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

Лабораторная работа №2
По дисциплине «ОМО»
Отыскание экстремумов функции одной переменной.

Выполнил:
Студент 3-го курса
Группы АС-65
Егоренков Н. Д.
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 5

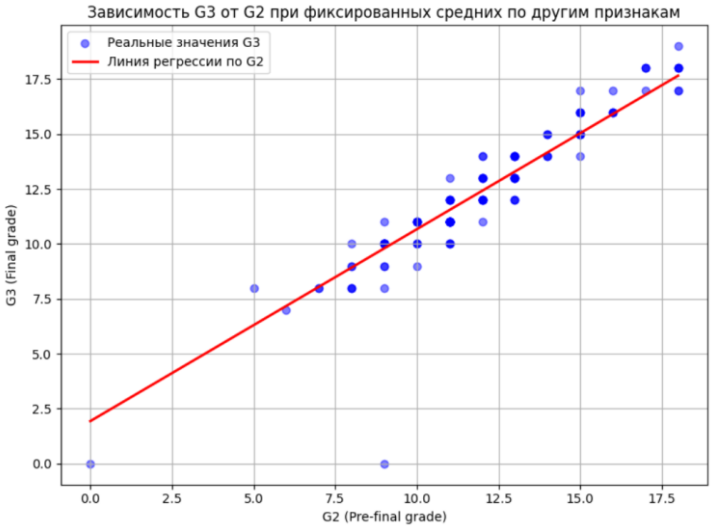
Ход работы

- **Регрессия (Предсказание итоговой оценки студента)**
  1. Student Performance
  2. Предсказать итоговую оценку (G3)
  3. **Задания:**
     - загрузите данные, выбрав в качестве признаков studytime, failures, G1, G2;
     - обучите модель **линейной регрессии**;
     - рассчитайте **MAE** и **R^2**;
     - визуализируйте зависимость G3 от G2 (оценка за предыдущий период) с линией регрессии.

Код:

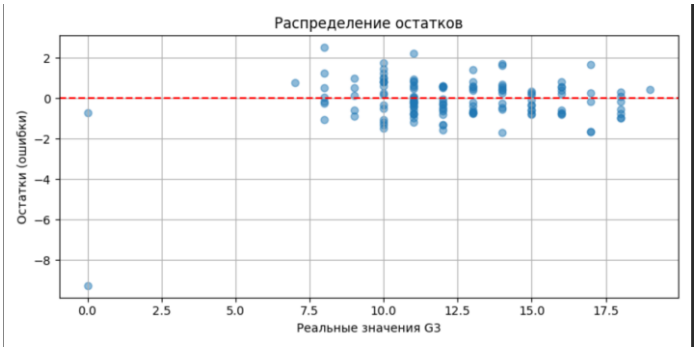
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, r2_score
import seaborn as sns

df = pd.read_csv('student-por.csv', sep=';')

required_columns = ['studytime', 'failures', 'G1', 'G2', 'G3']
missing_columns = [col for col in required_columns if col not in df.columns]
if missing_columns:
    raise ValueError(f"Отсутствуют необходимые столбцы: {missing_columns}")

print("Количество пропущенных значений по столбцам:\n", df.isnull().sum())

X = df[['studytime', 'failures', 'G1', 'G2']]
y = df['G3']

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

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

y_pred = model.predict(X_test)

