WROCŁAW UNIVERSITY OF TECHNOLOGY

MASTER'S THESIS

Combined Radiology and Pathology Classification of Brain Tumors

Author:
Piotr GIEDZIUN

Supervisor:
dr hab. inż. Henryk
MACIEJEWSKI

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Computer Science
Faculty of Electronics

January 19, 2016

I would like to dedicate this thesis to my parents. For their support and encouragement through my life.

I would also like to thank my colleagues Witold Dyrka, PhD., Jakub Czakon, Piotr Krajewski and Grzegorz Żurek, who worked with me on Pathology Segmentation and Classification. For their support, knowledge and effort that they put into the project.

Contents

Chapter 1

Introduction

1.1 Introduction

Brain tumors are classified based on the location of the tumor, the type of tissue were the tumor emerges and whether it is benign or malignant. In my thesis, I will only pay attention to the type of the cancer, specifically whether it is Oligodendroglioma or Astrocytoma.

Over time, a low-grade tumor can become a high-grade tumor. Therefore quick and accurate diagnosis is essential for the patient life expectancy prognosis. Manual classification of Magnetic Resonance (MR) images is a time-consuming task. The Computer-aided system should, at least, be able to speed up the process by selecting the region of interest (ROI) and narrowing potential cancer types. MRI scan is performed long before the tumor biopsy, therefore, classifier based on the MRI could speed up the decisions regarding the treatment. Due to variations in intensity and ambiguous tumor contrast in the MRI types, it is extremely difficult task. With the current advance in the technology, more advanced and complex models can be tested. The bottleneck of progress in automatic MRI-based classification of brain tumors is number of available samples. The Medical Image Computing and Computer Assisted Intervention (MICCAI) Society, alongside with medical partners is working toward the goal of open access to the medical data. However, availability of experimental data sets is often limited to the competition contestants i.e. external access is prohibited.

Image segmentation is a well known problem in the field of computer vision. Medical imaging is one of its applications. Several techniques have been introduced by the community, such as clustering methods, thresholding, histogram-based methods, edge detection, graph partitioning methods among others. Even though they are considered general-purpose algorithms, they have their individual characteristics and limitations. Researcher has to fully exploit strengths and weaknesses of tested method to find out if given method can be applied to the case at hand.

1.2 Goal of the thesis

The main goal of the thesis was to build a segmentation mechanism for MRI (Magnetic resonance imaging) slices. On segmented image, I had to detect tumor on the sequence of multiple MR scans. I had to find a sufficient method and adopt it to the requirements of my data set.

Another goal was to find out whether it is possible to build a binary tumor type classifier based on the sequence of radiology images. The data set had two classes. The objective was to classify each sample (data vector) into Oligodendrogliomas or Astrocytomas. Then combine it with already built pathology image-based classifier and find out if it is able to enhance it.

1.3 Related work

Medical image segmentation is a well-known topic in the literature. It is a very wide field, where data sets tend to vary considerably. Countless different techniques of obtaining medical images were introduced. Moreover, medical images of various parts of the human organism tend to differ in terms of individual characteristics, dimensions and purpose.

Main focus of my research was brain image segmentation and classification. The topic of brain segmentation is relatively popular thanks to BraTS challenge [NMB13]. Multimodal Brain Tumor Image Segmentation (BraTS) challenge is organized in conjunction with the MICCAI conference.

Before segmentation is performed input data has to be pre-processed. In the data sets without bias correction (such as MRI), N4ITK method was proposed [Tus+10]. N4ITK is an intensity normalization algorithm, that allows to normalize sequence to chosen reference. It was proposed for supervised methods, that were sensitive to intensity modulation. Gaussian and Poisson noise are usually present in MR images, most of de-noising algorithms leave some kind of artifacts. Median filter was proposed as an effective method for dealing with this issue [AMEAA15]. It has edge preserving capability, that is a key for many segmentation techniques.

As for the segmentation part, several supervised and unsupervised algorithms were proposed. Among the supervised techniques Random Decision Forest that classifies voxels (features such as neighborhood information, context information and texture) [Fes+13] should be mentioned. Fuzzy C-means clustering, Mean Shift and K-means clustering were proposed in the unsupervised paradigm [CMK10]. Segmented image had to be processed in the post-segmentation step, to extract the tumor from segmented clusters. Proposed post-processing based on feature selection had problems with tumor extraction [CMK10]. They assumed that tumor is a cluster with the highest average intensity level, that had an area above a constant threshold.

Classification however is a slightly less popular subject. Current diagnosis fully rely on histopathology imaging [Ass16c] [Ass16a]. Radiology imaging based classification is subject of scientific studies. There are over 120 types of brain tumors [Ass16d], thus there is still plenty of work.

Feature extraction technique was slightly modified in the literature, although the idea was the same i.e. extraction of structure information of intensity, shape, and texture [RD12]. Then features were evaluated by feature selection process. To reduce dimension of feature vectors PCA was proposed [RD12].

Another approach that was able to detect the grade of the cancer used gray-level co-occurrence matrix (GLCM). The GLCM functions characterize the texture of the image based on the spatial relationship of the pixels. Feed forward neural network was used for classification based on features extracted by GLCM [ZP12].

1.4 Magnetic resonance imaging

Magnetic resonance imaging (MRI) scanner uses magnetic field and radio waves to create a detailed image of human body [Han09]. Equivalent technique used to accomplish similar results is known as Computed Tomography (CT). They both are noninvasive (does not require removal of tissue/incision into the body) and painless methods, that are commonly used for exploring the brain, each with its own strengths and weaknesses. Currently, MRI is the most sensitive imaging technique, providing detailed and accurate information. Moreover, CT is less recommended due to ionizing radiation exposure [PEP09].

In produced image, contrast is a product of tissue density (mainly proton density) as well as relaxation properties (T1 relaxation and T2 relaxation) [JRH16]. Two essential scanning parameters are TR (repetition time) and TE (echo time). Their combinations result in different image-weighting techniques (such as PD-weighted image, T1-weighted image etc.). T1 is the time when 63% of the longitudinal magnetization has recovered, meanwhile, T2 is the time when 63% of the transverse magnetization has decayed [ECM15].

In most of my cases, I had only T1, T1C, T2, and FLAIR. Therefore, I will only describe those types. Each of them has their own distinct characteristics, that can be used for specific purpose. Sample slices of each are presented in Figure 1.1.

MRI scan types: [ECM15]

- T1-weighted image is used mainly to extract the anatomical structure of the brain. Scanning parameters: short TR/short TE. Edema and most lesions have dark color. It is a pre-contrast image
- T1C-weighted image is a T1-weighted image with contrast applied to the patient blood system. It has brighter arteries and veins
- T2-weighted image is used for CSF (Cerebrospinal fluid) compartments. Scanning parameters: long TR/long TE. Brain cells filled with water are brighter. It will make lesions and edema easily distinguishable. Unfortunately blood and CSF is also bright

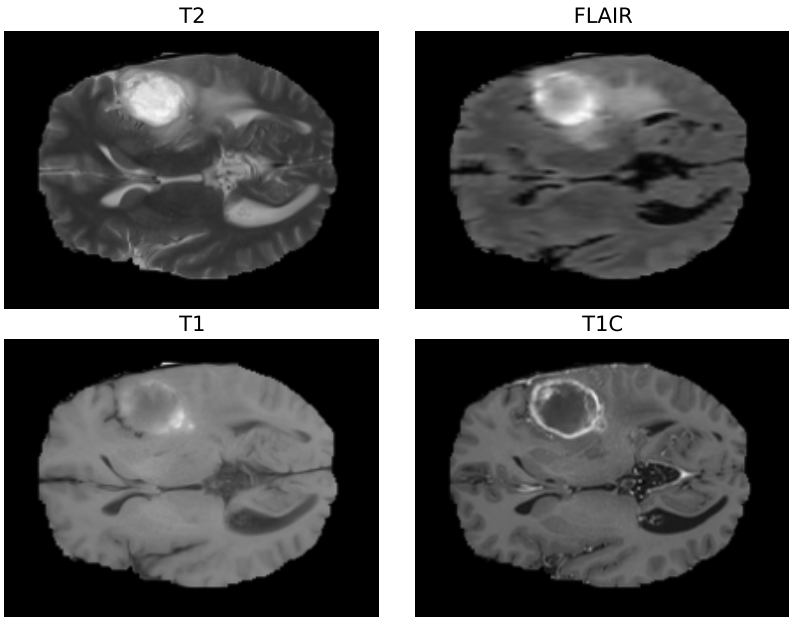

FIGURE 1.1: Types of MRI

- FLAIR (Fluid Attenuated Inversion Recover) is almost the same as T2-weighted image except CSF is dark here. Still most of the lesions should be bright. It was not the case in my data set. About 60% of lesions were dark or mixed

1.5 Histopathology

Histopathology is a study of unhealthy tissue, it is a field related to histology - the study of healthy tissue. In order to find anomalies we have to possess the knowledge of the healthy state. In a biopsy procedure surgeon removes a part of affected tissue for future examination. Later, histopathological tissue analysis is performed by a trained pathologist, whose role is to confirm the occurance of the disease and grade the found disease. In this thesis, I am not going to go into details of pathology. Histopathology images were used in the Imaging & Digital Pathology competition, where image-based classifier was created. I am not the only author of that model, it was mainly created by Grzegorz Żurek. I am using only results of that work, in form of Oligodendroglioma model estimates based on the cross-validation. Therefore, histopathological images are not used directly. More information about the pathology imaging and authors can be found in Chapter 3.

1.6 Brain Tumors

Brain tumor is an abnormal growth of tissue in the brain. The cause of this activity is currently unknown [Ass16b]. It is very important to classify brain tumors correctly. Based on that doctor will determine treatment and will be able to prepare prognosis for the patient.

There are two main types of tumors: [Ass16b]

- Benign do not invade nearby tissue, and is rather slow-growing. It has a clear outline and thus it can be removed by a surgeon. In special cases, benign tumor can be described as malignant due to its location (i.e. if it is located in vital areas of the brain)

- Malignant tumors are life threatening. Significantly different than normal brain cells, with cancerous background. Cells of malignant tumor can invade other tissues and spread through the blood stream or lymphatic system.

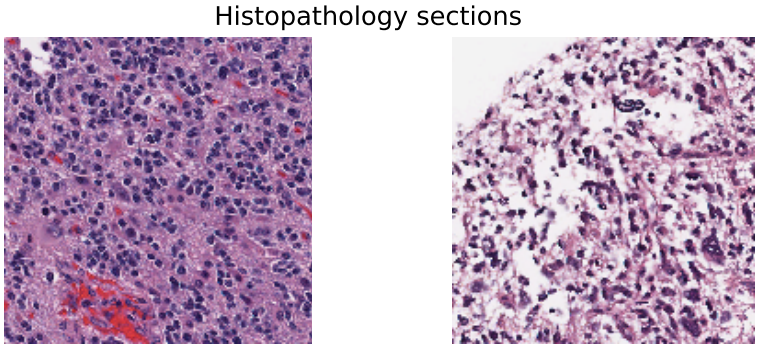

FIGURE 1.2: Selected slices of histopathological tissue

1.6.1 Oligodendroglioma and Astrocytoma

Oligodendroglioma and Astrocytoma are sub-types of glioma tumor.

