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
Осовец А.О.
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.

Вариант 7

• Регрессия (Прогнозирование цены автомобиля)
1. Car Price Prediction
2. Предсказать цену автомобиля (price)
3. Задания:
§ загрузите данные. Выберите 5-6 числовых признаков (например, horsepower, citympg, enginesize);
§ обучите модель линейной регрессии;
§ рассчитайте R2 и MAE;
§ визуализируйте зависимость price от horsepower с линией регрессии.

• Классификация (Прогнозирование уровня дохода)
1. Adult Census Income
2. Предсказать, превышает ли доход $50 тыс. в год (income >50K)
3. Задания:
§ загрузите данные, обработайте пропуски и категориальные признаки;
§ обучите модель логистической регрессии;
§ рассчитайте Accuracy, Precision и Recall;
§ постройте матрицу ошибок.

Код:

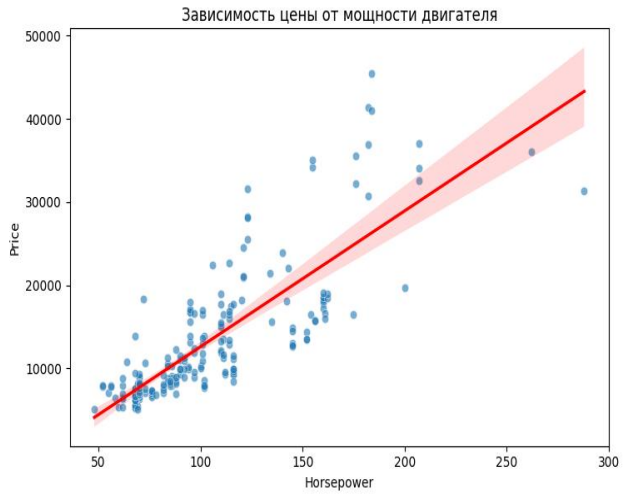
```
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error
from sklearn.preprocessing import StandardScaler
import warnings

warnings.filterwarnings('ignore')
# --------------------------------
# 1. РЕГРЕССИЯ: Car Price Prediction
# --------------------------------
def enhanced_regression_analysis():
    print("=== УЛУЧШЕННЫЙ АНАЛИЗ: Car Price Prediction ===")

    current_dir = os.path.dirname(os.path.abspath(__file__))
    project_root = os.path.abspath(os.path.join(current_dir, '..', '..', '..', '..', '..'))
    file_path = os.path.join(project_root, 'CarPrice_Assignment.csv')

    try:
        cars = pd.read_csv(file_path)
    except FileNotFoundError:
        print(f"Файл не найден: {file_path}")
        return None, None, None

    cars['power_to_weight'] = cars['horsepower'] / cars['curbweight']
    cars['mpg_combined'] = (cars['citympg'] + cars['highwaympg']) / 2
```

```
    cars['engine_efficiency'] = cars['horsepower'] / cars['enginesize']

    features = ["horsepower", "citympg", "highwaympg", "enginesize", "curbweight",
                "wheelbase", "carwidth", "carlength", "power_to_weight", "mpg_combined", "engine_efficiency"]

    X = cars[features]
    y = cars["price"]

    Q1 = X.quantile(0.25)
    Q3 = X.quantile(0.75)
    IQR = Q3 - Q1
    outlier_mask = ~((X < (Q1 - 1.5 * IQR)) | (X > (Q3 + 1.5 * IQR))).any(axis=1)
    X = X[outlier_mask]
    y = y[outlier_mask]

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, shuffle=True)

    reg = LinearRegression()


    reg.fit(X_train, y_train)
    y_pred = reg.predict(X_test)

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

    print(f"R² на тесте: {r2:.4f}")
    print(f"MAE: {mae:.2f}")

    fig, axes = plt.subplots(1, 2, figsize=(15, 5))

    sns.scatterplot(x=cars["horsepower"], y=cars["price"], alpha=0.6, ax=axes[0])
    sns.regplot(x=cars["horsepower"], y=cars["price"], scatter=False, color="red", ax=axes[0])
    axes[0].set_xlabel("Horsepower")
    axes[0].set_ylabel("Price")
    axes[0].set_title("Зависимость цены от мощности двигателя")

    axes[1].scatter(y_test, y_pred, alpha=0.6)
    axes[1].plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
    axes[1].set_xlabel('Horsepower')
    axes[1].set_ylabel('Price')
    axes[1].set_title('Предсказания vs Реальные значения')

    plt.tight_layout()
    plt.show()

    return reg, scaler, features


# --------------------------------
# 2. КЛАССИФИКАЦИЯ: Adult Census Income
# --------------------------------
def enhanced_classification_analysis():
    import pandas as pd
    import numpy as np
    import os
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.preprocessing import StandardScaler
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LogisticRegression
    from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
    from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix

    print("\n=== АНАЛИЗ: Adult Census Income ===")

    current_dir = os.path.dirname(os.path.abspath(__file__))
    project_root = os.path.abspath(os.path.join(current_dir, '..', '..', '..', '..', '..'))
    file_path_adult = os.path.join(project_root, 'adult.csv')

    cols = [
        "age", "workclass", "fnlwgt", "education", "education_num",
        "marital_status", "occupation", "relationship", "race", "sex",
```

```
    "capital_gain", "capital_loss", "hours_per_week", "native_country", "income"
]

try:
    adult = pd.read_csv(file_path_adult, header=None, names=cols, na_values=" ?", skipinitialspace=True)
