МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №2

Специальность АС-65

Выполнил:
Д. А.Чмель,
студент группы АС-65
Проверил
А. А. Крощенко,
ст. преп. кафедрыИИТ,
« » _____ 2025 г.

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 3

Задание **1. Регрессия (Прогнозирование расхода топлива)**

Задание 1.1 Загрузка данных, обработка пропусков и категориальных признаков;

```
#загрузка данных
print("1. Загрузка данных")
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data'
column_names = ['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'model year', 'origin', 'car name']
df_auto = pd.read_csv(url, delim_whitespace=True, names=column_names, na_values='?')

#просмотр данных
print(df_auto.head())
print(df_auto.info())
```

```
#заменяем пропуски медианой
print("\n2. Обработка пропусков")
print("Пропуски в horsepower:", df_auto['horsepower'].isnull().sum())
imputer = SimpleImputer(strategy='median')
df_auto['horsepower'] = imputer.fit_transform(df_auto[['horsepower']])
```

```
2. Обработка пропусков
Пропуски в horsepower: 6
```

Задание 1.2 Обучение модели линейной регрессии, используя в качестве признаков cylinders, horsepower, weight;

```
print("\n3. Подготовка данных для модели")

# Выбор признаков и целевой переменной
X = df_auto[['cylinders', 'horsepower', 'weight']]
y = df_auto['mpg']

print(f"Признаки: {X.columns.tolist()}")
print(f"Целевая переменная: mpg")
print(f"Размер X: {X.shape}")

# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, shuffle=True)
print(f"Обучающая выборка: {X_train.shape[0]} samples")
print(f"Тестовая выборка: {X_test.shape[0]} samples")
```

```
3. Подготовка данных для модели
Признаки: ['cylinders', 'horsepower', 'weight']
Целевая переменная: mpg
Размер X: (398, 3)
Обучающая выборка: 318 samples
Тестовая выборка: 80 samples
```

```
print("\n4. Обучение модели линейной регрессии")

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

print(f"Коэффициенты: {model.coef_}")
print(f"Intercept: {model.intercept_:.2f}")
```

```
4. Обучение модели линейной регрессии
Коэффициенты: [-0.36692159 -0.04342671 -0.00551446]
Intercept: 46.45
```

Задание 1.3 рассчет MSE и R^2;

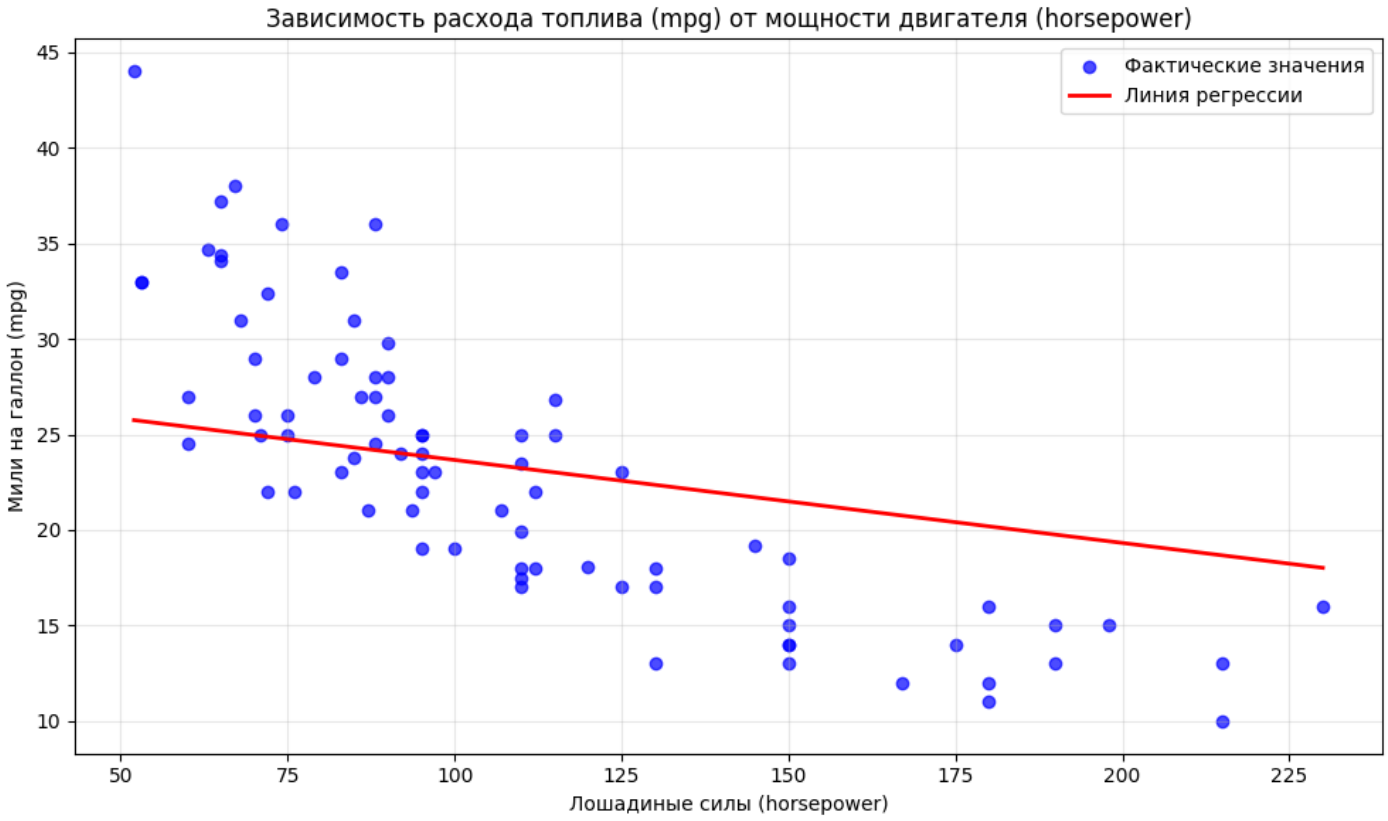
```
print("\n5. Оценка модели")

# Предсказания
y_pred = model.predict(X_test)

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

print(f"MSE (Mean Squared Error): {mse:.2f}")
print(f"R² (Coefficient of Determination): {r2:.2f}")
```

```
5. Оценка модели
MSE (Mean Squared Error): 14.54
R² (Coefficient of Determination): 0.73
```

