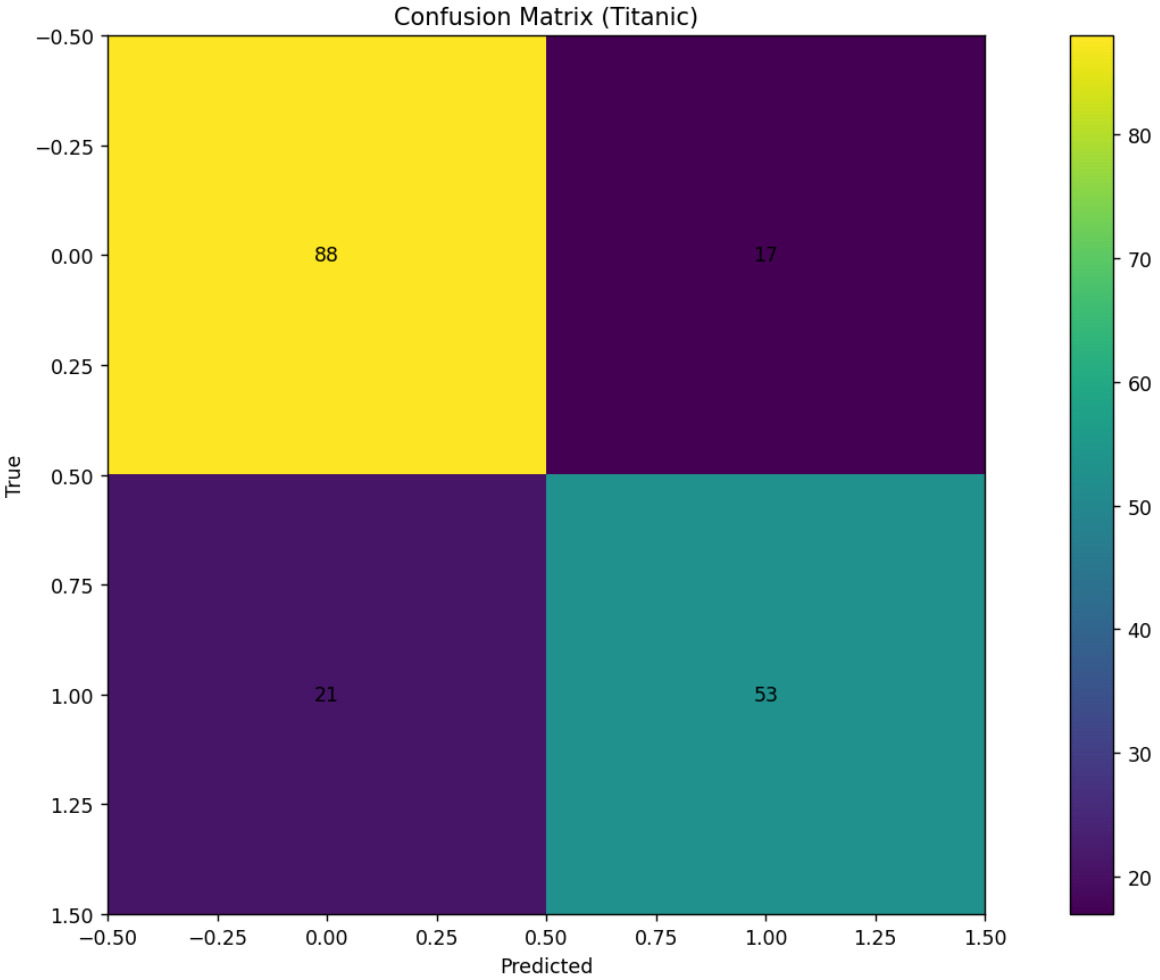



```
# ========================================================
#                  PART 2 — CLASSIFICATION
# ========================================================

# === Load Titanic dataset ===
df = pd.read_csv("Titanic-Dataset.csv")

# === Preprocessing ===

# Заполним пропуски в Age медианой
if "Age" in df.columns:
    df["Age"] = df["Age"].fillna(df["Age"].median())

# Embarked заполним модой
if "Embarked" in df.columns:
    df["Embarked"] = df["Embarked"].fillna(df["Embarked"].mode()[0])

# Удалим ненужные столбцы
drop_cols = ["PassengerId", "Name", "Ticket", "Cabin"]
for c in drop_cols:
```

```
    if c in df.columns:
        df.drop(columns=c, inplace=True)

# Sex → 0/1
df["Sex"] = df["Sex"].map({"male": 0, "female": 1})

# One-hot encoding Embarked
if "Embarked" in df.columns:
    embarked_dummies = pd.get_dummies(df["Embarked"], prefix="Emb", drop_first=True)
    df = pd.concat([df, embarked_dummies], axis=1)
    df.drop(columns=["Embarked"], inplace=True)

# === Target and features ===
y_t = df["Survived"]
X_t = df.drop(columns=["Survived"])

# Удалим любые остаточные object-колонки
for col in X_t.columns:
    if X_t[col].dtype == "object":
        X_t.drop(columns=[col], inplace=True)

# === Train/test split ===
X_train_t, X_test_t, y_train_t, y_test_t = train_test_split(
    X_t, y_t, test_size=0.2, random_state=42
)

# === Logistic Regression ===
logreg = LogisticRegression(max_iter=2000, solver="liblinear")
logreg.fit(X_train_t, y_train_t)

# === Predict ===
y_pred_t = logreg.predict(X_test_t)

# === Metrics ===
acc = accuracy_score(y_test_t, y_pred_t)
prec = precision_score(y_test_t, y_pred_t, zero_division=0)
rec = recall_score(y_test_t, y_pred_t, zero_division=0)

print("\n==== Titanic: Logistic Regression ====")
print("Accuracy:", acc)
print("Precision:", prec)
print("Recall:", rec)

# === Confusion Matrix ===
cm = confusion_matrix(y_test_t, y_pred_t)
print("\nConfusion Matrix:\n", cm)

plt.figure(figsize=(5, 4))
plt.imshow(cm)
plt.colorbar()
plt.title("Confusion Matrix (Titanic)")
plt.xlabel("Predicted")
plt.ylabel("True")

for i in range(cm.shape[0]):
```

```
    for j in range(cm.shape[1]):
        plt.text(j, i, str(cm[i, j]), ha="center", va="center")
plt.show()
```

```
==== 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.