**Министерство образования Республики Беларусь**

**Учреждение образования**

**«Брестский Государственный технический университет»**

**Кафедра ИИТ**

**Лабораторная работа №2**

По дисциплине: «Основы машинного обучения»

Тема: «Линейные модели для задач регрессии и классификации»

**Выполнила:**
Студентка 3 курса
Группы АС-65
Рапин Е. Ю.
**Проверил:**
Крощенко А. А.

Брест 2025

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

## Вариант 11

**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;

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_csv('california_housing.csv')

print("=== ЗАДАНИЕ 1: Загрузка данных ===")
print(f"Размерность данных: {data.shape}")
print(f"Первые 5 строк:\n{data.head()}")
print(f"Информация о данных:\n{data.info()}")
print(f"Пропущенные значения:\n{data.isnull().sum()}")
```

```
=== ЗАДАНИЕ 1: Загрузка данных ===
Размерность данных: (20640, 10)
Первые 5 строк:
   longitude  latitude  ...  median_house_value  ocean_proximity
0    -122.23     37.88  ...            452600.0         NEAR BAY
1    -122.22     37.86  ...            358500.0         NEAR BAY
2    -122.24     37.85  ...            352100.0         NEAR BAY
3    -122.25     37.85  ...            341300.0         NEAR BAY
4    -122.25     37.85  ...            342200.0         NEAR BAY
```

```
print("\n Разделение данных на обучающую и тестовую выборки")

X = data[['median_income']]
y = data['median_house_value']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

print(f"Размер обучающей выборки: {X_train.shape}")
print(f"Размер тестовой выборки: {X_test.shape}")
```

```
Разделение данных на обучающую и тестовую выборки
Размер обучающей выборки: (16512, 1)
Размер тестовой выборки: (4128, 1)
```

- обучите модель **линейной регрессии** на обучающих данных;

```
print("\n=== ЗАДАНИЕ 2: Обучение модели линейной регрессии ===")

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

print("Модель успешно обучена!")
print(f"Коэффициент (наклон): {model.coef_[0]:.4f}")
print(f"Пересечение: {model.intercept_:.4f}")
```

```
=== ЗАДАНИЕ 2: Обучение модели линейной регрессии ===
Модель успешно обучена!
Коэффициент (наклон): 41933.8494
Пересечение: 44459.7292
```

- сделайте предсказания для тестовой выборки;

```
print("\n=== ЗАДАНИЕ 3: Предсказания для тестовой выборки ===")

y_pred = model.predict(X_test)

print(f"Первые 5 предсказаний: {y_pred[:5]}")
print(f"Первые 5 реальных значений: {y_test.values[:5]}")
```

```
=== ЗАДАНИЕ 3: Предсказания для тестовой выборки ===
Первые 5 предсказаний: [114958.91676996 150606.88213964 190393.71844449 285059.38345102
 200663.31816103]
Первые 5 реальных значений: [ 47700.  45800. 500001. 218600. 278000.]
```

- оцените качество модели, рассчитав метрики **MSE** (Mean Squared Error) и **$R^2$** (Coefficient of Determination);

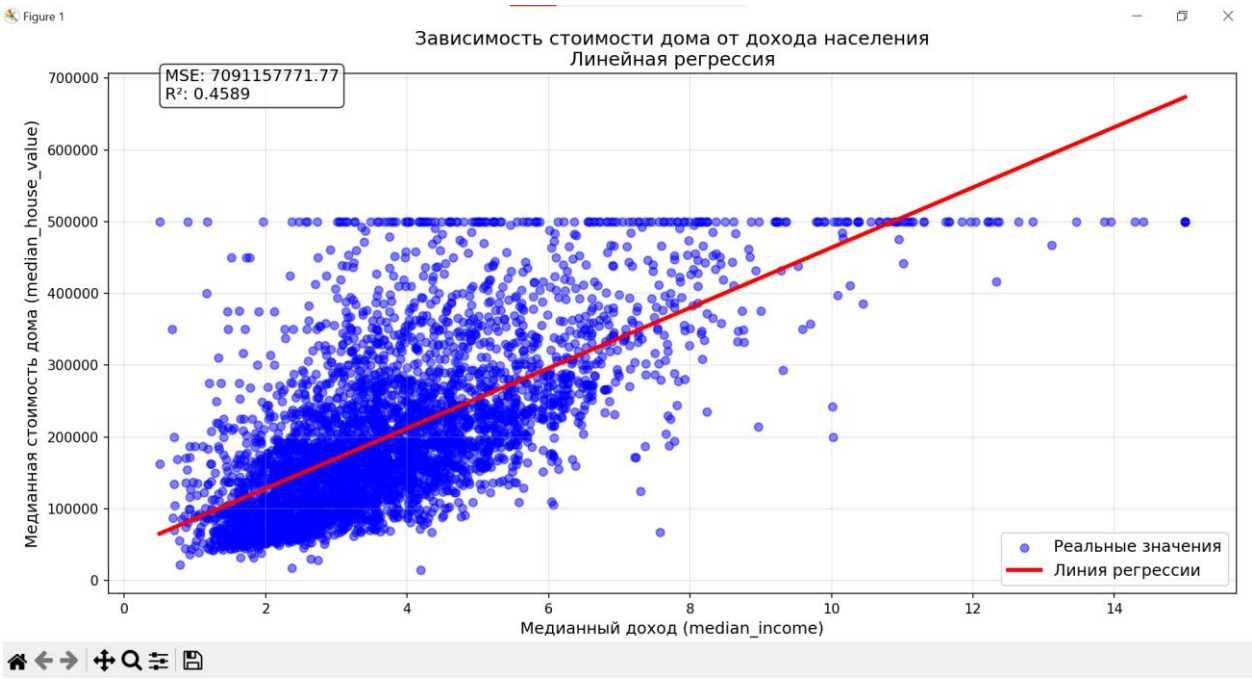
```
print("\n=== ЗАДАНИЕ 4: Оценка качества модели ===")

