МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра теории вероятностей и математической статистики

ОТЧЕТ по лабораторным работам №3, 4, 5

Константинова Николая Андреевича
студента 3 курса, 8 группы

Преподаватель
Полузёров Тимофей Дмитриевич

Минск, 2025

Содержание

Лабораторная работа №3

Кластерный анализ данных

Для кластерного анализа будем использовать уже нормализованные данные, полученные в предыдущих лабораторных работах.

Добавим кластеры к данным и вычислим средние кредитные показатели по ним:

```
X_clustered = X.copy()
X_clustered['Cluster'] = clusters
X_clustered['Credit_Score'] = credit_score_normalized_cluster

cluster_means = X_clustered.groupby('Cluster')['Credit_Score'].mean().sort_values(ascending=False)
```

Вывод:

```
СРЕДНИЕ ЗНАЧЕНИЯ CREDIT SCORE ПО КЛАСТЕРАМ:
Cluster
3    74.11
2    64.17
0    49.95
1    36.31
```

Перенумеруем кластеры в порядке убывания для удобства:

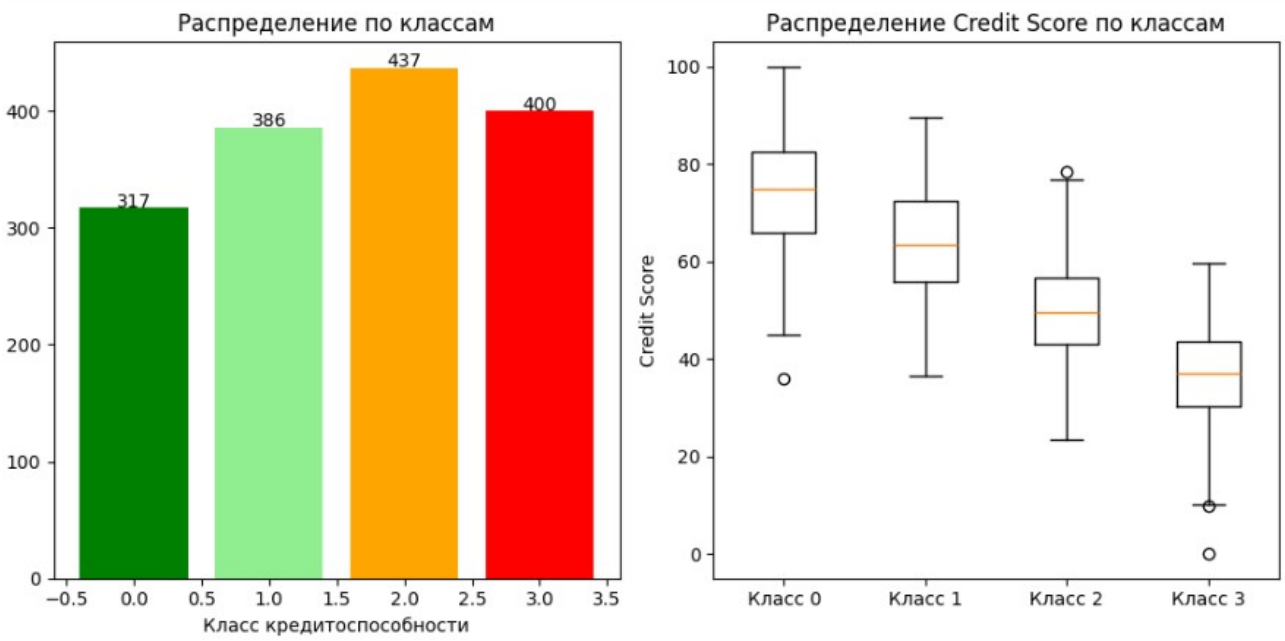
```
cluster_mapping = {old: new for new, old in enumerate(cluster_means.index)}
X_clustered['Cluster_Renumbered'] = X_clustered['Cluster'].map(cluster_mapping)

print(f"\Переномеровка кластеров:")
for old_cluster, new_cluster in cluster_mapping.items():
    mean_score = cluster_means[old_cluster]
    print(f"Кластер {old_cluster} → Класс {new_cluster} (средний score: {mean_score:.2f})")
```

Вывод:

Cluster_Renumbered	count	mean	std	min	max
0	317	74.11	11.90	36.06	100.00
1	386	64.17	10.46	36.53	89.61
2	437	49.95	9.43	23.59	78.52
3	400	36.31	9.99	0.00	59.66

Как видно из вычислений и построенных графиков, малочисленными являются кластеры 0 и 1, многочисленными — 2 и 3.



