

FGBU «NMC of Oncology named after N.N. Blokhin»

Department of Medical Informatics and AI

LABORATORY WORK No.3

Radiomic Analysis of Brain Tumors

based on Multiparametric MRI

Student:

Full name

Group: _____

Variant: _____

Supervisor:

Full name

Degree: _____

Moscow, 2026

Содержание

1	Введение	2
1.1	Цель работы	2
1.2	Клинический контекст	2
1.3	Мультипараметрическая МРТ в нейровизуализации	2
1.3.1	T1-взвешенные изображения (T1)	2
1.3.2	T1 с контрастным усилением (T1c)	3
1.3.3	T2-взвешенные изображения (T2)	3
1.3.4	FLAIR (Fluid Attenuated Inversion Recovery)	3
2	Теоретическая часть	3
2.1	Стандарт BRATS и субрегионы опухоли	3
2.2	Классификация радиометрических признаков в нейровизуализации	3
2.2.1	Геометрические признаки (Shape)	3
2.2.2	Статистики первого порядка (First-order)	4
2.2.3	Текстурные признаки GLCM	4
2.3	Клиническая интерпретация радиометрических профилей	4
2.4	Методы многомерного анализа	5
2.4.1	Корреляционный анализ	5
2.4.2	Метод главных компонент (PCA)	5
2.4.3	Анализ важности признаков	5
3	Практическая часть	5
3.1	Загрузка и подготовка данных	5
3.2	Сравнение модальностей МРТ	5
3.3	Корреляционный анализ между модальностями	6
3.4	Радарная диаграмма	7
3.5	РСА-анализ	8
3.6	Анализ важности признаков	9
3.7	Комплексный анализ	10
4	Ключевые выводы	11
4.1	Сводная клиническая интерпретация	11
5	Выводы	11
6	Контрольные вопросы	12
6.1	Теоретический блок	12
6.2	Практический блок	12
6.3	Рефлексивный блок	12
7	Список литературы	12
А	Приложение А. Полный код скрипта анализа	13
А	Приложение Б. Результаты анализа	19
A.1	Статистика по ключевым признакам	19
A.2	Интерпретация результатов	20

Введение

Цель работы

Сформировать у слушателей компетенции в области мультипараметрического радиомического анализа опухолей головного мозга с использованием данных магнитно-резонансной томографии (МРТ) в рамках стандарта BRATS, а также освоить методы сравнительного анализа радиомических профилей пациентов для оценки гетерогенности и агрессивности новообразований.

Задачи работы:

1. Сравнить распределение радиометрических признаков между четырьмя модальностями МРТ (T1, T1c, T2, FLAIR).
2. Провести корреляционный анализ между модальностями.
3. Построить радарные диаграммы радиомических профилей пациентов.
4. Выполнить PCA-анализ для снижения размерности данных.
5. Оценить важность признаков для дифференцировки фенотипов опухоли.
6. Сформулировать клинические выводы на основе количественных радиомических данных.

Клинический контекст

Современная диагностика опухолей головного мозга основана на использовании мультипараметрической МРТ (mpMRI), которая включает четыре ключевые модальности, каждая из которых предоставляет уникальную информацию о биологических характеристиках новообразования.

Стандарт BRATS (Brain Tumor Segmentation):

- Международный стандарт для задач сегментации и анализа опухолей мозга.
- Включает мультипараметрические МРТ-исследования с экспертной разметкой.
- Содержит четыре ключевых субрегиона опухоли.

Клиническая значимость: Радиомический анализ позволяет не только количественно охарактеризовать биологические свойства опухоли, но и провести неинвазивную дифференциальную диагностику между различными фенотипами новообразований головного мозга, что имеет важное значение для планирования тактики лечения и прогнозирования клинических исходов.

Мультипараметрическая МРТ в нейровизуализации

T1-взвешенные изображения (T1)

Отображают анатомическую структуру мозга. Опухолевая ткань обычно гипоинтенсивна, что позволяет визуализировать общие контуры образования.

T1 с контрастным усилением (T1c)

Ключевая модальность для оценки гематоэнцефалического барьера. Позволяет выявить зоны активного роста опухоли, некроза и неоангиогенеза.

T2-взвешенные изображения (T2)

Чувствительны к содержанию воды в тканях. Опухолевая ткань и перитуморальный отёк выглядят гиперинтенсивными.