- Oligodendroglioma tumor mainly develops in frontal and temporal lobes, although they can be found anywhere in the brain. These cells resemble fried eggs in the histopathology image [Ass16c], see Figure 1.3.
- Astrocytoma tumor can appear in various parts of the brain. In the histopathological image these cells exhibit a star-shaped pattern [Ass16a]. Astrocytomas are generally isointense on T1-weighted images and hyperintense on T2-weighted images [M.D16]

Figure 1.3 shows MRI scan and histopathology image of two patients. One with Astrocytoma and another with Oligodendroglioma.

1.7 NIFTI-1 file format

The Neuroimaging Informatics Technology Initiative (known as NIFTI) was introduced as a unified file format for radiological images. Format was built upon the *ANALYZE* file format, extending its predecessor with additional information such as orientation in space [HHS04]. This functionality was essential, without it distinction between left and right sides of brain was problematic. Manufacturers of radiological equipment were using different positions as a default one. In *ANALYZE* file format user had to adjust each sample manually before processing.

1.7.1 Headers

The first block of the file contains all image information. Headers are stored as key-value pairs. Most of them were inherited from the predecessor, although in a few cases their meaning was slightly changed. In order to be compatible with *ANALYZE* format, this segment is 348 bytes long. I will describe only selected subset of headers, that were used in my work.

More detailed description of headers: [HHS04]

- Dim is an array containing number of image dimensions (max 7). First index *dim[0]* indicates number of dimensions used. Following values are represented by positive integer equal to length of n-th dimension.
- Intent is an integer corresponding to enum of possible types (such as Correlation, Gama distribution or z score). In data set used in my thesis this parameter was always equal to 0 (None).
- Data type is a char describing type of data (boolean, RGB, int, etc).
- Slice start/end is an array index corresponding to first and last slice of scanned head. This information is also stored in the *dim* field.

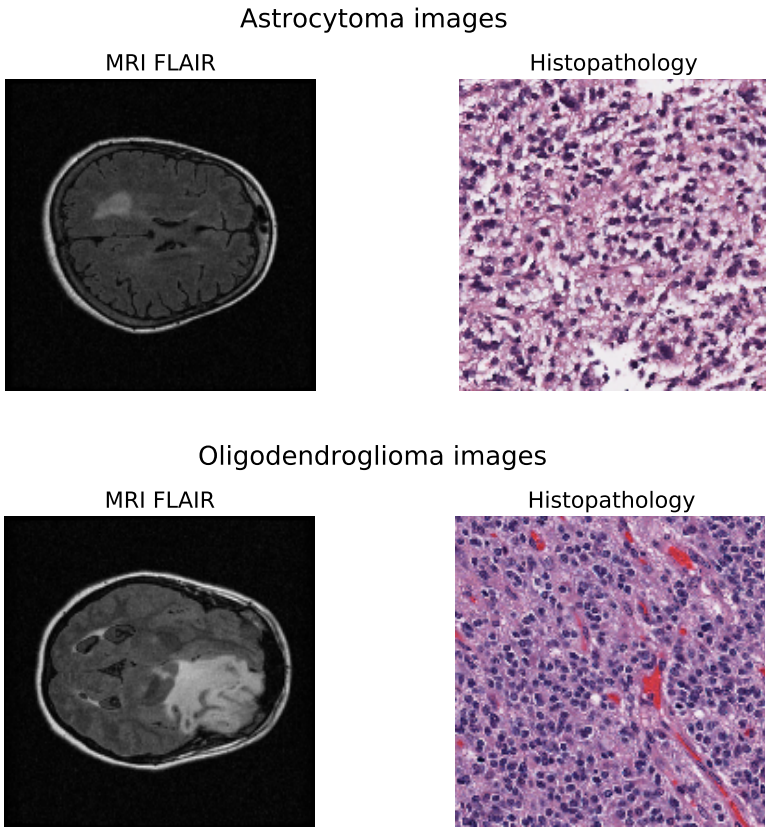

FIGURE 1.3: Oligodendroglioma and Astrocytoma images

- Voxel dimensions is a vector of floats, each with respect to n-th dimension. In this case index *pixdim[0]* is either -1 or 1. This value is used to calculate orientation transformation. Other values are used to calculate distance between slices/pixels. The unit is stored in *xyzt_units* field.
- Unit is a spatial and temporal measurement unit. It is an enum of units (0 - Unknown, 1 - Meter, 2 - Millimeter and so on).
- Description is not precisely defined in standard. It can hold up to 80 characters. Using this field I was able to distinguish the place where images were taken.
- Offset alongside with *pixdim* is used to calculate real position of an object in image.

Listing 1.1 shows the output of selected headers.

LISTING 1.1: Selected header pairs extracted from one of the samples

```
dim          : [ 3 224 256  26   1   1   1   1 ]
intent_p1    : 0.0
intent_p2    : 0.0
intent_p3    : 0.0
datatype     : int16
slice_start  : 0
pixdim       : [ -1. 0.8984375  0.8984375 6.5  ... ]
slice_end    : 25
xyzt_units   : 2
descrip      : M R H G FLAIR AX
qoffset_x    : 103.323791504
qoffset_y    : -88.4630126953
```

TABLE 1.1: Overview of used header

Type	Name	Description
short	dim[8]	Data array dimensions.
float	intent_p1	1st intent parameter.
float	intent_p2	2nd intent parameter.
float	intent_p3	3rd intent parameter.
short	datatype	Data type.
short	slice_start	First slice index.
float	pixdim[8]	Grid spacings.
short	slice_end	Last slice index.
char	xyzt_units	Units of pixdim[1..4].
char	descrip[80]	Device name.
float	qoffset_x	Quaternion x shift.
float	qoffset_y	Quaternion y shift.
float	qoffset_z	Quaternion z shift.
char	magic[4]	Magic string.

```
qoffset_z       : -83.2740936279
magic           : n+1
```

1.7.2 Data

The main section of the file is the Data block. Slices are stored according to the time of the acquisition. This is one of the reason why we have to read *slice_start* and *slice_stop* first. Based on their values we can proceed to reconstruction phase. In most cases we will be able to obtain 3 different viewing angles as shown in Fig 1.4.

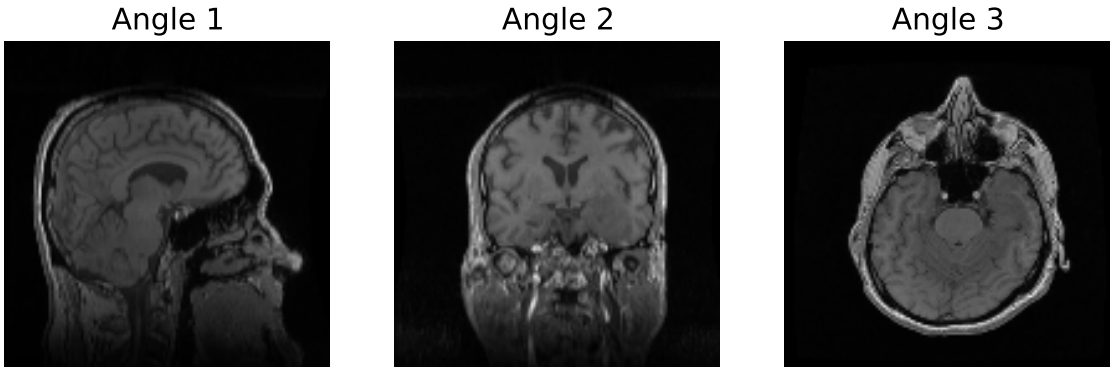

FIGURE 1.4: Viewing angles of MRI scan

In Figure 1.4 Angle 1 and Angle 2 were resized to Angle 3 shape. In reality their shape is *(512, 124)*, while Angle 3 is *(512,512)*.

In this paper I will only use the Angle 3 (axial view) for further processing. It turns out to be the input of the best resolution. It is one of the most popular perspective used in the literature.

Chapter 2

Radiology imaging

2.1 Data set

Original data set consists of 32 cases with lower grade glioma tumors (Oligodendroglioma and Astrocytoma). Each case has 3 or 4 MRI scans (T1, T1C, FLAIR, and T2) described in Section 1.4. Presented in Section 2.4 segmentation process requires T2 and FLAIR images. Unfortunately given data set was lacking T2 or/and FLAIR images in 7 cases. Due to similar properties of T1 and FLAIR, where it was possible T1 was used instead. T2 scan is unique, therefore I was unable to provide substitute for it. Final data consisted of 27 cases (13 of them with Oligodendroglioma and 14 with Astrocytoma). List of samples alongside with diagnosis is shown in Table 2.2.

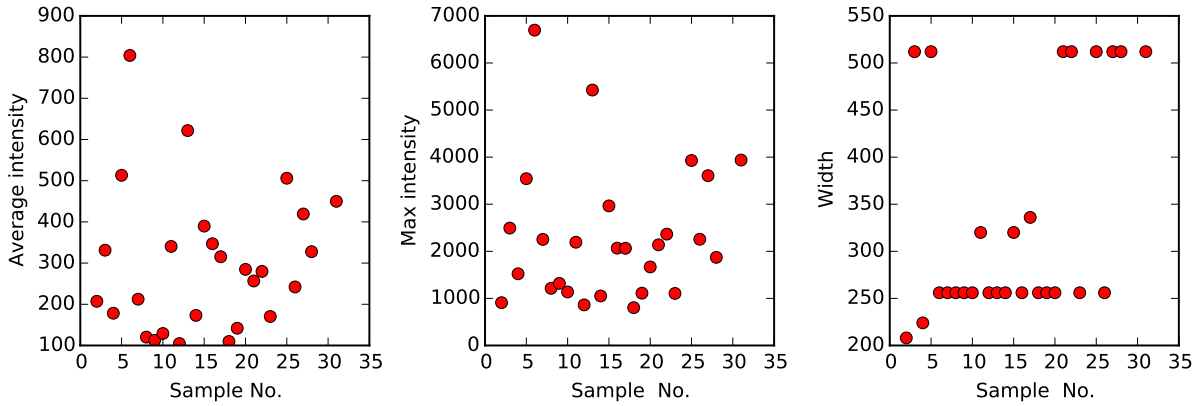

FIGURE 2.1: Plots of different attributes of the data set

Selected attributes of the samples are plot in Figure 2.1. It shows the diversity of the data set in most of the values. Standard deviation ($\sigma = \sqrt{\mu_2}$) of individual attributes is very high. Values were extracted from the middle slide of each sample, and shown in the Table 5.3.

Provided samples were taken using different hardware, therefore aliment and normalization of data was required (process is described in Section 2.3).