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

```
print(f"Средняя абсолютная ошибка (MAE): {mae:.2f}")
print(f"Коэффициент детерминации (R^2): {r2:.2f}")

coefficients = pd.Series(model.coef_, index=X.columns)
print("Коэффициенты модели:\n", coefficients)
print("Важность признаков по абсолютным значениям:\n",
coefficients.abs().sort_values(ascending=False))

corr_matrix = df[['studytime', 'failures', 'G1', 'G2', 'G3']].corr()
print("Корреляционная матрица:\n", corr_matrix)

G2_min, G2_max = X_test['G2'].min(), X_test['G2'].max()
G2_range = np.linspace(G2_min, G2_max, 100)

mean_studytime = X_test['studytime'].mean()
mean_failures = X_test['failures'].mean()
mean_G1 = X_test['G1'].mean()

X_line = pd.DataFrame({
    'studytime': [mean_studytime] * 100,
    'failures': [mean_failures] * 100,
    'G1': [mean_G1] * 100,
    'G2': G2_range
})

G3_pred_line = model.predict(X_line)

plt.figure(figsize=(8, 6))
plt.scatter(X_test['G2'], y_test, color='blue', alpha=0.5, label='Реальные значения
G3')
plt.plot(G2_range, G3_pred_line, color='red', linewidth=2, label='Линия регрессии по
G2')

plt.xlabel('G2 (Pre-final grade)')
plt.ylabel('G3 (Final grade)')
plt.title('Зависимость G3 от G2 при фиксированных средних по другим признакам')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.savefig("regression_dependence.png")
```




```
residuals = y_test - y_pred
```

```
print("Среднее значение остатков:", residuals.mean())
print("Стандартное отклонение остатков:", residuals.std())

predictions_df = pd.DataFrame({
    'Реальное G3': y_test.values,
    'Предсказанное G3': y_pred
})
print("Первые 10 предсказаний:\n", predictions_df.head(10))

plt.figure(figsize=(8, 4))
plt.scatter(y_test, residuals, alpha=0.5)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('Реальные значения G3')
plt.ylabel('Остатки (ошибки)')
plt.title('Распределение остатков')
plt.grid(True)
plt.tight_layout()
plt.savefig("regression_residuals.png")
```




```
print("Описание остатков:\n", residuals.describe())

sample_students = pd.DataFrame({
    'studytime': [2, 3],
    'failures': [0, 1],
    'G1': [10, 12],
    'G2': [10, 14]
})
sample_predictions = model.predict(sample_students)
print("Прогноз для гипотетических студентов:\n", sample_students)
print("Предсказанные G3:\n", sample_predictions)

print("\nФункционал завершен. Проверьте изображение 'regression_dependence.png' для визуализации.")
```

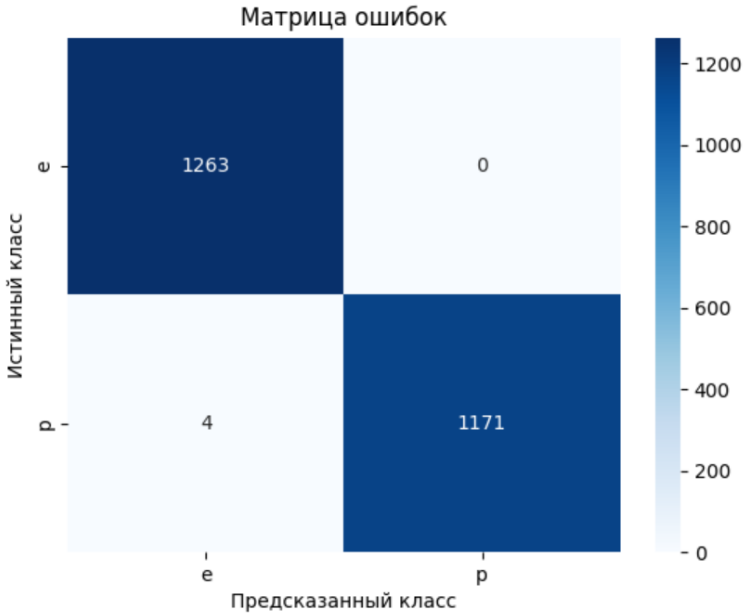
```
Средняя абсолютная ошибка (MAE): 0.74
Коэффициент детерминации (R^2): 0.86
Коэффициенты модели:
 studytime    0.104495
failures    -0.140904
G1           0.175813
G2           0.873408
dtype: float64
Важность признаков по абсолютным значениям:
 G2           0.873408
G1           0.175813
failures     0.140904
studytime    0.104495
dtype: float64
Корреляционная матрица:
            studytime  failures        G1        G2        G3
studytime    1.000000 -0.147441  0.260875  0.240498  0.249789
failures    -0.147441  1.000000 -0.384210 -0.385782 -0.393316
G1           0.260875 -0.384210  1.000000  0.864982  0.826387
G2           0.240498 -0.385782  0.864982  1.000000  0.918548
G3           0.249789 -0.393316  0.826387  0.918548  1.000000
```