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

print(f"MSE: {mse:.2f}")
print(f"R²: {r2:.4f}")
```

```
=== ЗАДАНИЕ 4: Оценка качества модели ===
MSE: 7091157771.77
R²: 0.4589
```

- визуализируйте результат: постройте диаграмму рассеяния для признака median_income (медианный доход) и целевой переменной, нанеся на неё линию регрессии.

```
plt.figure(figsize=(12, 8))

plt.scatter(X_test, y_test, alpha=0.5, color='blue', label='Реальные
значения')
```

```
x_line = np.linspace(X_test.min(), X_test.max(), 100).reshape(-1, 1)
y_line = model.predict(x_line)
plt.plot(x_line, y_line, color='red', linewidth=3, label='Линия регрессии')

plt.xlabel('Медианный доход (median_income)', fontsize=12)
plt.ylabel('Медианная стоимость дома (median_house_value)', fontsize=12)
plt.title('Зависимость стоимости дома от дохода населения\nЛинейная
регрессия', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True, alpha=0.3)

plt.text(0.05, 0.95, f'MSE: {mse:.2f}\nR²: {r2:.4f}',
         transform=plt.gca().transAxes, fontsize=12,
         bbox=dict(boxstyle="round,pad=0.3", facecolor="white", alpha=0.8))

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




## 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);

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score,
confusion_matrix
from sklearn.preprocessing import LabelEncoder, StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('Titanic-Dataset.csv')

print("=== ЗАДАНИЕ 1: Загрузка данных ===")
```

```
print(f"Размерность данных: {df.shape}")
print(f"Первые 5 строк:\n{df.head()}")
print(f"Пропущенные значения:\n{df.isnull().sum()}")

df = df.copy()  # Создание копии для безопасной модификации
df['Age'] = df['Age'].fillna(df['Age'].median())
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])

df['HasCabin'] = df['Cabin'].notna().astype(int)
df.drop('Cabin', axis=1, inplace=True)

df = df.dropna(subset=['Fare'])

print(f"Пропущенные значения после обработки:\n{df.isnull().sum()}")
```

```
Пропущенные значения:
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
Пропущенные значения после обработки:
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
HasCabin       0
dtype: int64
```

```
df_processed = df.copy()

le_sex = LabelEncoder()
df_processed['Sex_encoded'] = le_sex.fit_transform(df_processed['Sex'])

le_embarked = LabelEncoder()
df_processed['Embarked_encoded'] =
le_embarked.fit_transform(df_processed['Embarked'])

df_processed['FamilySize'] = df_processed['SibSp'] + df_processed['Parch'] +
1

df_processed['IsAlone'] = (df_processed['FamilySize'] == 1).astype(int)

columns_to_drop = ['PassengerId', 'Name', 'Sex', 'Ticket', 'Embarked']
df_processed = df_processed.drop(columns_to_drop, axis=1)

print(f"\nДанные после предобработки:")
print(df_processed.head())
print(f"\nРазмерность после обработки: {df_processed.shape}")

X = df_processed.drop('Survived', axis=1)
y = df_processed['Survived']

print(f"\nПризнаки (X): {X.shape}")
print(f"Целевая переменная (y): {y.shape}")

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
```

```
)

print(f"\nОбучающая выборка: {X_train.shape}")
print(f"Тестовая выборка: {X_test.shape}")

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

- обучите модель **логистической регрессии**;

```
print("\n=== ЗАДАНИЕ 2: Обучение модели логистической регрессии ===")

logreg = LogisticRegression(random_state=42, max_iter=1000)
logreg.fit(X_train_scaled, y_train)

print("Модель логистической регрессии обучена!")
```