TABLE 2.1: Standard deviation of data attributes

ATTRIBUTE	AVERAGE	STD
Slice width	336.59	116.56
Slice height	339.55	113.91
Max intensity of slice	2315.03	1409.52
Average intensity of slice	299.53	167.97

TABLE 2.2: List of used data (A - Astrocytoma, O - Oligodendroglioma)

PATIENT No.	MRI SCANS	DIAGNOSIS
2	FLAIR, T2, T1, T1C	A
3	FLAIR, T2, T1, T1C	O
4	FLAIR, T2, T1, T1C	O
5	FLAIR, T2, T1, T1C	O
6	FLAIR, T2, T1, T1C	O
7	FLAIR, T2, T1, T1C	A
8	FLAIR, T2, T1, T1C	A
9	**T2, T1, T1C**	**A**
10	FLAIR, T2, T1, T1C	O
11	FLAIR, T2, T1, T1C	A
12	FLAIR, T2, T1, T1C	O
13	FLAIR, T2, T1	O
14	FLAIR, T2, T1	A
15	FLAIR, T2, T1, T1C	A
16	FLAIR, T2, T1, T1C	O
17	FLAIR, T2, T1, T1C	A
18	**T2, T1**	**A**
19	FLAIR, T2, T1, T1C	A
20	FLAIR, T2, T1C	O
21	FLAIR, T2, T1, T1C	O
22	FLAIR, T2, T1, T1C	O
23	FLAIR, T2	A
25	FLAIR, T2, T1, T1C	O
26	**T2, T1**	**A**
27	FLAIR, T2, T1, T1C	O
28	FLAIR, T2, T1, T1C	A
31	FLAIR, T2, T1, T1C	A

2.2 Process

In this chapter, I will present the steps to extract the features from input data. The simplified version of entire process is show in Figure 2.2. Each individual step will be explained in details in separate sections.

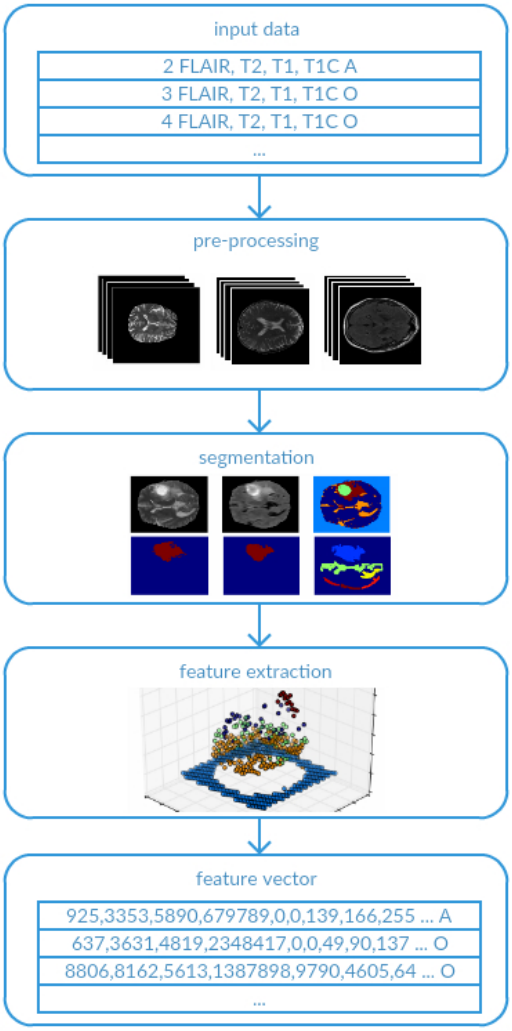

FIGURE 2.2: Simplified diagram of feature extraction process

2.3 Pre-processing

Input data used in this paper had to be unified and normalized. This section will cover all the steps that were necessary to meet segmentation requirements. But first I will list problems that I found in the data set.

Encountered problems:

- MRI scans were performed on different devices. Despite common file format they vary in intensity levels.
- Different types of MRI (performed on the same patient) are provided in different resolutions (X,Y and Z axes). Moreover aspect ratio might also differ.
- Patients were moving during scanning process or changed head position between them. It's especially visible in T1C image. Standard procedure consists of 3 scans (T1, T2 and FLAIR). T1C (T1

with contrast) is performed when there is anomaly detected. Patient is taken out of MRI scanner in order to inject the dye.

In order to explain the process I will use data of patient 4 (see Table 2.2). The same process is used for others.

Pre-processing steps:

- Standardize number of slices in T2 and FLAIR scans
- Change size of each slice to be the same (maintaining the aspect ratio)
- Skull stripping, adjust brain position
- Data processing (morphological operations, scale intensity)

In the following steps, I use only T2 and FLAIR images. Several researchers used T1, T2 and FLAIR for their segmentation [AMEAA15]. In my case I found addition of T1-weighted images resulted in worst performance. Information-wise, T1 and T1C is negligible (based on information provided in Section 1.4). The area that had low intensity in both T1 and FLAIR (it was a common case) would be recognized as a separate cluster.

LISTING 2.1: Reading data from NII file

```
>> n = NII2DInput(verbose=False)
>> n.load(4, data_types=("T2", "FLAIR"))
>> print n.header("T2")["dim"][1:4]
[256 256  92]
>> print n.header("FLAIR")["dim"][1:4]
[224 256  26]
```

For the patient no. 4 (shown in Listing 2.1), I have 92 slices (256x256) of T2 images and 26 slices (224x256) of FLAIR images. First step is to reduce number of slices, to be the same in loaded MRI scan types. Of course I have to chose the minimum of their slices count. It's done to avoid data augmentation, reduction does no produce false data. In this case it's 26. For the volume, that has more than 26 slices, I had to interpolate the slices. The formula is to pick every $number\_of\_slices/26$ slice.

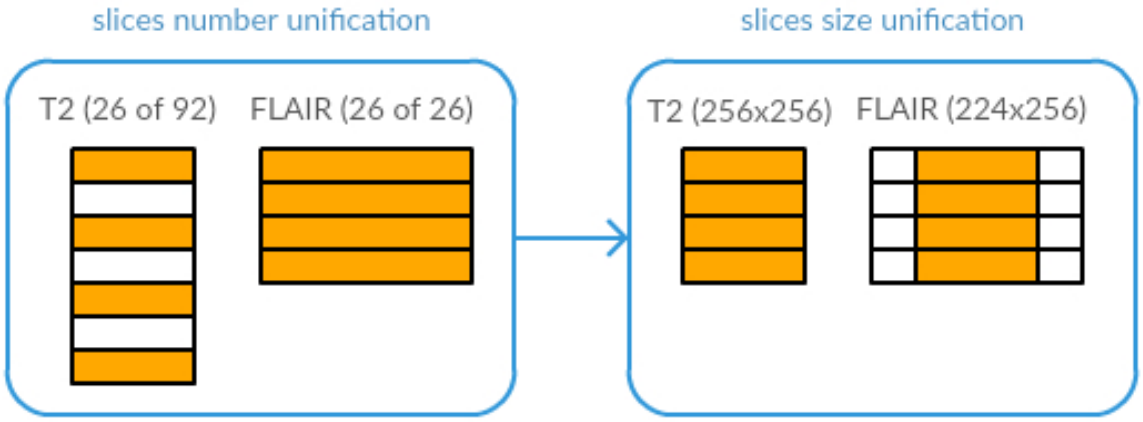

FIGURE 2.3: T2 and FLAIR unification

For T2-weighted slices I reduce them by 3.5-th starting from 0 (illustrated in Figure 2.3). As a result I will end-up with 26 slices as well.

Next phase is to unify slices sizes. I had to map each slice to be exactly the same size. Output array size is minimum of with and height of inputs. Here we have T2 with 256 by 256 size and FLAIR with 224 by 256 size. Output size is 224 by 256. If I would simply resize T2 image to output size it would break the aspect ratio of an image. Geometry of the head would be altered. In order to keep aspect ratio I had to resize it to smaller size, that's still compatible with original aspect ratio. Then fit it in the center of output image. The process is shown in Listing 2.2.

LISTING 2.2: Resize and keep aspect ratio

```
# aspect ratio of input image
in_aspect = float(data.shape[0])/float(data.shape[1])
# aspect ratio of output image
out_aspect = float(min_h)/float(min_w)

# resize the input image
if in_aspect >= out_aspect:
        output = resize(data,
        (min_h, int((min_h/in_aspect)+0.5)),
        preserve_range=True)
else:
        output = resize(data,
        (int((min_w*in_aspect) + 0.5), min_w),
        preserve_range=True)

# difference in width and height
dh = min_h-data_resize.shape[0]
dw = min_w-data_resize.shape[1]

# put the resized image in center of output image
data[   dh/2:data.shape[0]-(dh/2),
           dw/2:data.shape[1]-(dw/2)   ] = output

return data
```

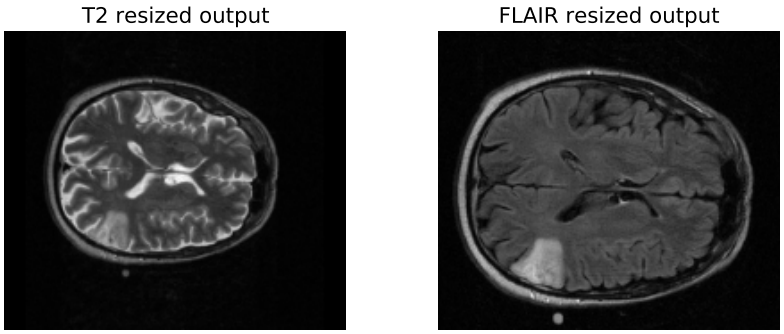

FIGURE 2.4: Resized slices of T2 and FLAIR

Now we have unified data, so that each slice of T2 corresponds to the same slice of FLAIR. In order to keep aspect ratio in some cases I had to center the scaled image. Even though we have images of the same size, heads can vary as shown in Figure 2.4.

The easiest way to achieve it is to segment the skulls on each image and match their size. They have the same aspect ratio already. The background of the image is not constant, there is a lot of noise there. I'm using histogram (Figure 2.5) to threshold the image and create the binary version of it. Most of the low intensity pixels in the image belongs to the background, presented in the first peak in the histogram (first red dot in Figure 2.5). Another peak (second red dot) corresponds to the white/gray matter in brain. I extracted a local minimum between those points (marked as green point).

This values is used to threshold the input image. Thresholded image is run through removal of connected components smaller than the specified size (the circle visible in Figure 2.5). Last step is to surround the skull using Convex Hull. It is smallest convex polygon that surround all non-zero pixels in the image.