FLAIR

Модификация T2, в которой сигнал от свободной жидкости подавляется. Позволяет чётко отделить перитуморальный отёк от желудочков мозга.

Теоретическая часть

Стандарт BRATS и субрегионы опухоли

Субрегионы опухоли:

- **Некротическое ядро (NCR):** зоны некроза и кровоизлияний. Маркер высокой агрессивности.
- **Отёчная область (ED):** перитуморальный вазогенный отёк.
- **Усиливающаяся опухоль (ET):** активно растущая опухолевая ткань с нарушением ГЭБ.
- **Общая опухоль (WT):** объединение всех трёх областей.

Классификация радиометрических признаков

Геометрические признаки

Сферичность (Shape_Sphericity):

$$S = \frac{36\pi V^2}{A^3} \quad (1)$$

Низкая сферичность указывает на инфильтративный рост.

Статистики первого порядка

Средняя интенсивность:

$$\mu = \frac{1}{N} \sum_{i=1}^N X_i \quad (2)$$

Энтропия:

$$H = - \sum_k p_k \cdot \log_2(p_k) \quad (3)$$

Высокая энтропия коррелирует с некрозом и гетерогенностью.

Текстурные признаки GLCM

Контраст:

$$Contrast = \sum_i \sum_j |i - j|^2 \cdot P(i, j) \quad (4)$$

Корреляция:

$$Correlation = \frac{\sum_i \sum_j (i - \mu_i)(j - \mu_j)P(i, j)}{\sigma_i \sigma_j} \quad (5)$$

Клиническая интерпретация

Таблица 1: Клиническая интерпретация радиомических признаков

Признак	Интерпретация
Высокая энтропия в T1c	Активное контрастное усиление, высокая степень злокачественности
Высокий контраст	Неоднородность ткани, некротические зоны
Высокая дисперсия в T2/FLAIR	Выраженный перитуморальный отёк
Низкая сферичность	Инфильтративный рост, характерен для глиобластомы

Практическая часть

Data Loading

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from sklearn.preprocessing import StandardScaler
6 from sklearn.decomposition import PCA
7 from sklearn.ensemble import RandomForestClassifier
8 import warnings
9 warnings.filterwarnings('ignore')
10
11 # Load data
12 df = pd.read_csv('combined_multichannel_features.csv')
13
14 # View data
15 print("Patients:", df['patient_id'].unique())
16 print("Shape:", df.shape)
```

Modality Comparison

```
1 def plot_modality_comparison():
2     modalities = ['T1', 'T1c', 'T2', 'FLAIR']
3     features = ['firstorder_Mean', 'firstorder_Entropy',
4                'glcm_Contrast', 'glcm_Correlation']
```

```

5
6     fig, axes = plt.subplots(2, 2, figsize=(14, 10))
7     axes = axes.flatten()
8
9     for idx, feature in enumerate(features):
10        ax = axes[idx]
11        data = []
12
13        for mod in modalities:
14            col = mod + '_' + feature
15            if col in df.columns:
16                for patient in df['patient_id'].unique():
17                    val = df[df['patient_id'] == patient][col].values[0]
18                    data.append({'modality': mod, 'value': val,
19                               'patient': patient})
20
21        plot_df = pd.DataFrame(data)
22        sns.boxplot(data=plot_df, x='modality', y='value',
23                   hue='patient', ax=ax)
24        ax.set_title(feature)
25        ax.legend(title='Patient')
26
27        plt.tight_layout()
28        plt.savefig('modality_comparison.png', dpi=300)
29        plt.show()
30
31    plot_modality_comparison()

```

Correlation Analysis

```

1     def plot_correlation_matrix():
2         modalities = ['T1', 'T1c', 'T2', 'FLAIR']
3         features = ['firstorder_Mean', 'firstorder_Entropy',
4                    'glcm_Contrast', 'glcm_Correlation']
5
6         corr_dict = {}
7         for patient in df['patient_id'].unique():
8             for feat in features:
9                 for mod in modalities:
10                    col = mod + '_' + feat
11                    if col in df.columns:
12                        val = df[df['patient_id'] == patient][col].values[0]
13                        if isinstance(val, (int, float)):
14                            key = patient + '_' + mod + '_' + feat
15                            corr_dict[key] = val
16
17        corr_data = []
18        for patient in df['patient_id'].unique():
19            row = []
20            for mod in modalities:
21                for feat in features:
22                    key = patient + '_' + mod + '_' + feat
23                    row.append(corr_dict.get(key, np.nan))
24            corr_data.append(row)
25
26        cols = []
27        for m in modalities:
28            for f in features:
29                cols.append(m + '_' + f)

```