except FileNotFoundError:
    print(f"Файл не найден: {file_path_adult}")
    return None, None, None, None

adult.replace('?', np.nan, inplace=True)
adult.dropna(inplace=True)

numeric_columns = ['age', 'fnlwgt', 'education_num', 'capital_gain', 'capital_loss', 'hours_per_week']
for col in numeric_columns:
    adult[col] = pd.to_numeric(adult[col], errors='coerce')
adult.dropna(inplace=True)

y = (adult["income"].str.strip() == ">50K").astype(int)

# Преобразования признаков
adult['age_group'] = adult['age'].apply(lambda x:
                                        '18-25' if x <= 25 else
                                        '26-35' if x <= 35 else
                                        '36-45' if x <= 45 else
                                        '46-55' if x <= 55 else
                                        '56-65' if x <= 65 else '65+')
adult['hours_category'] = adult['hours_per_week'].apply(lambda x:
                                                'part-time' if x <= 30 else
                                                'full-time' if x <= 40 else
                                                'overtime' if x <= 50 else 'excessive')
adult['capital_net'] = adult['capital_gain'] - adult['capital_loss']
adult['is_married'] = adult['marital_status'].str.contains('Married').astype(int)
adult['is_white_collar'] = adult['occupation'].isin(['Prof-specialty', 'Exec-managerial']).astype(int)

categorical_cols = ['workclass', 'education', 'marital_status', 'occupation',
                    'relationship', 'race', 'sex', 'native_country', 'age_group', 'hours_category']
categorical_cols = [col for col in categorical_cols if col in adult.columns]

numerical_cols = ['age', 'education_num', 'capital_gain', 'capital_loss',
                  'hours_per_week', 'capital_net', 'is_married', 'is_white_collar']
numerical_cols = [col for col in numerical_cols if col in adult.columns]

X_encoded = pd.get_dummies(adult[categorical_cols], drop_first=True)
X_numerical = adult[numerical_cols]
X = pd.concat([X_numerical, X_encoded], axis=1)

if X.empty or len(y) == 0:
    print("Ошибка: Нет данных для обучения")
    return None, None, None, None

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

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

# Обучение моделей
log_reg = LogisticRegression(max_iter=1000, class_weight='balanced', C=1.0, solver='liblinear')
log_reg.fit(X_train, y_train)
y_pred_log = log_reg.predict(X_test)

rf = RandomForestClassifier(n_estimators=100, class_weight='balanced', random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

gb = GradientBoostingClassifier(n_estimators=100, random_state=42)
gb.fit(X_train, y_train)
y_pred_gb = gb.predict(X_test)

models = {
    'Logistic Regression': y_pred_log,
    'Random Forest': y_pred_rf,
    'Gradient Boosting': y_pred_gb
}

# Сравнение моделей только по Accuracy, Precision, Recall
results = []
```

```
    for name, y_pred in models.items():
        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred, zero_division=0)
        recall = recall_score(y_test, y_pred, zero_division=0)
        results.append({
            'Model': name,
            'Accuracy': accuracy,
            'Precision': precision,
            'Recall': recall
        })

    results_df = pd.DataFrame(results)
    print("\nСравнение моделей:")
    print(results_df.round(4))

    # Выбор лучшей модели по Accuracy
    best_model_name = results_df.loc[results_df['Accuracy'].idxmax(), 'Model']
    best_y_pred = models[best_model_name]

    # Матрица ошибок
    cm = confusion_matrix(y_test, best_y_pred)
    plt.figure(figsize=(6,5))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
                xticklabels=["<=50K", ">50K"], yticklabels=["<=50K", ">50K"])
    plt.xlabel("Предсказано")
    plt.ylabel("Истинное значение")
    plt.title(f"Матрица ошибок ({best_model_name})")
    plt.show()

    return log_reg, scaler, X.columns, results_df

# --------------------------------
# ЗАПУСК АНАЛИЗА
# --------------------------------
if __name__ == "__main__":
    print("Запуск улучшенного ML анализа...")

    print("\n" + "=" * 60)
    reg_result = enhanced_regression_analysis()
    print("\n" + "=" * 60)
    clf_result = enhanced_classification_analysis()

    print("\n" + "=" * 60)
    print("Анализ завершен!")
    print("=" * 60)
```

