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»**

Выполнил:
Студент 3 курса
Группы АС-65
Макарский А. Э.
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 10

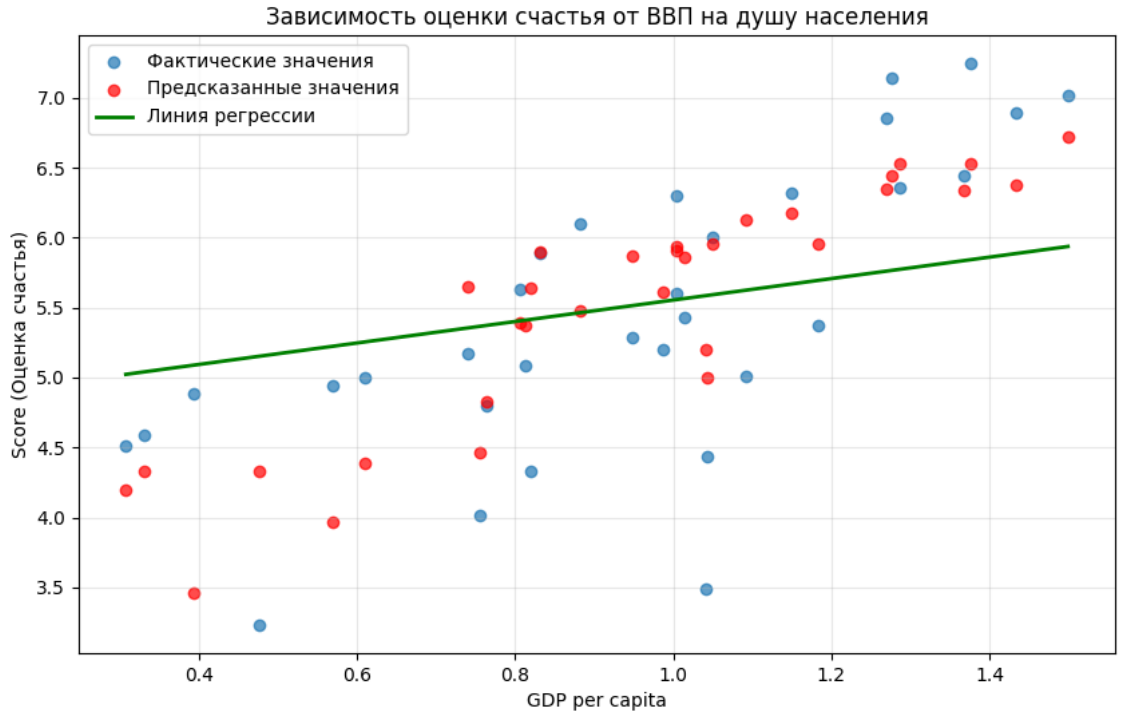
Ход работы:

Регрессия (Прогнозирование счастья в странах)

1. World Happiness Report
2. Предсказать оценку счастья (Score)
3. Задания:
   - загрузите данные. В качестве признаков используйте GDP per capita, Social support, Healthy life expectancy;
   - обучите модель линейной регрессии;
   - рассчитайте MSE и R2;
   - визуализируйте зависимость Score от GDP per capita с
   - линией регрессии.

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


df_happiness = pd.read_csv('World_Happiness_Report.csv')  # замените на ваш путь к
файлу

features = ['GDP per capita', 'Social support', 'Healthy life expectancy']
X = df_happiness[features]
y = df_happiness['Score']

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

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

y_pred = model_reg.predict(X_test)

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

print("=== РЕГРЕССИЯ: World Happiness Report ===")
print(f"MSE: {mse:.4f}")
print(f"R2 Score: {r2:.4f}")
print(f"Коэффициенты: {model_reg.coef_}")
print(f"Intercept: {model_reg.intercept_:.4f}")

plt.figure(figsize=(10, 6))
plt.scatter(X_test['GDP per capita'], y_test, alpha=0.7, label='Фактические значения')
plt.scatter(X_test['GDP per capita'], y_pred, alpha=0.7, label='Предсказанные
значения', color='red')
```

```
x_line = np.linspace(X_test['GDP per capita'].min(), X_test['GDP per capita'].max(),
100)

temp_X = X_test.copy()
temp_X['Social support'] = X_test['Social support'].mean()
temp_X['Healthy life expectancy'] = X_test['Healthy life expectancy'].mean()
y_line = model_reg.predict(pd.DataFrame({
    'GDP per capita': x_line,
    'Social support': [X_test['Social support'].mean()] * 100,
    'Healthy life expectancy': [X_test['Healthy life expectancy'].mean()] * 100
}))

plt.plot(x_line, y_line, color='green', linewidth=2, label='Линия регрессии')
plt.xlabel('GDP per capita')
plt.ylabel('Score (Оценка счастья)')
plt.title('Зависимость оценки счастья от ВВП на душу населения')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```