Binary mask allows to measure properties of brains in T2 and FLAIR. I am obtaining additional information about the skull, such as the geometric center and the figure orientation. That information

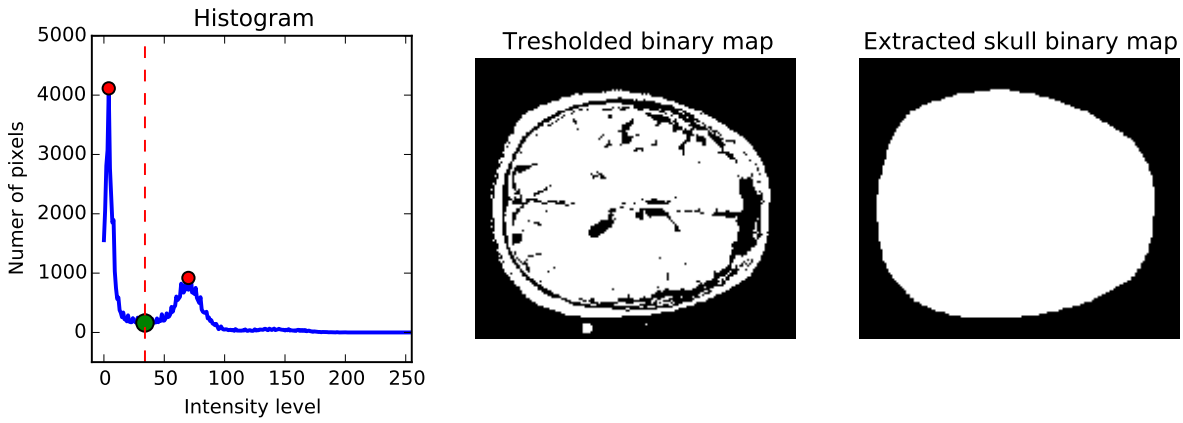

FIGURE 2.5: Process of skull extraction

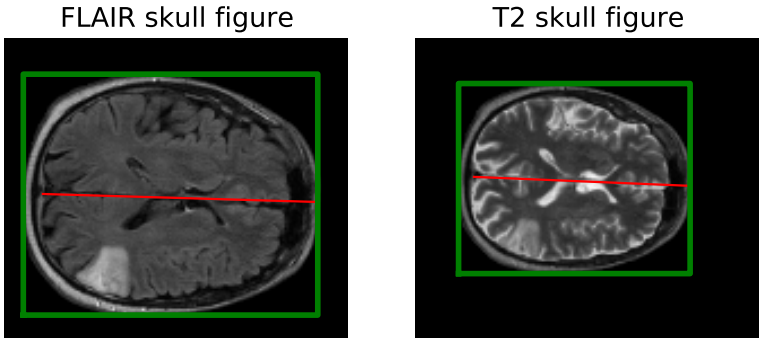

FIGURE 2.6: Skulls properties in FLAIR and T2

will be used later on. Similar approach as previously is used. I look for bigger box (T2 in shown in Figure 2.6) to fit the smaller one. I resize the skull from FLAIR image to shape of the skull in T2 image. The result of this process is visible in Figure 2.7. Using morphological dilation, I am also able to remove the skull. In the literature, cases often had data sets with skull already removed. I will have a choice to use both and find out if it is helpful or not.

Next step of pre-process is to normalize data. As I mentioned before, scans were produced using different machines. Normalized data is a value in range $< 0, 1 >$ based on following formula shown in Equation 2.1.

$$\forall x \in [0, width], \forall y \in [0, height].O_{(x,y)} = O_{(x,y)}/maxO \tag{2.1}$$

The last problem to be resolved was noise reduction. I tested Gaussian filter and median filter. Median filter is less sensitive to outliers, therefore it preserves edges. It is a fast method, that iterates through the image pixel by pixel, replacing its values with median value of its neighbors.

Effect of the median filter applied on FLAIR image is shown in Figure 2.8. The characteristic visible on the histogram after the filtering (bottom left in Figure 2.8) will increase effectiveness of segmentation.

2.4 Segmentation

Image segmentation is a process where labels are assigned to each pixel of an input. Labels (categories) corresponds to different objects (or parts of them) in the image. The most common solution is to distinguish them based on intensity of individual pixels and their neighbors.

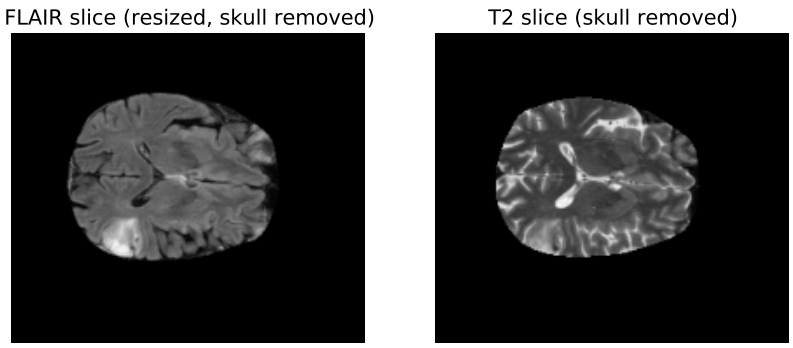

FIGURE 2.7: Extracted figures in FLAIR and T2

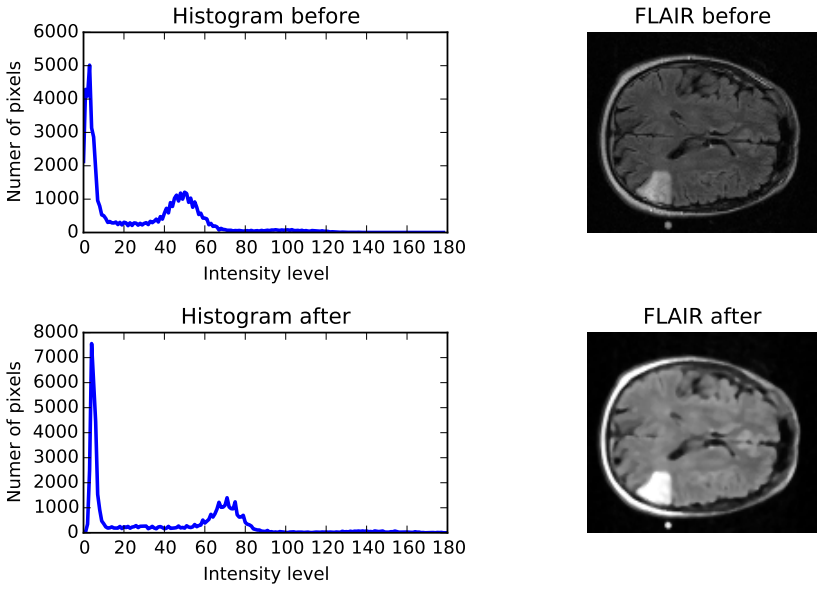

FIGURE 2.8: Median filter effect on image histogram

Unified approach for images segmentation has not been found yet. When new concept is introduced it's important to point out restrictions of proposed solution. In my case solution is built only for brain tumors viewed from axial view. Moreover I do require an T2 and FLAIR scans due to its unique mapping. T2 can not be replaced with T1 or T1C scan, as they are similar to FLAIR in their tumor intensity.

The segmentation process is split into three steps. **These steps are:**

- K-means based segmentation
- Symmetry based segmentation
- Selection of clusters based on combined approaches

2.4.1 K-means based segmentation

K-means is one of the most popular clustering algorithms, it is a unsupervised vector quantization method used as partitional clustering algorithm. Mainly used to find groups of data which are similar to one another based on objective to minimize sum of squared error in given data set. It is wildly used as a fast technique for medial image segmentation [AMEAA15] [Moh+13], with computational complexity of $O(iknd)$ where n - number of elements, k - number of clusters, i - number of iterations and d - the dimension of data.

K-Means clustering steps:

- Initialize with K (number of clusters) and set of n points $X_1 ... X_n$
- Random or k-means++ centroids initialization $c_1 ... c_K$
- Until convergence repeat:
  - for each point x_i:
    - find nearest centroid c_j (using Euclidean distance)
    - assign the point X_i to cluster j
  - for each cluster $< 1, K >$:
    - c_j = mean(X_i)
- Finish when cluster assignment hasn't change

I decided to use K-Means based on recommendations in literature and overall complexity time. K-Means implementation is straightforward, it works for any dimension. K-Means does not takes pixels position into consideration. In case of the 2D or 3D image it does require the reconstruction phase and conversion of an input image into the input vector. Slices of following layers are flatter and combined in the single vector. I am using two vectors, one vector with the intensity level and another with the corresponding position. Only the vector with the intensities is used as an input for K-Means. The vector with the position is used to reconstruct the K-Means results. The code of this process is shown in Listing 2.3.

LISTING 2.3: Convert an input array into K-Means input vector

```
def image_to_coordinates_with_intensity(i, i2):
    """Creates an array of position and intensity

    :param i: input image FLAIR
    :param i2: input image T2
    :return: input vector for K-Means
                        and image matrix
    """
    dim = tuple(image.shape)

        # store pixel intensity
    row_list = []
    # store pixel position in the image
    row_list_coord = []

    # iterate through all dimensions
    for dim in itertools.product(*map(xrange, dim)):
        # take an absolute value of intensity
        pixel_intensity = np.absolute(image.item(dim))
        row_list.append(pixel_intensity)
        # store the position of an element
        row_list_coord.append(dim)

    # an input of K-Means
    output = np.vstack(row_list)
    # matrix used to receive an image out of K-Means result
    output_coord = np.vstack(row_list_coord)

    return output, output_coord
```

I use *KMeans* from *sklearn.cluster* package. It is a *Python* implementation, that by default uses k-means++ algorithm to chose the initial values for centers, to speed up convergence. Based on the manual analysis of sufficient number of slices, I decided to segment into 5 clusters. It seemed to be an optimal

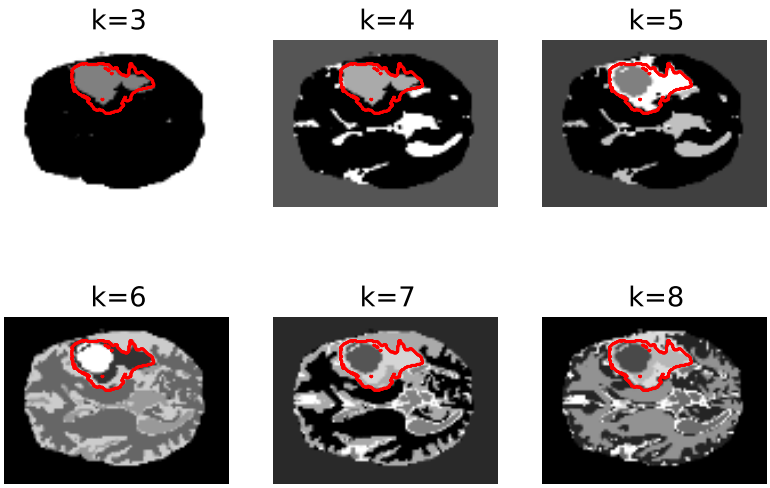

FIGURE 2.9: K-Means results for n clusters

value for my data set. With smaller K small tumors were ignored, when K (number of clusters) was larger than 6, I had to deal with over segmentation problems (Figure 2.9).

In my case K-Means was used on 3 dimensional array (z axis - number of slice, x - width of slices, y - height of slices). Clusters were persistent on each slice, therefore it was much easier for further segmentation - to extract the tumor. For the individual segmentation of each slice separately, cluster labels differed between them. Unfortunately in most cases, I was unable to fit K-Means computation into the memory. For 3D arrays of 512x521 and average 40 slices, I had over 10 millions intensity levels (each represented as a float). I had to limit the range of a slices with ROIs selection. For all 27 cases, I established an area of interests in form of a range of slices (average 20). The tool, I created for that is described in Section 4.4. Figure 2.10 shows results of K-Means algorithm run on the *sample 2*.