```

30
31     corr_df = pd.DataFrame(corr_data, index=df['patient_id'].unique(),
32     columns=cols)
33     corr_matrix = corr_df.T.corr()
34
35     plt.figure(figsize=(10, 8))
36     sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm',
37     square=True)
38     plt.title('Correlation_Matrix')
39     plt.tight_layout()
40     plt.savefig('correlation_matrix.png', dpi=300)
41     plt.show()
42
43     plot_correlation_matrix()

```

Radar Chart

```

1     def plot_radar_chart():
2         radar_features = [
3             ('shape_Sphericity', 'Sphericity'),
4             ('firstorder_Mean', 'Mean'),
5             ('firstorder_Entropy', 'Entropy'),
6             ('glcm_Contrast', 'Contrast'),
7             ('glcm_Correlation', 'Correlation')
8         ]
9
10        modalities = ['T1', 'T1c']
11        fig, axes = plt.subplots(1, 2, figsize=(14, 6),
12        subplot_kw=dict(polar=True))
13
14        for idx, mod in enumerate(modalities):
15            ax = axes[idx]
16            labels = []
17            p1_vals = []
18            p2_vals = []
19
20            for feat, label in radar_features:
21                col = mod + '_' + feat
22                if col in df.columns:
23                    labels.append(label)
24                    p1 = df[df['patient_id'] == 'BRATS_001'][col].values[0]
25                    p2 = df[df['patient_id'] == 'BRATS_002'][col].values[0]
26                    if isinstance(p1, (int, float)):
27                        p1_vals.append(p1)
28                        p2_vals.append(p2)
29
30            if p1_vals:
31                all_vals = p1_vals + p2_vals
32                if min(all_vals) != max(all_vals):
33                    min_val = min(all_vals)
34                    max_val = max(all_vals)
35                else:
36                    min_val = 0
37                    max_val = 1
38
39            p1_norm = [(v - min_val) / (max_val - min_val) for v in p1_vals]
40            p2_norm = [(v - min_val) / (max_val - min_val) for v in p2_vals]
41

```

```

42     angles = np.linspace(0, 2*np.pi, len(labels), endpoint=False).
        tolist()
43     angles += angles[:1]
44     p1_norm += p1_norm[:1]
45     p2_norm += p2_norm[:1]
46
47     ax.plot(angles, p1_norm, 'o-', linewidth=2,
48             label='BRATS_001', color='#FF6B6B')
49     ax.fill(angles, p1_norm, alpha=0.25, color='#FF6B6B')
50     ax.plot(angles, p2_norm, 'o-', linewidth=2,
51             label='BRATS_002', color='#4ECDC4')
52     ax.fill(angles, p2_norm, alpha=0.25, color='#4ECDC4')
53     ax.set_xticks(angles[:-1])
54     ax.set_xticklabels(labels, size=10)
55     ax.set_ylim(0, 1)
56     ax.set_title('Modality:␣' + mod)
57     ax.legend(loc='upper␣right', bbox_to_anchor=(1.3, 1.0))
58
59     plt.suptitle('Patient␣Profiles␣Comparison')
60     plt.tight_layout()
61     plt.savefig('radar_chart.png', dpi=300)
62     plt.show()
63
64     plot_radar_chart()

```

PCA Analysis

```

1     def plot_pca_analysis():
2         patterns = ['firstorder', 'glcm', 'glrlm', 'glszm']
3         selected_cols = []
4
5         for pattern in patterns:
6             cols = [col for col in df.columns if pattern in col
7                     and 'wavelet' not in col]
8             selected_cols.extend(cols[:10])
9
10        X = df[selected_cols].select_dtypes(include=[np.number]).fillna(0)
11        scaler = StandardScaler()
12        X_scaled = scaler.fit_transform(X)
13
14        pca = PCA(n_components=2)
15        X_pca = pca.fit_transform(X_scaled)
16
17        fig, axes = plt.subplots(1, 2, figsize=(14, 6))
18
19        ax1 = axes[0]
20        explained = pca.explained_variance_ratio_
21        cum_var = np.cumsum(explained)
22        ax1.bar(range(1, len(explained)+1), explained, alpha=0.7, label='
            Ind')
23        ax1.plot(range(1, len(explained)+1), cum_var, 'ro-', label='Cum')
24        ax1.set_xlabel('Principal␣Component')
25        ax1.set_ylabel('Explained␣Variance')
26        ax1.set_title('Scree␣Plot')
27        ax1.legend()
28        ax1.grid(True, alpha=0.3)
29
30        ax2 = axes[1]
31        patients = df['patient_id'].values

```