- оцените качество модели, рассчитав **Accuracy**, **Precision** и **Recall**;

```
print("\n=== ЗАДАНИЕ 3: Оценка качества модели ===")

y_pred = logreg.predict(X_test_scaled)
y_pred_proba = logreg.predict_proba(X_test_scaled)[:, 1]

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
```

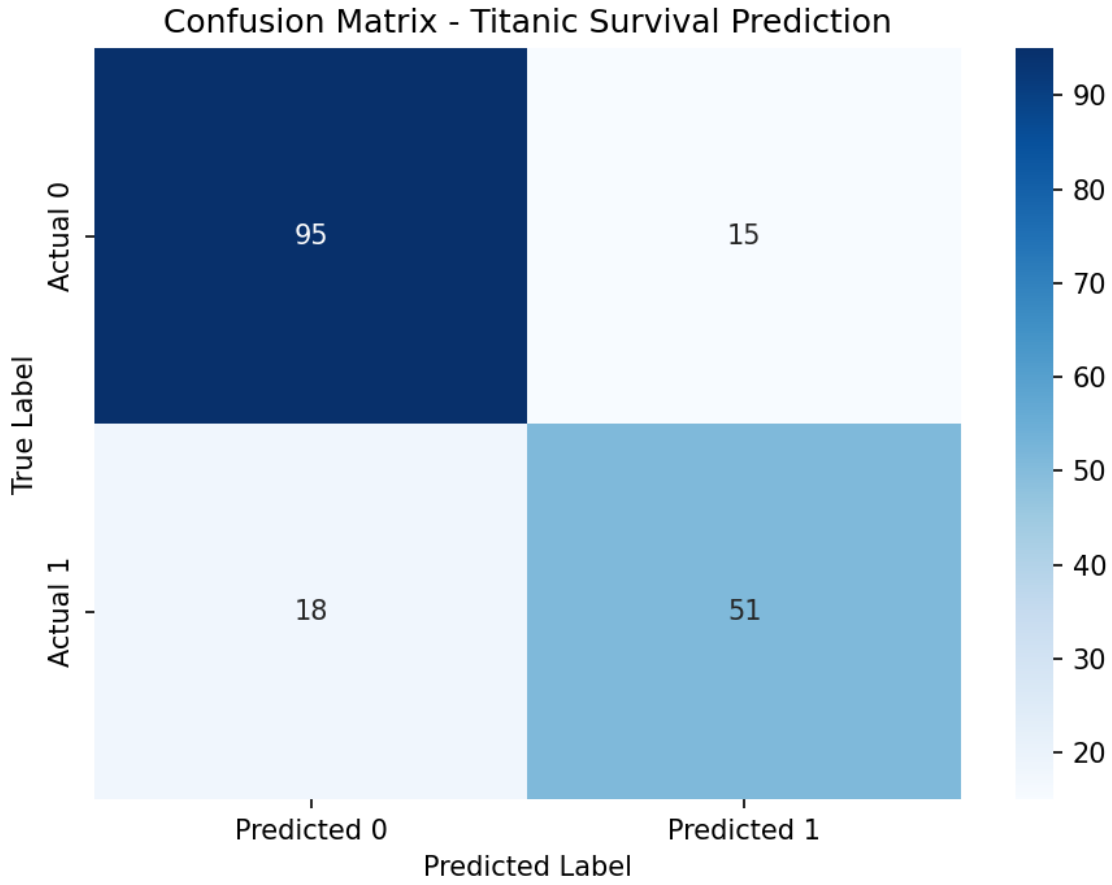
```
=== ЗАДАНИЕ 3: Оценка качества модели ===
Accuracy: 0.8156
Precision: 0.7727
Recall: 0.7391
```

- постройте и проанализируйте **матрицу ошибок** (confusion matrix).

```
print("\n=== ЗАДАНИЕ 4: Построение и анализ матрицы ошибок ===")

cm = confusion_matrix(y_test, y_pred)
print(f"Матрица ошибок:\n{cm}")
```

```
=== ЗАДАНИЕ 4: Построение и анализ матрицы ошибок ===
Матрица ошибок:
[[95 15]
 [18 51]]
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Predicted 0', 'Predicted 1'],
            yticklabels=['Actual 0', 'Actual 1'])
plt.title('Confusion Matrix - Titanic Survival Prediction')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()
```




```
tn, fp, fn, tp = cm.ravel()
print(f"\nАнализ матрицы ошибок:")
print(f"True Negatives (TN): {tn} - правильно предсказали смерть")
print(f"False Positives (FP): {fp} - ошибочно предсказали выживание")
print(f"False Negatives (FN): {fn} - ошибочно предсказали смерть")
print(f"True Positives (TP): {tp} - правильно предсказали выживание")
```

```
Анализ матрицы ошибок:
True Negatives (TN): 95 - правильно предсказали смерть
False Positives (FP): 15 - ошибочно предсказали выживание
False Negatives (FN): 18 - ошибочно предсказали смерть
True Positives (TP): 51 - правильно предсказали выживание
```

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