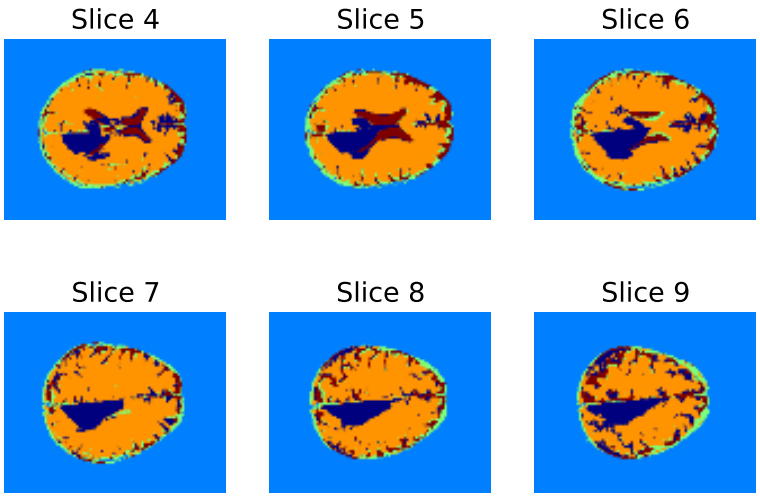

FIGURE 2.10: K-Means on 3D array, 6 following slices

Optimization of K-Means is performed in Section 2.6.

2.4.2 Symmetry based segmentation

K-means segmentation worked very well for most of the cases. I had to find a way to distinguish cluster that marks tumor from the rest of them. In the literature they tend to pick low number of clusters, so that they can pick the tumor by the elimination. In my case I had 5 different clusters, therefore I needed more sophisticated solution. Of course, I was able to eliminate at least one label - the background of the scan, although it was not enough. I tried to eliminate them based on average intensity level in T2 slices. In all the T2 scans tumor was very intensive, therefore it should allow to narrow search space. However, it did not worked out. In the T2 scans blood and cerebrospinal fluid are very intensive as well. I had no clue how to distinguish the blood from the tumor later on. I did not find anything helpful in the literature.

I decided to use another segmentation technique, that I came across. I could combine them and use them for the cluster selection. In their work they were using symmetry analysis and deformable models for brain tumor segmentation [KCB07]. The proposed approach was based on selection of asymmetric areas with respect to the symmetry plane of the brain. After implementation part was done, I decided to confront this idea to healthily regions of the brain. I wanted to find out, if the solution will have clean result in healthy parts of the brain. The Figure 2.11 is an example of symmetry analysis on the healthy slice. The result of the segmentation had few low intensity, small areas.

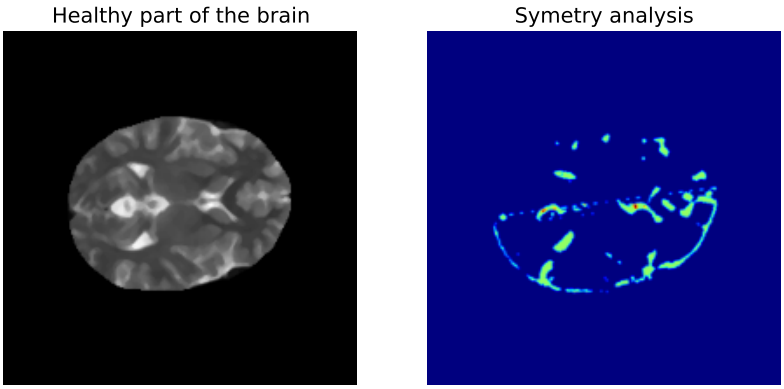

FIGURE 2.11: Healthy part of the brain symmetry analysis

Using the same approach as in pre-processing I was able to find the geometrical symmetric line of skull. In practice it is also the symmetrical line of the brain.

First step was to extract the skull from the input image. To reduce the intensity to the binary level. Authors suggested to use Otsu method, where algorithm is searching for the threshold that minimizes the variance [KCB07]. It simplified previously implemented process, where I had to build histogram and based on two following local maxims, find the local minimum. The result is shown in Figure 2.12.

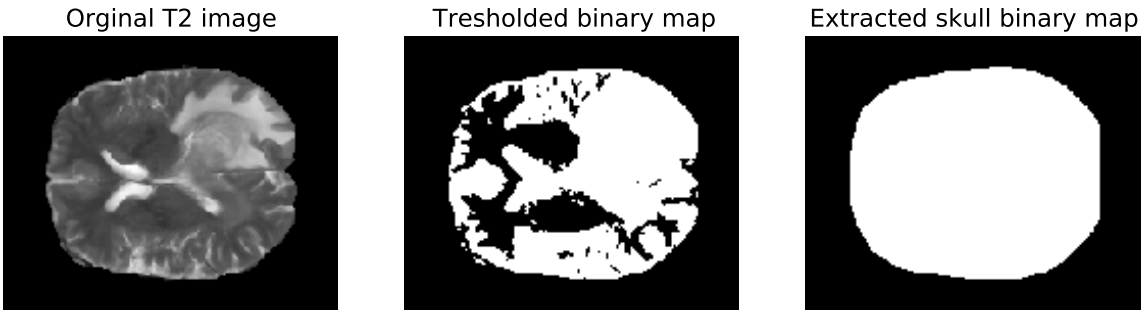

FIGURE 2.12: Skull extraction for symmetry analysis

Now using *skimage regionprops* method from *measure* package, I was able to extract the center of the figure and its orientation (α). The measurements were calculated based on properties of the skull. Then using the cos and sin functions, I can calculate the symmetry line of the object (as shown in Listing 2.4).

LISTING 2.4: Calculate symmetry line of the object

```
# extract region properties
prop = regionprops(label_img2)[0]
# center of our figure
y0, x0 = prop.centroid
# degree of figure orientation
alpha = prop.orientation

# calculation of symmetry line (x1, x2) (y1, y2)
x1 = x0 + math.cos(alpha) * 0.5 * prop.major_axis_length
y1 = y0 - math.sin(alpha) * 0.5 * prop.major_axis_length
x2 = x0 - math.cos(alpha) * 0.5 * prop.minor_axis_length
y2 = y0 + math.sin(alpha) * 0.5 * prop.minor_axis_length
```

The skull orientation α (see Figure 2.4) varied between 5 to 25 degrees. It does hinders the access to hemispheres of the brain by making operations such as subtraction difficult to implement. I assumed that, the image is α degrees askew, where α is a degree of the skull orientation. To fix it, I had to rotate the image the α degrees, which I did using *interpolation.rotate* function. Result of the separation is visible in Figure 2.13. The right part of the hemispheres is already rotated so that, it is ready to subtraction process.

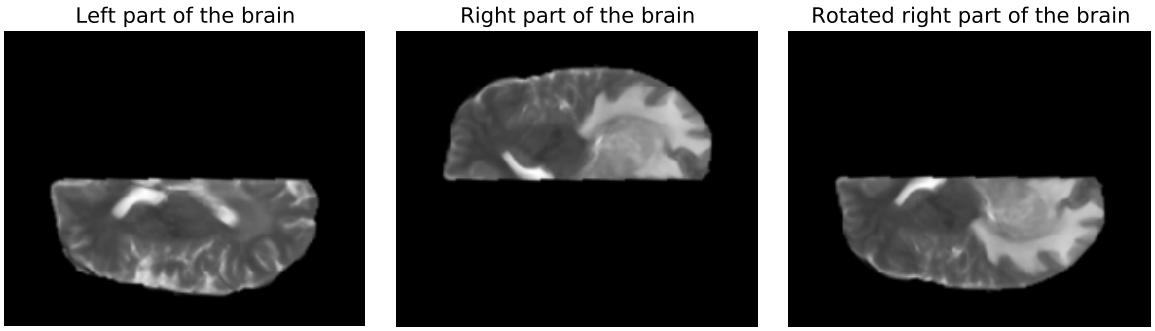

FIGURE 2.13: Result of hemispheres of the brain separation

Hemispheres subtraction is executed with *extract_diff* function (Listing 2.5). It is almost like regular subtract with one important difference. It's only subtracting non-zero values. Therefore if one hemisphere is slightly bigger (as a result of displaced symmetric line) the edges will not be highlighted. Subtraction is performed twice. Once left hemisphere from the right, and second one in reverse order. In order to explain it I have to highlight one more thing in Listing 2.5. Penultimate line is the part where I am restricting the results only to positive values. It allows me to control the result, when I am subtracting A from B with a restriction of only positive values, it will return parts of an A, that are missing on B. The results of subtractions is shown in Figure 2.14.

LISTING 2.5: Subtraction of matixes

```
def extract_diff(a,b):
    assert a.shape == b.shape

    # output matrix - initialize with zeros
    diff = np.zeros(a.shape)

    # subtracts only non-zero values
```

```
for (x,y), value in np.ndenumerate(a):
    if a[x,y] != 0 and b[x,y] != 0:
        diff[x, y] = a[x,y] - b[x,y]

# restrict the diff to positive values only
diff[diff<0] = 0
return diff
```

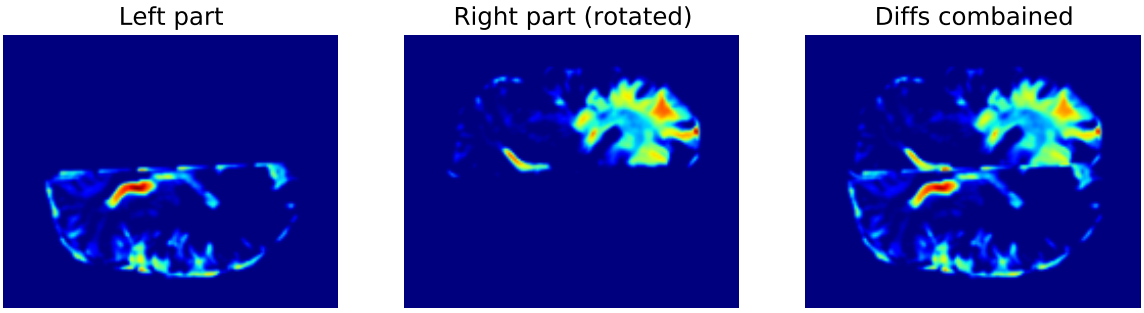

FIGURE 2.14: Diffs of hemispheres subtraction

I end up with a figure with values, that corresponds to intensity differences in asymmetric regions of the brain. Since it is a segmentation, I have to cluster the results. Received image is really clean, therefore there is no need for sophisticated methods. I am using Otsu threshold combine with *label* function from *skimage.morphology* package.

Otsu function was used in previous section, in the skull extraction process. This time around, I am using it again to minimize the intra-class variance. This process results in a cleaned-up binary mask of the asymmetric regions (first plot in Figure 2.15).

Then *label* returns labeled connected regions of an input array (second plot in Figure 2.15). Algorithm is looping through the image and looking for each pixel neighbors (8 pixels that are surrounding iterated pixel). If they are neighbors and have the same value, pixels are connected - labeled as the same cluster.

Regions smaller than constant value (value is calculated based on image size) were removed using *remove_small_objects* function. Removal criterium is based on figure area.

In Figure 2.15 tumor area is labeled with blue color.