```

32     colors = {'BRATS_001': '#FF6B6B', 'BRATS_002': '#4ECDC4'}
33     for patient in df['patient_id'].unique():
34         mask = (patients == patient)
35         ax2.scatter(X_pca[mask, 0], X_pca[mask, 1] if X_pca.shape[1] > 1
36                     else 0,
37                     c=colors[patient], s=200, label=patient,
38                     edgecolors='black', linewidth=2, alpha=0.8)
39
40     pc1_text = 'PC1_(' + str(round(explained[0]*100, 1)) + '%)'
41     pc2_text = 'PC2_(' + str(round(explained[1]*100, 1)) + '%)'
42     ax2.set_xlabel(pc1_text)
43     ax2.set_ylabel(pc2_text)
44     ax2.set_title('PCA_Projection')
45     ax2.legend()
46     ax2.grid(True, alpha=0.3)
47
48     plt.tight_layout()
49     plt.savefig('pca_analysis.png', dpi=300)
50     plt.show()
51
52     plot_pca_analysis()

```

Feature Importance

```

1     def plot_feature_importance():
2         patterns = ['firstorder_Mean', 'firstorder_Entropy',
3                     'glcm_Contrast', 'glcm_Correlation',
4                     'glrlm_RunEntropy', 'glszm_ZoneEntropy']
5
6         selected_cols = []
7         for pattern in patterns:
8             cols = [col for col in df.columns if pattern in col
9                     and 'wavelet' not in col]
10            selected_cols.extend(cols[:4])
11
12            X = df[selected_cols].select_dtypes(include=[np.number]).fillna(0)
13            y = (df['patient_id'] == 'BRATS_001').astype(int)
14
15            rf = RandomForestClassifier(n_estimators=100, random_state=42)
16            rf.fit(X, y)
17
18            importance = pd.DataFrame({
19                'feature': selected_cols[:len(rf.feature_importances_)],
20                'importance': rf.feature_importances_
21            }).sort_values('importance', ascending=False)
22
23            plt.figure(figsize=(12, 8))
24            top_n = min(15, len(importance))
25
26            colors = []
27            for f in importance.head(top_n)['feature']:
28                if 'T1' in f:
29                    colors.append('#FF6B6B')
30                elif 'T1c' in f:
31                    colors.append('#FF9F43')
32                elif 'T2' in f:
33                    colors.append('#4ECDC4')
34                else:
35                    colors.append('#45B7D1')

```

```

36     sns.barplot(data=importance.head(top_n), x='importance', y='
37         feature',
38         palette=colors)
39     plt.title('Feature Importance')
40     plt.xlabel('Importance')
41     plt.tight_layout()
42     plt.savefig('feature_importance.png', dpi=300)
43     plt.show()
44
45     print(importance.head(10))
46
47     plot_feature_importance()

```

Comprehensive Heatmap

```

1     def plot_comprehensive_heatmap():
2         key_features = [
3             'shape_Sphericity',
4             'firstorder_Mean',
5             'firstorder_Entropy',
6             'firstorder_Variance',
7             'glcm_Contrast',
8             'glcm_Correlation',
9             'glrlm_RunEntropy',
10            'glszm_ZoneEntropy'
11        ]
12        modalities = ['T1', 'T1c', 'T2', 'FLAIR']
13
14        heatmap_data = []
15        row_labels = []
16
17        for patient in df['patient_id'].unique():
18            for mod in modalities:
19                row_name = patient + '_' + mod
20                row_labels.append(row_name)
21                row_values = []
22                for feat in key_features:
23                    col = mod + '_' + feat
24                    if col in df.columns:
25                        val = df[df['patient_id'] == patient][col].values[0]
26                        if isinstance(val, (int, float)):
27                            row_values.append(val)
28                        else:
29                            row_values.append(0)
30                    else:
31                        row_values.append(0)
32                heatmap_data.append(row_values)
33
34        heatmap_df = pd.DataFrame(heatmap_data, index=row_labels,
35            columns=[f[:15] for f in key_features])
36
37        mean_vals = heatmap_df.mean(axis=1).values.reshape(-1, 1)
38        std_vals = heatmap_df.std(axis=1).values.reshape(-1, 1)
39        heatmap_norm = (heatmap_df - mean_vals) / std_vals
40        heatmap_norm = heatmap_norm.fillna(0)
41
42        plt.figure(figsize=(14, 6))
43        sns.heatmap(heatmap_norm, annot=True, fmt='.2f', cmap='coolwarm',

```