Задание 1.4 Визуализация зависимость mpg от horsepower с линией регрессии

```
print("\n6. Визуализация зависимости mpg от horsepower")

plt.figure(figsize=(10, 6))

# Диаграмма рассеяния
plt.scatter(X_test['horsepower'], y_test, alpha=0.7, color='blue', label='Фактические значения')

# Линия регрессии
# Для визуализации используем предсказания только по horsepower
horsepower_range = np.linspace(X_test['horsepower'].min(), X_test['horsepower'].max(), 100)
# Создаем фиктивные данные для предсказания (средние значения других признаков)
cylinders_mean = X_test['cylinders'].mean()
weight_mean = X_test['weight'].mean()

# Предсказания для линии регрессии
line_data = pd.DataFrame({
    'cylinders': [cylinders_mean] * 100,
    'horsepower': horsepower_range,
    'weight': [weight_mean] * 100
})
regression_line = model.predict(line_data)
```

```
plt.plot(horsepower_range, regression_line, color='red', linewidth=2, label='Линия регрессии')

plt.xlabel('Лошадиные силы (horsepower)')
plt.ylabel('Мили на галлон (mpg)')
plt.title('Зависимость расхода топлива (mpg) от мощности двигателя (horsepower)')
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```



Задание 2. Классификация (Диагностика диабета)

Задание 2.1 Загрузка данных, выполнение стандартизацию признаков

```
print("\n2. Подготовка данных для модели")

# Разделение на признаки и целевую переменную
X = df_diabetes.drop('outcome', axis=1)
y = df_diabetes['outcome']

print(f"Признаки: {X.columns.tolist()}")
print(f"Целевая переменная: outcome (0 - нет диабета, 1 - диабет)")
print(f"Размер X: {X.shape}")

# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y,  # Сохраняем распределение классов
    shuffle=True
)
```

```
2. Подготовка данных для модели
Признаки: ['pregnant', 'glucose', 'pressure', 'skin_thickness', 'insulin', 'bmi', 'pedigree', 'age']
Целевая переменная: outcome (0 - нет диабета, 1 - диабет)
Размер X: (768, 8)
```

Задание 2.2 Обучение модели логистической регрессии

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

print("\n3. Обучение модели логистической регрессии")
model = LogisticRegression(random_state=42, max_iter=1000)
model.fit(X_train_scaled, y_train)
```

```
3. Обучение модели логистической регрессии
LogisticRegression
LogisticRegression(max_iter=1000, random_state=42)
```