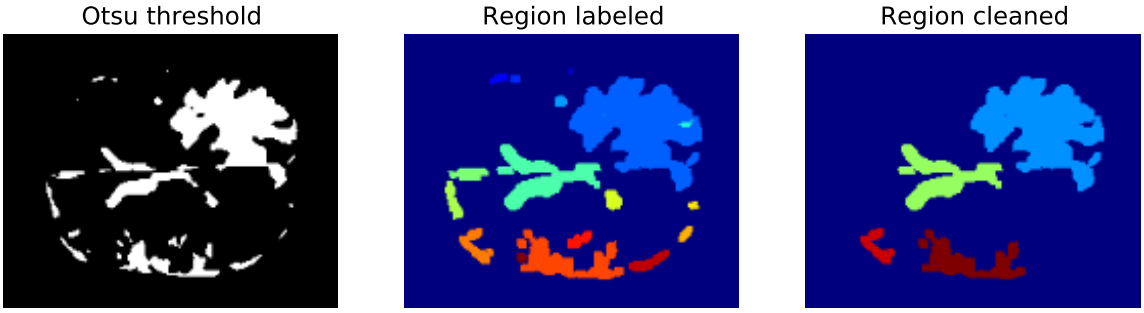

FIGURE 2.15

2.5 Selection of clusters based on combined K-Means and symmetry analysis

This section combines previously shown processes. It is designed to carry entire segmentation process as well tumor area identification in K-Means segmentation.

After K-Means segmentation and symmetry analysis segmentation is run for given input, full search through combined results is performed. For each symmetry analysis clusters, I'm iterating through K-Means clusters. Each pair is scored (as shown in Listing 2.6). Score is determined by *adjusted_rand_score* from *sklearn.metrics* package and combined median value. The median value is used as a threshold to eliminate dark regions (in both T2 and FLAIR image). Score is calculated based on similarity measure between two clusters. The adjusted Rand score is modified version of the Rand measure. For binary input returned value is depended on number of pairs that are in the same cluster and number of pairs that are not. It returns 1 for two identical clusters, 0 to random labeling, and -1 it there is not a single common pixel.

LISTING 2.6: Score pairs of clusters

```
# iterate through symmetry analysis clusters
for sc in symmetry_clusters:
    # iterate through K-Means clusters
    for kc in kmeans_clusters:
        # calculate median value of combined clusters
        #  and score their positional similarity
        results.append(
                {'median': sc['median']*kc['median'],
                'score': adjusted_rand_score(sc, kc)})

# get top results
top = sorted(results,
          key=lambda k: k['score'], reverse=True)
```

Last step is to choose a winner - K-Means cluster with the best score. Sometimes, K-Means clusters the tumor core as a separate label. It happens when tumor core is surrounded by edema. In those cases, I had to choose cluster that's inside of best cluster. In those cases, two clusters instead of one were chosen (see Figure 2.16).

Final results of segmentations along side with selected clusters are shown in Figure 2.16. This technique was used for each of 27-th samples. K-Means was applied on 3D array (build with ROI of patient brain), asymmetry of hemispheres was executed on each slice separately.

2.6 Segmentation optimization

Model built in Section 2.4 was fully unsupervised. In order to validate it I have acquired segmented samples (Table 2.3) from BraTS 2012 edition. BraTS is a Multimodal Brain Tumor Segmentation Challenge organized by MICCAI Society (The Medical Image Computing and Computer Assisted Intervention Society). Samples supplied in their competition had the grand truth table with them. However there was no information about the cancer type. Without it, I could not include them to original data set. I am using them only in this section, to optimize model and measure its results.

Pre-process and symmetry analysis segmentation were hand crafted. There is barely any parameters to optimize there. I will focus on the K-Means optimization and its alternatives.

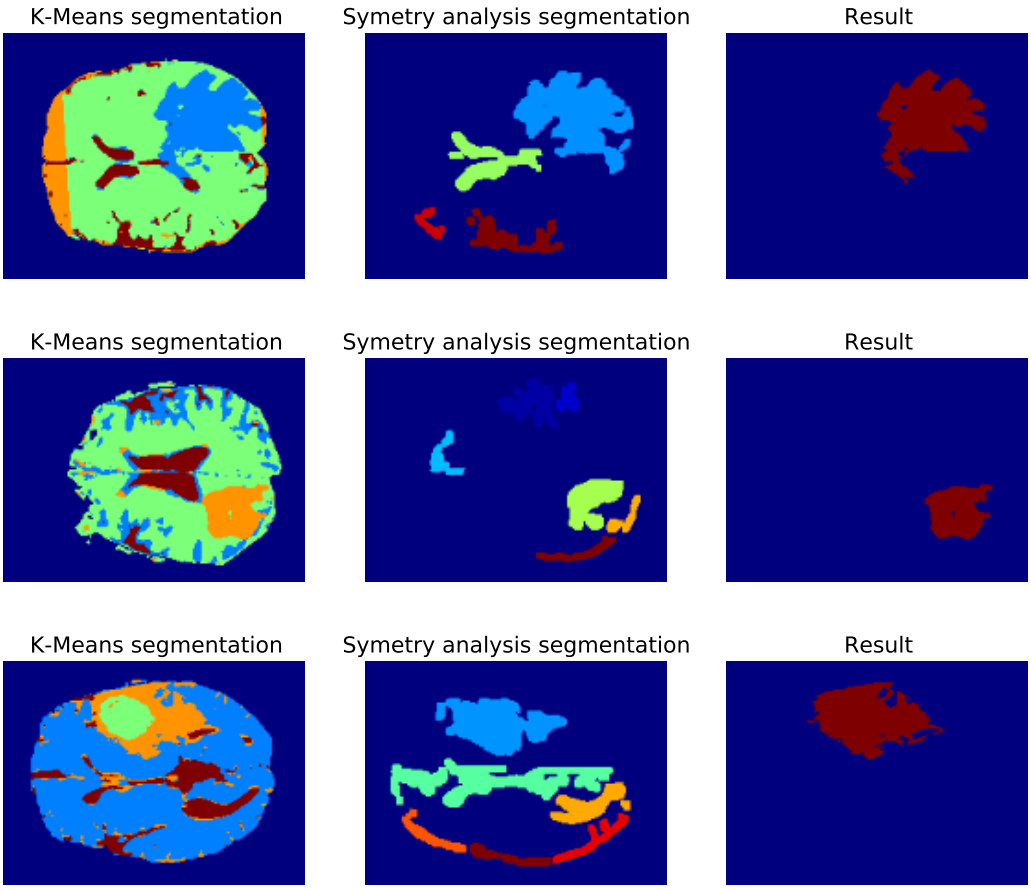

FIGURE 2.16: Segmentation with results

TABLE 2.3: List of used samples

PATIENT No.	MRI SCANS	DIAGNOSIS
1	FLAIR, T2, T1, T1C, Truth table	-
2	FLAIR, T2, T1, T1C, Truth table	-
3	FLAIR, T2, T1, T1C, Truth table	-
4	FLAIR, T2, T1, T1C, Truth table	-
5	FLAIR, T2, T1, T1C, Truth table	-
6	FLAIR, T2, T1, T1C, Truth table	-
8	FLAIR, T2, T1, T1C, Truth table	-
9	FLAIR, T2, T1, T1C, Truth table	-

2.6.1 K-Means optimization

K-Means can be optimized using supervised and unsupervised techniques. For the samples used in optimization section, I possess the ground truth tables, therefore, I can score them according to the truth table. For the supervised approach I used Grid Search technique. It is searching a parameter space for the best Cross-validation performance score. **These parameters were optimized:**

- Number of clusters (clusters) [2-10]

- Disk size for median filter (median0) on FLAIR image [0-10]
- Disk size for median filter (median1) on T2 image [0-10]
- K-Means method for initialization (init) [random, k-means++]
- Number of time the k-means algorithm will be run with different centroid seeds (n_init) [10, 100, 200, 500]

LISTING 2.7: Results of Grid Search

```
GridSearchCV took 61 minutes.
Model with rank: 1
Mean validation score: 87.240 (std: 12.607)
Parameters: {'median1': 3, 'clusters': 5,
             'init': 'k-means++', 'n_init': 10, 'median0': 6}

Model with rank: 2
Mean validation score: 87.201 (std: 12.690)
Parameters: {'median1': 0, 'clusters': 5,
             'init': 'k-means++', 'n_init': 10, 'median0': 6}

Model with rank: 3
Mean validation score: 87.174 (std: 12.747)
Parameters: {'median1': 0, 'clusters': 5,
             'init': 'k-means++', 'n_init': 500, 'median0': 6}
```

Results presented in Listing 2.7 were obtained using 3-fold cross-validation. Estimator score function was based on adjusted rand score from *sklearn.metrics* package. The best accuracy achieved was 87.240 (std: 12.607).

The standard deviation is rather high due to Sample 5 (Table 2.3). At this point, I'm not able to provide a sufficient explanation why tumor has low intensive in T2 image. In other cases tumor was hyperintense. The truth table for BraTS samples were provided by the doctor, who marked the area exceeding area of high intensity. I decided to drop that case from final evaluation. Without Sample 5 I had an accuracy of **88.168%** (std: 5.264) for the tumor segmentation.

K-Means optimization conclusions:

- Number of reruns of different centroid seeds is irrelevant. In case of k-means++ 1-2 initialization was enough.
- As for the initialization technique k-means++ is slightly better than random. Random requires more iterations, because it can easy fall into local minima.
- Number of cluster is the only attribute that really matter. It has an effect beyond standard deviation of measurement. 5 clusters were present in all top 10 results.
- Median filters slightly impacted final score. It seems like T2 is less sensitive, meanwhile FLAIR required higher radius (used to calculate the neighborhood, based on the Euclidean distance).

Another technique, this time unsupervised, is known as Silhouette analysis. Silhouette analysis can be used to study the separation distance between the resulting clusters. It's a visual method that express a measure of how close each point in cluster is to other points in the neighboring clusters. The measure is in $[-1, 1]$ range.

Due to large size of charts I will present only two cases using T2 and FLAIR of patient 3 (Figure 2.17). This process was performed for $K = [3, 4, 5, 6, 7]$ and three different samples. The background cluster was removed, in this process we care about distribution of clusters inside the brain. When the value is close to 1, it indicates that sample is far away from other clusters. If the value is close to 0 it does indicate that sample is close to the boundary of neighboring clusters. Negative value indicates that sample is probably assigned to wrong cluster.

For $K = 4$ (Figure 2.18) we have average silhouette score equal to 0.62. Silhoette coefficients is a value that indicate the distance to neighboring clusters. The green cluster marks tumor area. Some of

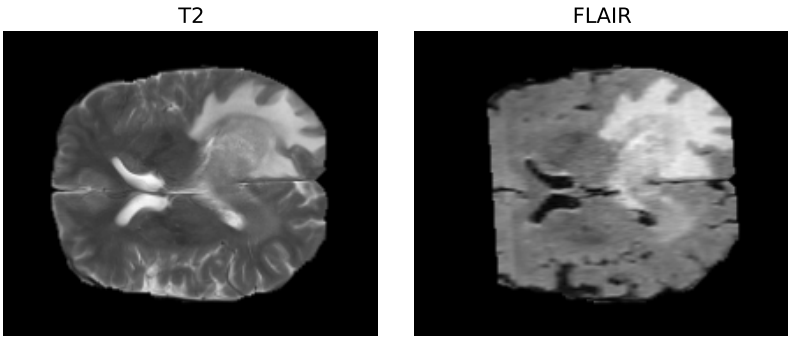

FIGURE 2.17: Sample used for silhouette analysis

the pixels of blue area could be also assigned to tumor, due to negative values visible in Figure 2.17 for cluster *1*. Cluster *0* thickness indicates and structure indicates that there is a place for more than 4 clusters.