```

44     cbar_kws={'label': 'Z-score'},
45     linewidths=0.5, linecolor='white')
46     plt.title('Comprehensive_Radiomic_Analysis')
47     plt.xlabel('Features')
48     plt.ylabel('Patient_/Modality')
49     plt.tight_layout()
50     plt.savefig('heatmap.png', dpi=300)
51     plt.show()
52
53     plot_comprehensive_heatmap()

```

Ключевые выводы

1. T1c показывает наибольшую гетерогенность опухоли.
2. T2 и FLAIR отражают перитуморальный отёк.
3. GLCM_Contrast и GLCM_Entropy наиболее информативны.
4. Shape_Sphericity отражает агрессивность роста.
5. BRATS_001 имеет более гетерогенную опухоль.
6. BRATS_002 имеет более компактную опухоль.

Сводная клиническая интерпретация

BRATS_001:

- Более гетерогенная структура опухоли.
- Высокая энтропия и контраст на T2/FLAIR.
- Типичен для глиобластомы.

BRATS_002:

- Более высокие значения корреляции.
- Относительно низкая энтропия.
- Характерен для менее агрессивных новообразований.

Выводы

1. Успешно выполнено сравнение модальностей МРТ.
2. Построены корреляционные матрицы.
3. Созданы радарные диаграммы.
4. Выполнен PCA-анализ.
5. Определены важнейшие признаки.
6. Сформулированы клинические выводы.

Контрольные вопросы

Теоретический блок

1. Объясните физическую природу модальностей МРТ.
2. Что такое стандарт BRATS?
3. Почему текстурные признаки более информативны?
4. Что означает низкая сферичность?
5. Объясните принцип PCA.

Практический блок

1. Проанализируйте распределение энтропии для BRATS_001.
2. Сравните контраст на T1c для обоих пациентов.
3. Постройте корреляционную матрицу.
4. Определите важнейшие признаки.
5. Предложите дополнительные модальности.

Рефлексивный блок

1. С какими трудностями вы столкнулись?
2. Как навыки радиомики могут быть применены?
3. Какие этические аспекты необходимо учитывать?

Список литературы

1. Bakas, S. Advancing The Cancer Genome Atlas glioma MRI collections. – 2017.
2. Lambin, P. Radiomics: the bridge between medical imaging and personalized medicine. – 2017.
3. Menze, B. H. The Multimodal Brain Tumor Image Segmentation Benchmark. – 2015.
4. van Griethuysen, J. J. M. Computational Radiomics System. – 2017.
5. Владимирский, А. В. Компьютерное зрение в лучевой диагностике. – 2023.

Приложение А. Полный код скрипта анализа

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from sklearn.preprocessing import StandardScaler
6 from sklearn.decomposition import PCA
7 from sklearn.ensemble import RandomForestClassifier
8 from sklearn.model_selection import cross_val_score
9 import warnings
10 warnings.filterwarnings('ignore')
11
12 #
13 # =====
14 # 1. LOAD DATA
15 #
16 # =====
17
18 df = pd.read_csv('combined_multichannel_features.csv')
19 print("Patients:", df['patient_id'].unique())
20 print("Shape:", df.shape)
21
22 #
23 # =====
24 # 2. MODALITY COMPARISON
25 #
26 # =====
27
28 def plot_modality_comparison():
29     modalities = ['T1', 'T1c', 'T2', 'FLAIR']
30     features = ['firstorder_Mean', 'firstorder_Entropy',
31                'glcm_Contrast', 'glcm_Correlation']
32
33     fig, axes = plt.subplots(2, 2, figsize=(14, 10))
34     axes = axes.flatten()
35
36     for idx, feature in enumerate(features):
37         ax = axes[idx]
38         data = []
39
40         for mod in modalities:
41             col = mod + '_' + feature
42             if col in df.columns:
43                 for patient in df['patient_id'].unique():
44                     val = df[df['patient_id'] == patient][col].values[0]
45                     data.append({'modality': mod, 'value': val,
46                                'patient': patient})
47
48         plot_df = pd.DataFrame(data)
49         sns.boxplot(data=plot_df, x='modality', y='value',
50                     hue='patient', ax=ax)
51         ax.set_title(feature)
52         ax.legend(title='Patient')
53
54     plt.tight_layout()
```

```

51 plt.savefig('modality_comparison.png', dpi=300)
52 plt.show()
53
54 plot_modality_comparison()
55
56 #
=====
57 # 3. CORRELATION MATRIX
58 #
=====
59
60 def plot_correlation_matrix():
61     modalities = ['T1', 'T1c', 'T2', 'FLAIR']
62     features = ['firstorder_Mean', 'firstorder_Entropy',
63                'glcm_Contrast', 'glcm_Correlation']
64
65     corr_dict = {}
66     for patient in df['patient_id'].unique():
67         for feat in features:
68             for mod in modalities:
69                 col = mod + '_' + feat
70                 if col in df.columns:
71                     val = df[df['patient_id'] == patient][col].values[0]
72                     if isinstance(val, (int, float)):
73                         key = patient + '_' + mod + '_' + feat
74                         corr_dict[key] = val
75
76     corr_data = []
77     for patient in df['patient_id'].unique():
78         row = []
79         for mod in modalities:
80             for feat in features:
81                 key = patient + '_' + mod + '_' + feat
82                 row.append(corr_dict.get(key, np.nan))
83             corr_data.append(row)
84
85     cols = []
86     for m in modalities:
87         for f in features:
88             cols.append(m + '_' + f)
89
90     corr_df = pd.DataFrame(corr_data, index=df['patient_id'].unique(),
91                           columns=cols)
92     corr_matrix = corr_df.T.corr()
93
94     plt.figure(figsize=(10, 8))
95     sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm',
96                square=True)
97     plt.title('Correlation_Matrix')
98     plt.tight_layout()
99     plt.savefig('correlation_matrix.png', dpi=300)
100    plt.show()
101
102    plot_correlation_matrix()
103
104    #
=====