Лабораторная работа №4

Дискриминантный анализ данных

Линейный дискриминантный анализ (LDA) проводим с помощью встроенной функции LinearDiscriminantAnalysis(): сначала обучаем модель, а затем предсказываем классы, построив вектор классификаций на всей выборке.

```
lda = LinearDiscriminantAnalysis()
lda.fit(X, y)
y_pred_lda_full = lda.predict(X)
```

Строим матрицу ошибок и вычисляем точность:

```
cm_lda_full = confusion_matrix(y, y_pred_lda_full)
accuracy_lda_full = accuracy_score(y, y_pred_lda_full)

print("Матрица ошибок:")
print(cm_lda_full)
print(f"\nТочность: {accuracy_lda_full:.4f}")
```

Вывод:

```
Матрица ошибок:
[[282  16  19   0]
 [  7 345  24  10]
 [  0   9 421   7]
 [  0  10  25 365]]

Точность: 0.9175
```

Вычисляем условные вероятности ошибок:

```python
for i in range(len(cm_lda_full)):
    total = cm_lda_full[i].sum()
    correct = cm_lda_full[i, i]
    error_rate = (total - correct) / total if total > 0 else 0
    print(f"Класс {i}: {error_rate:.4f}")
```

Чтобы вычислить безусловную вероятность ошибки, от единицы отнимаем точность классификации:

```python
print(f"Безусловная вероятность ошибки: {1 - accuracy_lda_full:.4f}")
```

Вывод:

```
УСЛОВНЫЕ ВЕРОЯТНОСТИ ОШИБОК:
Класс 0: 0.1104
Класс 1: 0.1062
Класс 2: 0.0366
Класс 3: 0.0875
БЕЗУСЛОВНАЯ ВЕРОЯТНОСТЬ ОШИБКИ: 0.0825
```

Режим экзамена (10%)

Для режима экзамена берём выборку предварительно исключённых наблюдений из различных классов (10%)

```
X_train, X_test, y_train, y_test = [], [], [], []

for class_label in sorted(y.unique()):
    class_indices = y[y == class_label].index
    n_test = max(1, int(0.1 * len(class_indices)))  # минимум 1 наблюдение

    test_indices = np.random.choice(class_indices, size=n_test, replace=False)
    train_indices = [idx for idx in class_indices if idx not in test_indices]

    X_train.append(X.loc[train_indices])
    X_test.append(X.loc[test_indices])
    y_train.append(y.loc[train_indices])
    y_test.append(y.loc[test_indices])

X_train = pd.concat(X_train)
X_test = pd.concat(X_test)
y_train = pd.concat(y_train)
y_test = pd.concat(y_test)
```

Как и раньше, проводим на тестовой выборке LDA:

```
lda_split = LinearDiscriminantAnalysis()
lda_split.fit(X_train, y_train)
y_pred_lda_test = lda_split.predict(X_test)
```

и считаем условные и безусловные вероятности ошибок:

```
print("\nУсловные вероятности ошибок (тест):")
for i in range(len(cm_lda_test)):
    if i < cm_lda_test.shape[0]:
        total = cm_lda_test[i].sum() if i < len(cm_lda_test) else 0
        correct = cm_lda_test[i, i] if i < len(cm_lda_test) else 0
        error_rate = (total - correct) / total if total > 0 else 0
        print(f"Класс {i}: {error_rate:.4f}")

print(f"Безусловная вероятность ошибки (тест): {1 - accuracy_lda_test:.4f}")
```

Вывод:

```
Обучающая выборка: 1388 наблюдений
Тестовая выборка: 152 наблюдений

Матрица ошибок (тест):
[[26  3  2  0]
 [ 1 32  3  2]
 [ 0  1 40  2]
 [ 0  1  2 37]]

Точность на тесте: 0.8882

УСЛОВНЫЕ ВЕРОЯТНОСТИ ОШИБОК (тест):
Класс 0: 0.1613
Класс 1: 0.1579
Класс 2: 0.0698
Класс 3: 0.0750
БЕЗУСЛОВНАЯ ВЕРОЯТНОСТЬ ОШИБКИ (тест): 0.1118
```

Лабораторная работа №5

Классификация на основе логит-модели множественного выбора

Строим логистическую регрессию:

```
logit = LogisticRegression(multi_class='multinomial', max_iter=1000, random_state=42)
```

Как и раньше, обучаем модель на всей выборке и строим матрицу ошибок.

```
logit.fit(X, y)
y_pred_logit_full = logit.predict(X)

#оценка качества
cm_logit_full = confusion_matrix(y, y_pred_logit_full)
accuracy_logit_full = accuracy_score(y, y_pred_logit_full)

print("Матрица ошибок:")
print(cm_logit_full)
print(f"\nТочность: {accuracy_logit_full:.4f}")
```

Вывод:

```
Матрица ошибок:
[[313   3   1   0]
 [  0 384   2   0]
 [  0   0 435   2]
 [  0   1   2 397]]

Точность: 0.9929
```