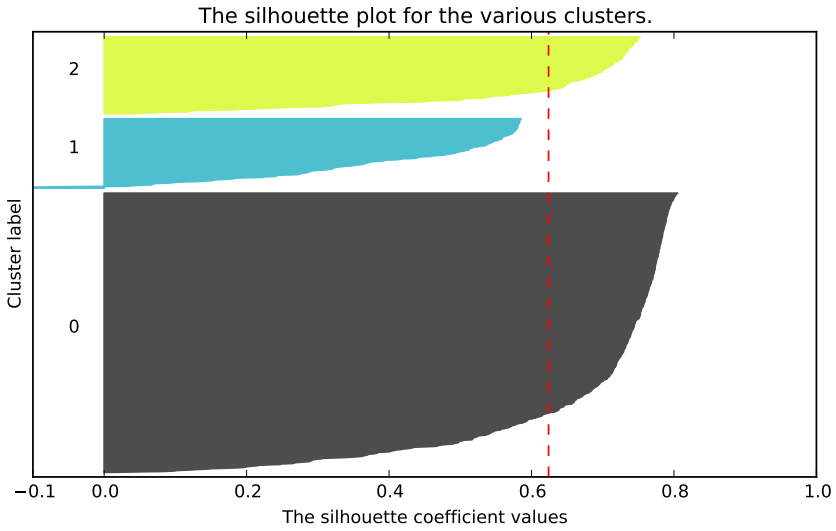

FIGURE 2.18: Silhouette analysis for K-Means(k=4)

In case of $K = 5$ (Figure 2.19) average silhouette score is slightly better - 0.65. Tumor is marked as cluster *1*. Almost all clusters exceeded the average score. Moreover, we got rid of leaking pixels.

Based on silhouette analysis it seems like 4-5 cluster were optimal for tested samples. The supervised Grid Search scored with adjusted rand score confirmed that.

2.6.2 Mini Batch K-Means

Mini Batch K-Means is an alternative implementation that does incremental updates of the centers positions using mini-batches. It's used for large volume of data (such as 10k+ samples).

Mini Batch K-Means is an modified version of K-Means. Its still trying to optimize the same objective function, although mini-batches (sampled from the data) are used instead of the full data. Algorithm was able to receive almost the same results as default K-Means (Figure 2.20). Another property I look into was execution time. K-Means executes for 58s average per sample, meanwhile Mini Batch K-Means took only 29s average. It's almost twice as fast with slightly worse than the standard algorithm score.

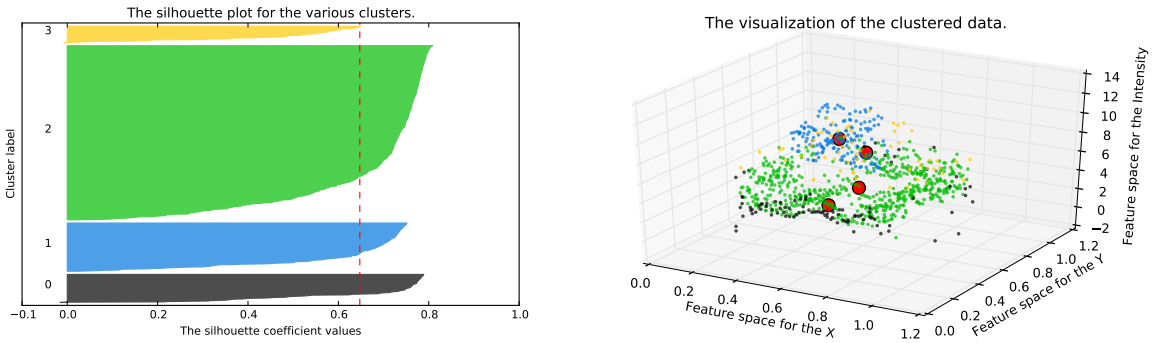

FIGURE 2.19: Silhouette analysis for K-Means(k=5)

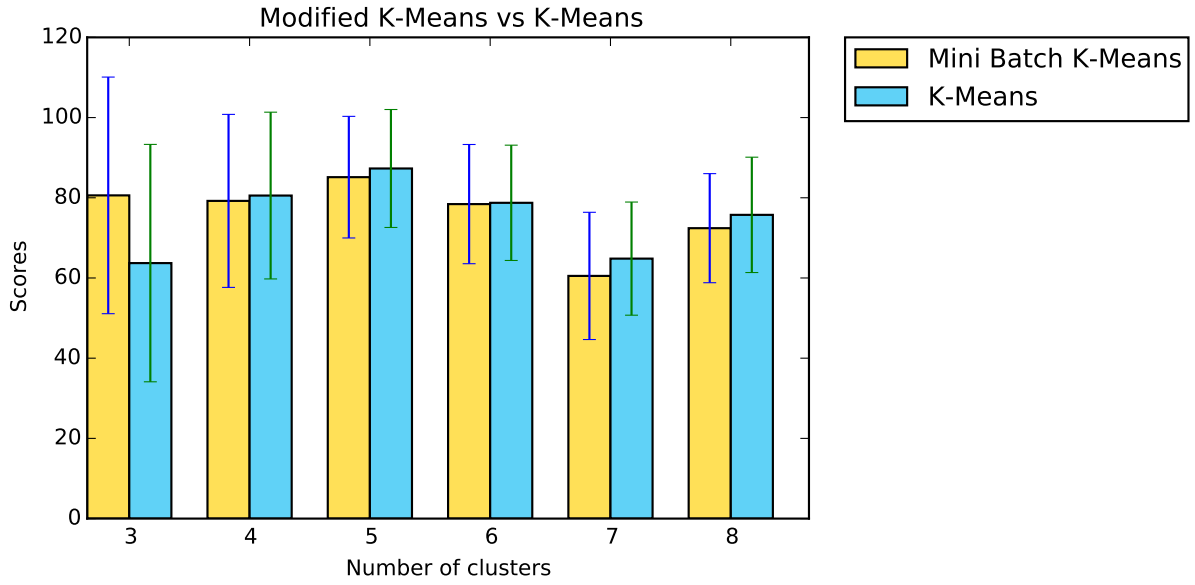

FIGURE 2.20: Median accuracy of the segmentation algorithms in respect to the number of clusters

2.6.3 K-Means with modified input vector

In this subsection I will explore modified K-Means approach suggested by my colleague - Jakub Czakon. In previously implemented K-Means information about pixel position is used only for image reconstruction. Only intensity values from T2 and FLAIR were used. Idea was to attach pixel position as well scaled by intensity importance parameter. This way K-Means will take into consideration distance between pixels. Pixel position is represented by x and y position in input matrix. Algorithm implementation is shown in Listing 2.8.

LISTING 2.8: Function that produce input for K-Means

```
def image_to_coordinates_with_intensity(i, i2,
    ii=1):
    assert i.shape == i2.shape
    w, h = tuple(image.shape)

    row_list = []
    # for each pixel in images (i and i2)
    for (ix, iy) in it.product(*map(xrange, (w, h))):
        # if their intensity is different than background
        if np.absolute(i[ix, iy]) != 0
```

```
            and np.absolute(i2[ix, iy]) != 0:
            # calculate intensity with
            # respect to importance parm.
            pi = ii * np.absolute(i[ix, iy])
            pi2 = ii * np.absolute(i2[ix, iy])
            # build vector for K-Means
            row_list.append(
                  np.array([ix, iy, pi, pi2])
               )

    coord = np.vstack(row_list)

    cs = np.vstack(row_list)
    # normalize positions (0-1)
    cs[:, 0] = cs[:, 0] / np.max(cs[:, 0])
    cs[:, 1] = cs[:, 1] / np.max(cs[:, 1])

    return cs, coord
```

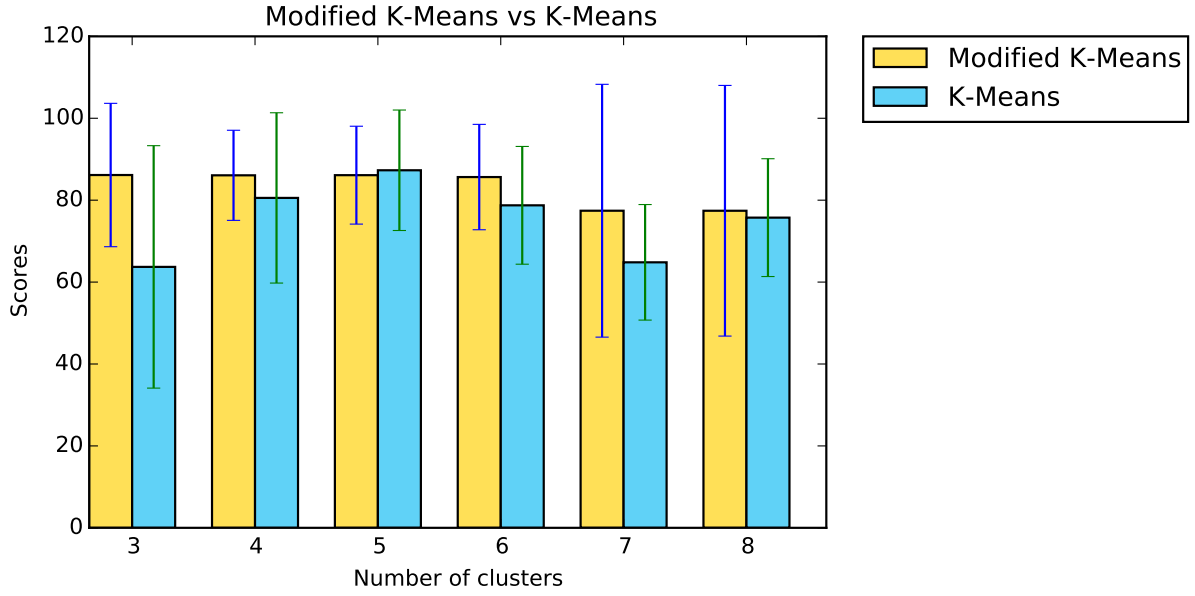

FIGURE 2.21: Median accuracy of the segmentation algorithms in respect to the number of clusters

Modified K-Means scored almost the same results as the default K-Means after optimization. It works well with small number of clusters, therefore it doesn't require additional segmentation to distinguish cluster with tumor. Results compared to default K-Means are shown in Figure 2.21.

2.6.4 Agglomerative clustering

I was trying others clustering techniques such as DBSCAN and MeanShift, although I was unable to receive satisfying score with them. Agglomerative clustering worked out in the first try, scoring almost 70% average. It's a sub type of hierarchical clustering, which is build with "bottom up" approach.