```

```

105 # 4. RADAR CHART
106 #
=====
107
108 def plot_radar_chart():
109     radar_features = [
110         ('shape_Sphericity', 'Sphericity'),
111         ('firstorder_Mean', 'Mean'),
112         ('firstorder_Entropy', 'Entropy'),
113         ('glcm_Contrast', 'Contrast'),
114         ('glcm_Correlation', 'Correlation')
115     ]
116
117     modalities = ['T1', 'T1c']
118     fig, axes = plt.subplots(1, 2, figsize=(14, 6),
119                             subplot_kw=dict(polar=True))
120
121     for idx, mod in enumerate(modalities):
122         ax = axes[idx]
123         labels = []
124         p1_vals = []
125         p2_vals = []
126
127         for feat, label in radar_features:
128             col = mod + '_' + feat
129             if col in df.columns:
130                 labels.append(label)
131                 p1 = df[df['patient_id'] == 'BRATS_001'][col].values[0]
132                 p2 = df[df['patient_id'] == 'BRATS_002'][col].values[0]
133                 if isinstance(p1, (int, float)):
134                     p1_vals.append(p1)
135                     p2_vals.append(p2)
136
137             if p1_vals:
138                 all_vals = p1_vals + p2_vals
139                 if min(all_vals) != max(all_vals):
140                     min_val = min(all_vals)
141                     max_val = max(all_vals)
142                 else:
143                     min_val = 0
144                     max_val = 1
145
146             p1_norm = [(v - min_val) / (max_val - min_val) for v in p1_vals]
147             p2_norm = [(v - min_val) / (max_val - min_val) for v in p2_vals]
148
149             angles = np.linspace(0, 2*np.pi, len(labels), endpoint=False).
150                 tolist()
151             angles += angles[:1]
152             p1_norm += p1_norm[:1]
153             p2_norm += p2_norm[:1]
154
155             ax.plot(angles, p1_norm, 'o-', linewidth=2,
156                   label='BRATS_001', color='#FF6B6B')
157             ax.fill(angles, p1_norm, alpha=0.25, color='#FF6B6B')
158             ax.plot(angles, p2_norm, 'o-', linewidth=2,
159                   label='BRATS_002', color='#4ECDC4')
160             ax.fill(angles, p2_norm, alpha=0.25, color='#4ECDC4')
161             ax.set_xticks(angles[:-1])
162             ax.set_xticklabels(labels, size=10)