График регрессия:



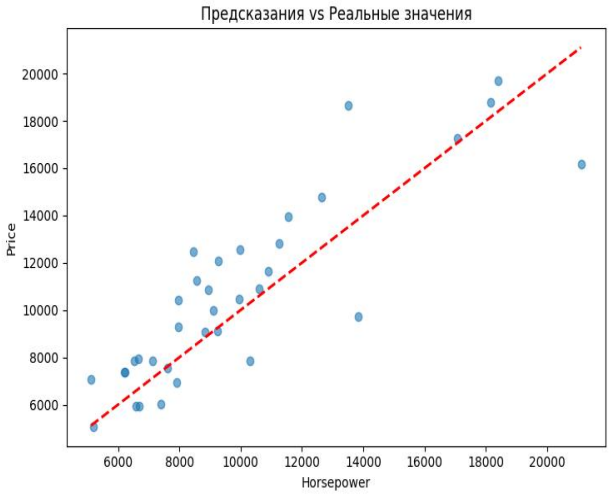
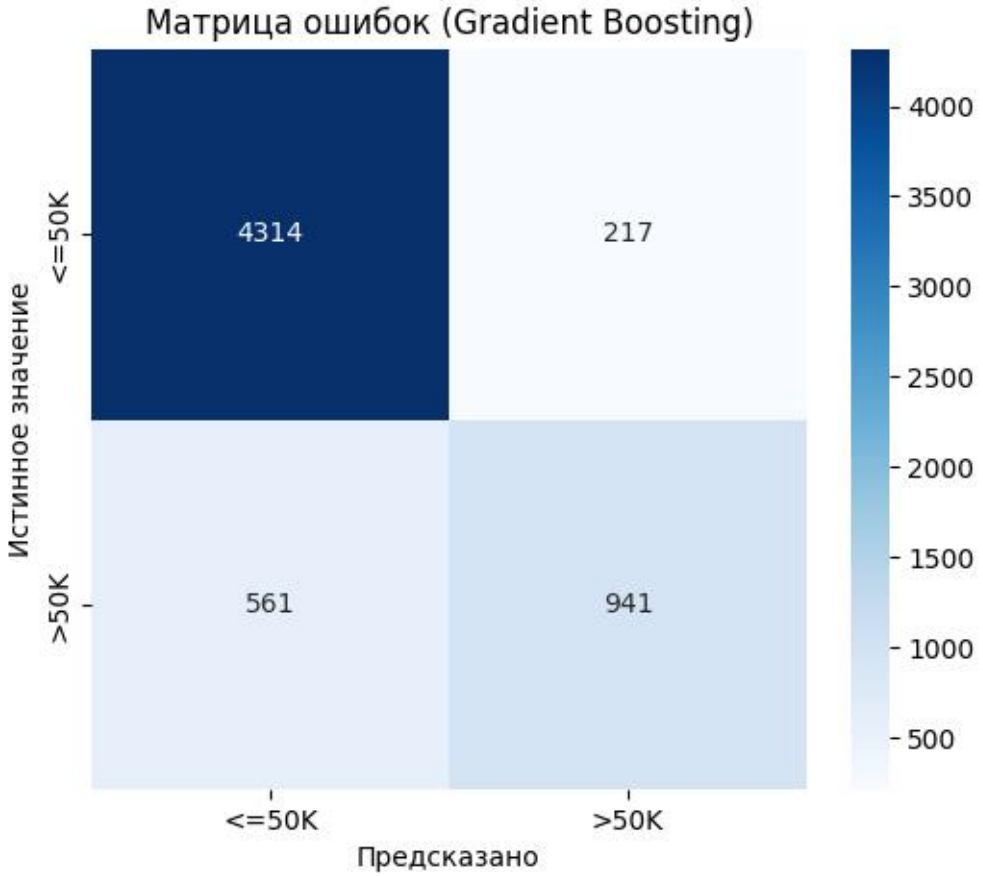


График классификация:



Результаты:

```
=== УЛУЧШЕННЫЙ АНАЛИЗ: Car Price Prediction ===
R² на тесте: 0.7002
MAE: 1629.85
```

```
Сравнение моделей:
                Model  Accuracy  Precision  Recall
0  Logistic Regression    0.8188     0.5942  0.8589
1        Random Forest    0.8520     0.7221  0.6591
2    Gradient Boosting    0.8710     0.8126  0.6265
```

Вывод: в результате выполнения данной лабораторной работы изучли применение линейной и логистической регрессии для решения практических задач. Научились обучать модели, оценивать их качество с помощью соответствующих метрик и интерпретировать результаты.