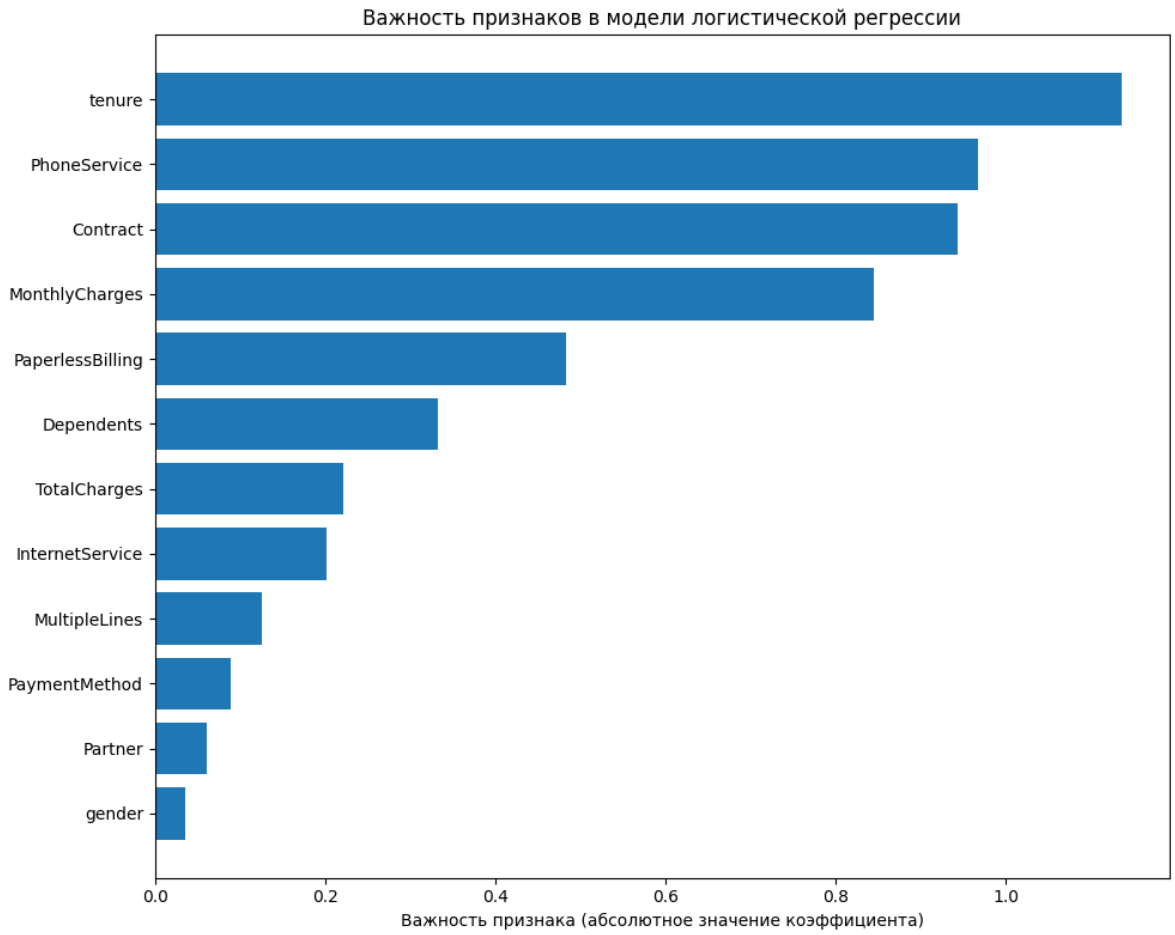


Классификация (Прогнозирование оттока клиентов)

1. Telco Customer Churn
2. Предсказать, уйдёт ли клиент (Churn = 'Yes')
3. Задания:
   - загрузите данные, обработайте категориальные признаки;
   - обучите модель логистической регрессии;
   - рассчитайте Accuracy, Precision и Recall для класса 'Yes';
   - постройте матрицу ошибок.

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

df_churn = pd.read_csv('Telco-Customer-Churn.csv')

df_churn = df_churn.drop('customerID', axis=1)
df_churn['TotalCharges'] = pd.to_numeric(df_churn['TotalCharges'], errors='coerce')
df_churn['TotalCharges'] =
df_churn['TotalCharges'].fillna(df_churn['TotalCharges'].median())

categorical_features = ['gender', 'Partner', 'Dependents', 'PhoneService',
'MultipleLines',
                        'InternetService', 'Contract', 'PaperlessBilling',
'PaymentMethod']
numerical_features = ['tenure', 'MonthlyCharges', 'TotalCharges']

label_encoders = {}
for col in categorical_features:
    le = LabelEncoder()
    df_churn[col] = le.fit_transform(df_churn[col].astype(str))
    label_encoders[col] = le

X = df_churn[categorical_features + numerical_features]
y = df_churn['Churn'].map({'Yes': 1, 'No': 0})

scaler = StandardScaler()
X[numerical_features] = scaler.fit_transform(X[numerical_features])

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

model_clf = LogisticRegression(random_state=42, max_iter=1000)
model_clf.fit(X_train, y_train)

y_pred = model_clf.predict(X_test)
y_pred_proba = model_clf.predict_proba(X_test)[:, 1]

# Расчет метрик
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, pos_label=1)
recall = recall_score(y_test, y_pred, pos_label=1)

print("\n=== КЛАССИФИКАЦИЯ: Telco Customer Churn ===")
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision (для 'Yes'): {precision:.4f}")
print(f"Recall (для 'Yes'): {recall:.4f}")

cm = confusion_matrix(y_test, y_pred)
print("\nМатрица ошибок:")
print(cm)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
           xticklabels=['No', 'Yes'],
           yticklabels=['No', 'Yes'])
plt.xlabel('Предсказанные значения')
plt.ylabel('Фактические значения')
plt.title('Матрица ошибок для прогнозирования оттока клиентов')
plt.show()
```

```
feature_importance = pd.DataFrame({
    'feature': X.columns,
    'importance': abs(model_clf.coef_[0])
}).sort_values('importance', ascending=True)

plt.figure(figsize=(10, 8))
plt.barh(feature_importance['feature'], feature_importance['importance'])
plt.xlabel('Важность признака (абсолютное значение коэффициента)')
plt.title('Важность признаков в модели логистической регрессии')
plt.tight_layout()
plt.show()
```



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