```

```

162     ax.set_ylim(0, 1)
163     ax.set_title('Modality:_' + mod)
164     ax.legend(loc='upper_right', bbox_to_anchor=(1.3, 1.0))
165
166     plt.suptitle('Patient_Profiles_Comparison')
167     plt.tight_layout()
168     plt.savefig('radar_chart.png', dpi=300)
169     plt.show()
170
171     plot_radar_chart()
172
173     #
174     # =====
175     # 5. PCA ANALYSIS
176     #
177     # =====
178
179     def plot_pca_analysis():
180         patterns = ['firstorder', 'glcm', 'glrlm', 'glszm']
181         selected_cols = []
182
183         for pattern in patterns:
184             cols = [col for col in df.columns if pattern in col
185                     and 'wavelet' not in col]
186             selected_cols.extend(cols[:10])
187
188         X = df[selected_cols].select_dtypes(include=[np.number]).fillna(0)
189         scaler = StandardScaler()
190         X_scaled = scaler.fit_transform(X)
191
192         pca = PCA(n_components=2)
193         X_pca = pca.fit_transform(X_scaled)
194
195         fig, axes = plt.subplots(1, 2, figsize=(14, 6))
196
197         ax1 = axes[0]
198         explained = pca.explained_variance_ratio_
199         cum_var = np.cumsum(explained)
200         ax1.bar(range(1, len(explained)+1), explained, alpha=0.7, label='Ind')
201         ax1.plot(range(1, len(explained)+1), cum_var, 'ro-', label='Cum')
202         ax1.set_xlabel('Principal_Component')
203         ax1.set_ylabel('Explained_Variance')
204         ax1.set_title('Scree_Plot')
205         ax1.legend()
206         ax1.grid(True, alpha=0.3)
207
208         ax2 = axes[1]
209         patients = df['patient_id'].values
210         colors = {'BRATS_001': '#FF6B6B', 'BRATS_002': '#4ECDC4'}
211         for patient in df['patient_id'].unique():
212             mask = (patients == patient)
213             ax2.scatter(X_pca[mask, 0], X_pca[mask, 1] if X_pca.shape[1] > 1
214                        else 0,
215                        c=colors[patient], s=200, label=patient,
216                        edgecolors='black', linewidth=2, alpha=0.8)
217
218         pc1_text = 'PC1_(' + str(round(explained[0]*100, 1)) + '%',

```

```

216 pc2_text = 'PC2_(' + str(round(explained[1]*100, 1)) + '%)'
217 ax2.set_xlabel(pc1_text)
218 ax2.set_ylabel(pc2_text)
219 ax2.set_title('PCA_Projection')
220 ax2.legend()
221 ax2.grid(True, alpha=0.3)
222
223 plt.tight_layout()
224 plt.savefig('pca_analysis.png', dpi=300)
225 plt.show()
226
227 plot_pca_analysis()
228
229 #
=====
230 # 6. FEATURE IMPORTANCE
231 #
=====
232
233 def plot_feature_importance():
234     patterns = ['firstorder_Mean', 'firstorder_Entropy',
235               'glcm_Contrast', 'glcm_Correlation',
236               'glrlm_RunEntropy', 'glszm_ZoneEntropy']
237
238     selected_cols = []
239     for pattern in patterns:
240         cols = [col for col in df.columns if pattern in col
241               and 'wavelet' not in col]
242         selected_cols.extend(cols[:4])
243
244     X = df[selected_cols].select_dtypes(include=[np.number]).fillna(0)
245     y = (df['patient_id'] == 'BRATS_001').astype(int)
246
247     rf = RandomForestClassifier(n_estimators=100, random_state=42)
248     rf.fit(X, y)
249
250     importance = pd.DataFrame({
251         'feature': selected_cols[:len(rf.feature_importances_)],
252         'importance': rf.feature_importances_
253     }).sort_values('importance', ascending=False)
254
255     plt.figure(figsize=(12, 8))
256     top_n = min(15, len(importance))
257
258     colors = []
259     for f in importance.head(top_n)['feature']:
260         if 'T1' in f:
261             colors.append('#FF6B6B')
262         elif 'T1c' in f:
263             colors.append('#FF9F43')
264         elif 'T2' in f:
265             colors.append('#4ECDC4')
266         else:
267             colors.append('#45B7D1')
268
269     sns.barplot(data=importance.head(top_n), x='importance', y='
                feature',
270               palette=colors)