- **Классификация (Определение съедобности грибов)**

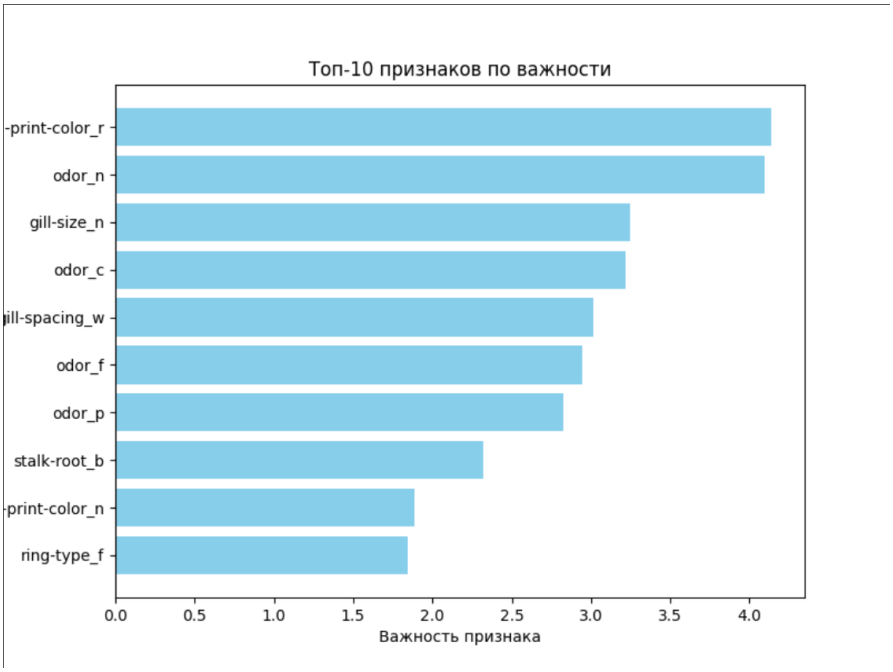
1. Mushroom Classification
2. Определить, является ли гриб ядовитым (class = 'p') или съедобным ('e')
3. **Задания:**
   - загрузите данные. Все признаки являются категориальными, примените к ним One-Hot Encoding;
   - обучите модель **логистической регрессии**;
   - рассчитайте **Accuracy**, **Precision** и **Recall**;
   - постройте **матрицу ошибок**.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, classification_report, roc_auc_score, roc_curve
import matplotlib.pyplot as plt
import seaborn as sns


df = pd.read_csv('mushrooms.csv')

X = df.drop('class', axis=1)
y = df['class']


le = LabelEncoder()
y_encoded = le.fit_transform(y)

onehot_encoder = OneHotEncoder(drop='first', sparse_output=False)
X_encoded = onehot_encoder.fit_transform(X)

feature_names = []
for i, col in enumerate(X.columns):
    categories = onehot_encoder.categories_[i][1:]
    for cat in categories:
        feature_names.append(f"{col}_{cat}")

print(f"\nКоличество признаков после One-Hot Encoding: {X_encoded.shape[1]}")

X_train, X_test, y_train, y_test = train_test_split(
    X_encoded, y_encoded, test_size=0.3, random_state=42, stratify=y_encoded
)

model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

```
specificity = ...

cm = confusion_matrix(y_test, y_pred)

print(f"Точность: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
print(f"Специфичность: {specificity}")

print("\nДетальный отчет:")
print(classification_report(y_test, y_pred, target_names=['Съедобный', 'Ядовитый']))

plt.figure()
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['e', 'p'],
yticklabels=['e', 'p'])
plt.title('Матрица ошибок')
plt.xlabel('Предсказанный класс')
plt.ylabel('Истинный класс')
plt.savefig('confusion_matrix.png')
```




```
feature_importance = pd.DataFrame({
    'feature': feature_names,
    'importance': abs(model.coef_[0])
}).sort_values('importance', ascending=False).head(10)

plt.figure(figsize=(8, 6))
plt.barh(feature_importance['feature'], feature_importance['importance'],
color='skyblue')
plt.xlabel('Важность признака')
plt.title('Топ-10 признаков по важности')
plt.gca().invert_yaxis()
plt.savefig('feature_importance.png')
```




```
sample_indices = np.random.choice(len(X_test), 5, replace=False)
for idx in sample_indices:
    print(f"Истинный класс: {y_test[idx]}, Предсказание: {y_pred[idx]}, Вероятность: {y_pred_proba[idx]:.4f}")
```

```
Количество признаков после One-Hot Encoding: 95
Точность: 0.9984
Precision: 1.0000
Recall: 0.9966
F1-Score: 0.9983
Специфичность: Ellipsis
```

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