Задание 2.3 Рассчет Accuracy, Precision и Recall

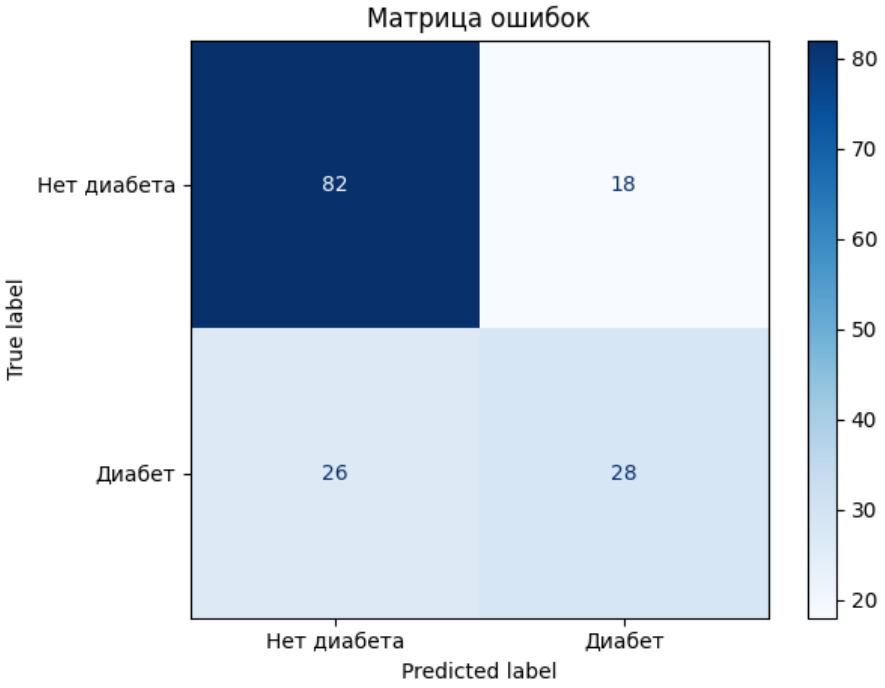
```
print("\n4. Оценка модели...")
y_pred = model.predict(X_test_scaled)

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:.3f}")
print(f"Precision: {precision:.3f}")
print(f"Recall:    {recall:.3f}")
```

```
4. Оценка модели...
Accuracy:  0.714
Precision: 0.609
Recall:    0.519
```

Задание 2.4 Построение матрицы ошибок

```
5. Матрица ошибок

Матрица ошибок:
TN: 82 | FP: 18
FN: 26 | TP: 28

Анализ ошибок:
Ложноположительные (FP): 18 пациентов
  - Здоровым ошибочно поставили диабет
Ложноотрицательные (FN): 26 пациентов
  - Больным диабетом ошибочно сказали, что здоровы
```

```python
print("\n5. Матрица ошибок")
cm = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = cm.ravel()

print(f"\nМатрица ошибок:")
print(f"TN: {tn} | FP: {fp}")
print(f"FN: {fn} | TP: {tp}")

print(f"\nАнализ ошибок:")
print(f"Ложноположительные (FP): {fp} пациентов")
print(f"  - Здоровым ошибочно поставили диабет")
print(f"Ложноотрицательные (FN): {fn} пациентов")
print(f"  - Больным диабетом ошибочно сказали, что здоровы")

# Визуализация матрицы ошибок
plt.figure(figsize=(6, 5))
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Нет диабета', 'Диабет'])
disp.plot(cmap='Blues', values_format='d')
plt.title('Матрица ошибок')
plt.tight_layout()
plt.show()
```



Выводы:

В ходе лабораторной работы были изучены и реализованы методы линейной и логистической регрессии для решения задач прогнозирования и классификации.

1) Линейная регрессия была применена для предсказания расхода топлива автомобиля по признакам cylinders, horsepower и weight. После обучения модели были рассчитаны метрики MSE и R^2, которые показали удовлетворительное качество аппроксимации. График зависимости mpg от horsepower с линией регрессии подтвердил наличие обратной зависимости между мощностью двигателя и экономичностью автомобиля.

2) Логистическая регрессия использовалась для диагностики наличия диабета по медицинским показателям. После стандартизации признаков модель показала хорошие значения Accuracy, Precision и Recall, что свидетельствует о способности алгоритма корректно классифицировать случаи заболевания. Матрица ошибок визуализировала распределение правильных и ошибочных предсказаний.