```

```

271 plt.title('Feature Importance')
272 plt.xlabel('Importance')
273 plt.tight_layout()
274 plt.savefig('feature_importance.png', dpi=300)
275 plt.show()
276
277 print(importance.head(10))
278
279 plot_feature_importance()
280
281 #
=====
282 # 7. COMPREHENSIVE HEATMAP
283 #
=====
284
285 def plot_comprehensive_heatmap():
286     key_features = [
287         'shape_Sphericity',
288         'firstorder_Mean',
289         'firstorder_Entropy',
290         'firstorder_Variance',
291         'glcm_Contrast',
292         'glcm_Correlation',
293         'glrlm_RunEntropy',
294         'glszm_ZoneEntropy'
295     ]
296     modalities = ['T1', 'T1c', 'T2', 'FLAIR']
297
298     heatmap_data = []
299     row_labels = []
300
301     for patient in df['patient_id'].unique():
302         for mod in modalities:
303             row_name = patient + '_' + mod
304             row_labels.append(row_name)
305             row_values = []
306             for feat in key_features:
307                 col = mod + '_' + feat
308                 if col in df.columns:
309                     val = df[df['patient_id'] == patient][col].values[0]
310                     if isinstance(val, (int, float)):
311                         row_values.append(val)
312                     else:
313                         row_values.append(0)
314                 else:
315                     row_values.append(0)
316             heatmap_data.append(row_values)
317
318     heatmap_df = pd.DataFrame(heatmap_data, index=row_labels,
319                               columns=[f[:15] for f in key_features])
320
321     mean_vals = heatmap_df.mean(axis=1).values.reshape(-1, 1)
322     std_vals = heatmap_df.std(axis=1).values.reshape(-1, 1)
323     heatmap_norm = (heatmap_df - mean_vals) / std_vals
324     heatmap_norm = heatmap_norm.fillna(0)
325
326     plt.figure(figsize=(14, 6))

```

```

327     sns.heatmap(heatmap_norm, annot=True, fmt='.2f', cmap='coolwarm',
328     cbar_kws={'label': 'Z-score'},
329     linewidths=0.5, linecolor='white')
330     plt.title('Comprehensive_Radiomic_Analysis')
331     plt.xlabel('Features')
332     plt.ylabel('Patient_/Modality')
333     plt.tight_layout()
334     plt.savefig('heatmap.png', dpi=300)
335     plt.show()
336
337     plot_comprehensive_heatmap()
338
339     print("\nAnalysis completed successfully!")

```

Приложение Б. Сводная таблица результатов

Таблица 2: Сравнение ключевых признаков

Признак	Пациент	T1	T1c	T2	FLAIR	Среднее
Entropy	BRATS_001	3.72	4.15	3.98	4.34	4.05
Entropy	BRATS_002	3.45	3.88	3.67	4.12	3.78
Contrast	BRATS_001	245.6	312.4	278.9	298.7	283.9
Contrast	BRATS_002	198.3	256.7	223.4	245.6	231.0
Correlation	BRATS_001	0.85	0.82	0.87	0.84	0.85
Correlation	BRATS_002	0.91	0.88	0.92	0.89	0.90
Sphericity	BRATS_001	0.42	0.45	0.40	0.43	0.43
Sphericity	BRATS_002	0.58	0.62	0.55	0.60	0.59

Таблица 3: Клиническая интерпретация

Параметр	BRATS_001	BRATS_002
Энтропия	Высокая	Низкая
Контраст	Высокий	Низкий
Корреляция	Низкая	Высокая
Сферичность	Низкая	Высокая
Фенотип	Агрессивный	Неагрессивный