Режим экзамена:

```
logit_split = LogisticRegression(multi_class='multinomial', max_iter=1000, random_state=42)
logit_split.fit(X_train, y_train)
y_pred_logit_test = logit_split.predict(X_test)

#оценка качества
cm_logit_test = confusion_matrix(y_test, y_pred_logit_test)
accuracy_logit_test = accuracy_score(y_test, y_pred_logit_test)

print("Матрица ошибок (тест):")
print(cm_logit_test)
print(f"\nТочность на тесте: {accuracy_logit_test:.4f}")
```

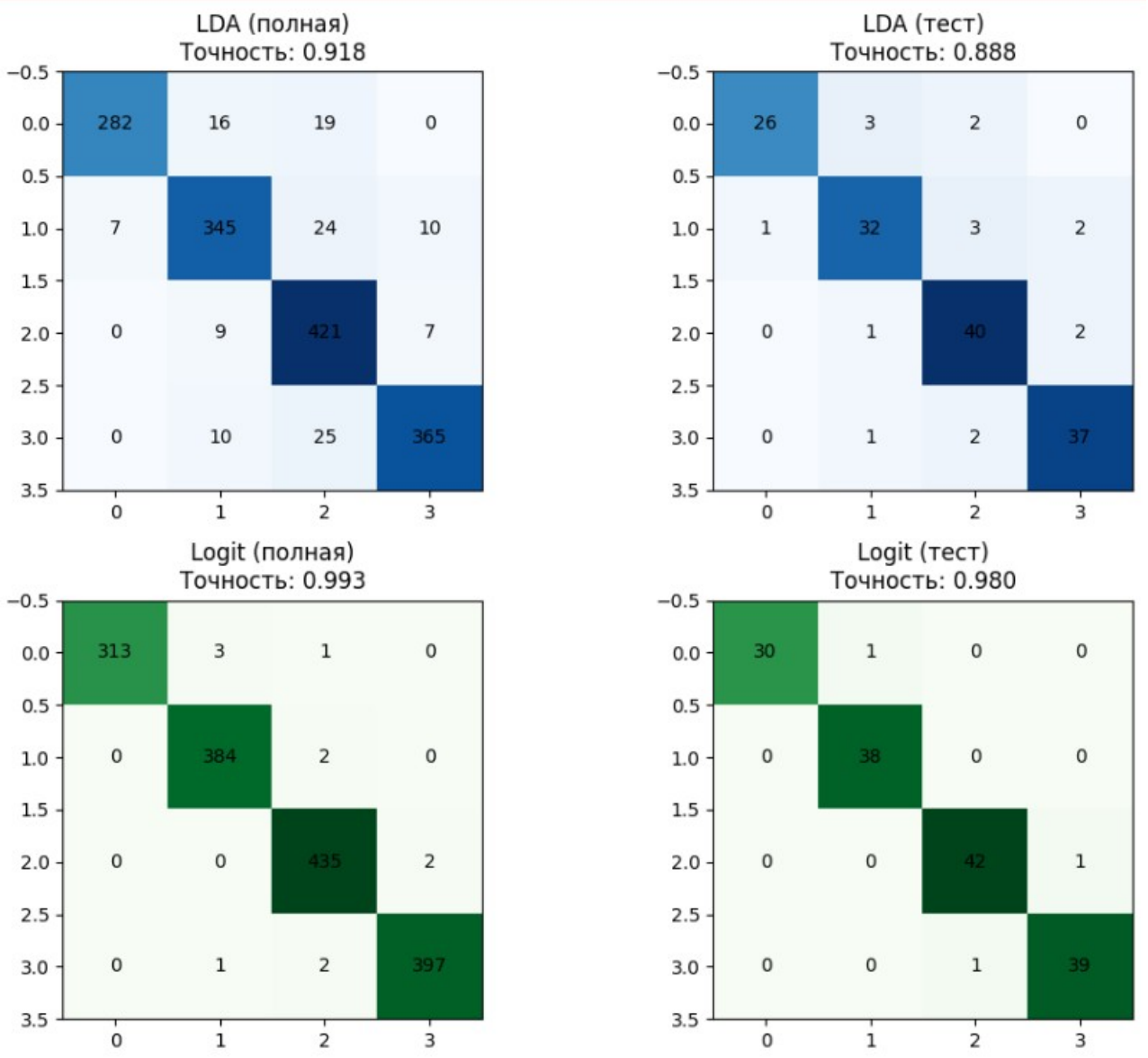
Вывод:

```
Матрица ошибок (тест):
[[30  1  0  0]
 [ 0 38  0  0]
 [ 0  0 42  1]
 [ 0  0  1 39]]

Точность на тесте: 0.9803
```

Результаты классификаций сравним с помощью графиков. Для этого используем уже полученные ранее данные:

	Метод	Точность	Ошибка
0	LDA (полная выборка)	0.9175	0.0825
1	LDA (тест)	0.8882	0.1118
2	Logit (полная выборка)	0.9929	0.0071
3	Logit (тест)	0.9803	0.0197



Из графиков и имеющихся данных видно, что Logit-модель даёт более высокую точность.