The main goal of algorithm is to ensure that nearby points end up in the same cluster. It doesn't require prior knowledge of number of clusters. Moreover we can apply adaptive stop function, based on tumor metrics. In case of K-Means due to constant number of cluster we can end-up with over segmentation. Another advantage comes with repeatability, unlike K-Means it will always be consistent. K-Means due to random initialization might end up with different results. The bottom neck of this

method is speed ($O(n^3)$ complexity). It's taking cubic amount of K-Means time to calculate on the same input data. The results of Agglomerative clustering combined with K-Means are shown in Figure 2.22.

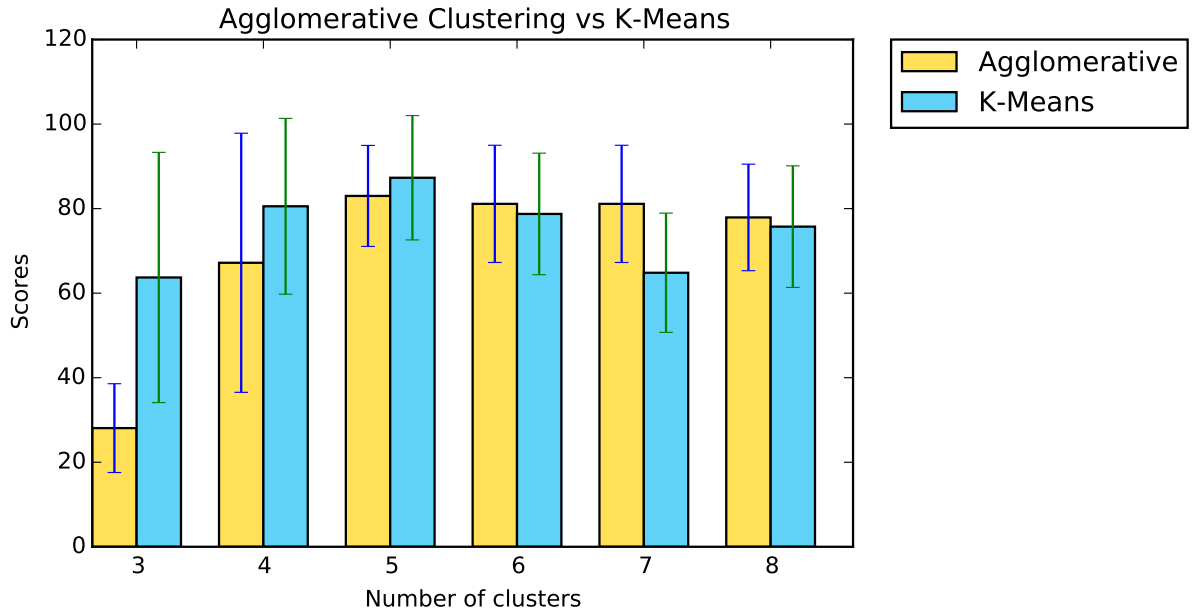

FIGURE 2.22: Median accuracy of the segmentation algorithms in respect to the number of clusters

Figure 2.23 contains few results of the entire process. White line in 2nd and 3rd column corresponds to the contour of a tumor from grand truth table. The difference is especially visible in 3-rd row. This is the 5th sample, I mentioned in subsection 2.6.1. Part of the tumor in K-Means segmentation (green cluster) was clustered with CSF (Cerebrospinal fluid). Even though there is clearly distinguishable border between them in T2 image. It is an effect of Ward's linkage used in Agglomerative Clustering. Ward's method minimizes the within-cluster variance, that is have effect on squared distance between cluster centers. In K-Means it did not really matter, since it is not using any connectivity matrix.

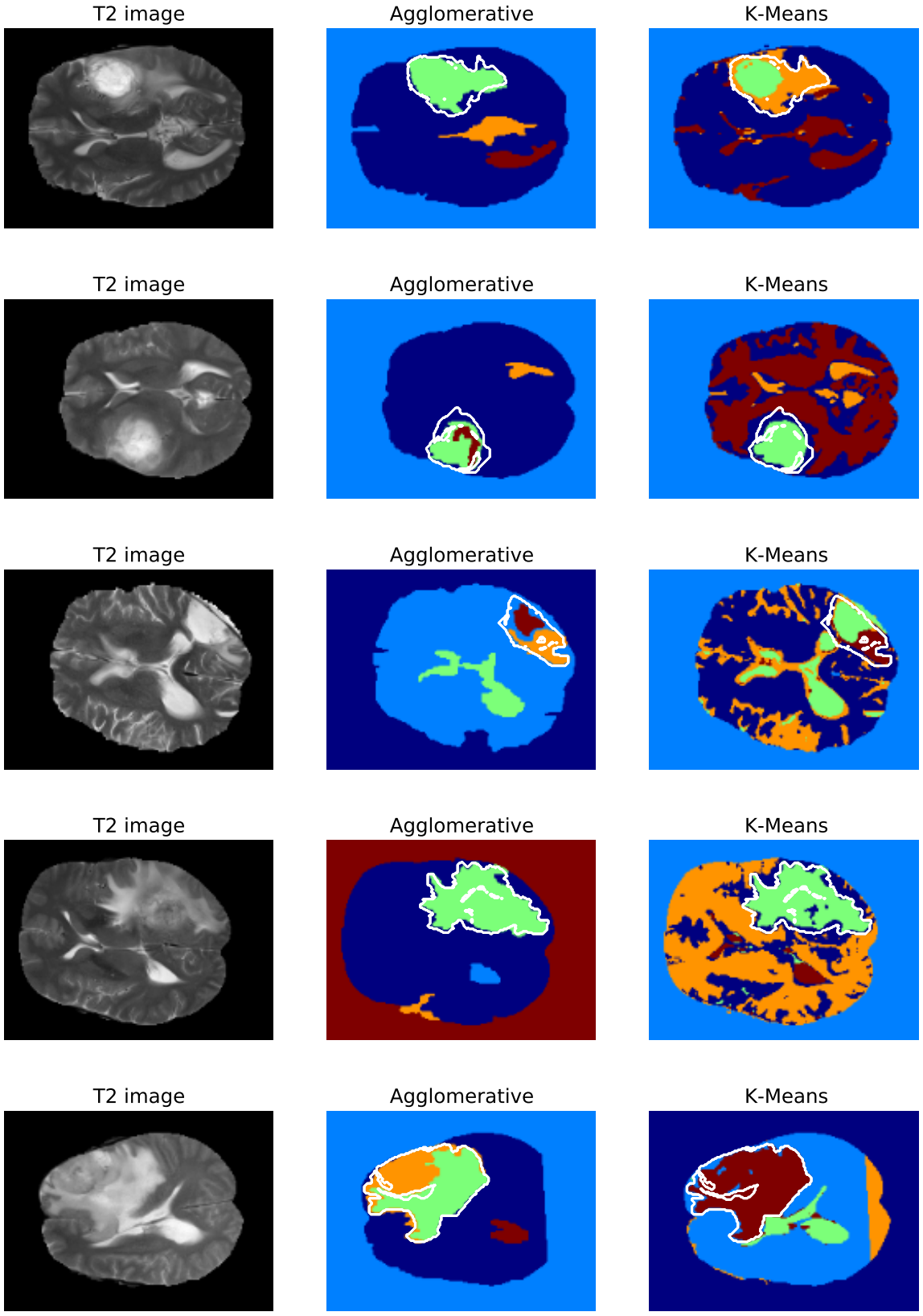

FIGURE 2.23: Results of Agglomerative and K-Means algorithms

2.7 Classification

The objective of this section is to introduce the key concepts behind classification model. Based on segmentation process described in Section 2.4.1 I have extracted tumors from all 27 cases. The number of samples is really low, therefore despite many attempts I was unable to use Convolutional neural network. I decided to stick to standard approach and build a feature vector based on extracted tumor. Segmentation described in previous sections gave me tumor mask for T2 and FLAIR samples. The same mask was used for T1 and T1C images.

2.7.1 Feature extraction

The feature extraction process is a step where informative and non-redundant measures are extracted from the input image. In my case it is a 3D area of segmented tumor. I have extracted as many feature as I could. Then each of them was evaluated to find out about the importance. Main groups of features are based on intensity, position and shape. I used traditional features such as histogram, median, average as well as intentionally designed for brain tumors. For example tumor position, as I mentioned in Section 1.6.1 - "Oligodendroglioma tumors mainly grow in frontal and temporal lobes" [Ass16a].

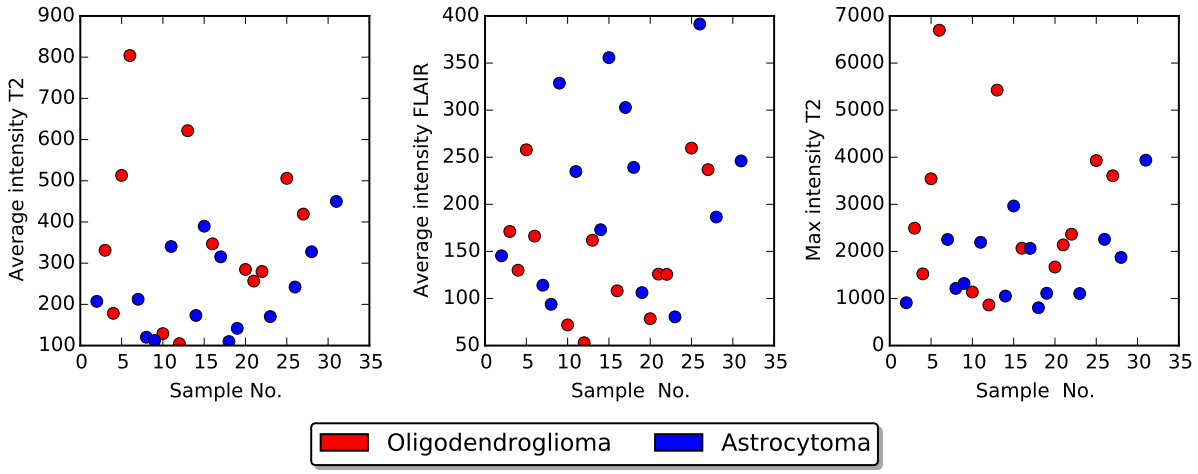

FIGURE 2.24: Selected features extracted from data set

Flowing features were extracted:

- Tumor volume (in mm^3)
- Tumor position (x,y,z) calculated from the middle of the brain
- Average intensity of tumor area in FLAIR, T2, T1, T1C
- Median intensity of tumor area in FLAIR, T2, T1, T1C
- Standard Variance of tumor are in FLAIR, T2, T1, T1C
- Max intensity of tumor area in FLAIR, T2, T1, T1C
- Min non-zero intensity of tumor area in FLAIR, T2, T1, T1C
- Variance of intensities in FLAIR, T2, T1, T1C
- 8 bins of intensity histogram (FLAIR, T2, T1, T1C)
- Skewness and kurtosis from histogram values
- Feature extracted from the middle slice of the tumor
  - Eccentricity - the ratio of the focal distance over the major axis length

- Area * Extent - Area scaled by extent of tumor bounding box
- Solidity - radio of pixels that are tumor to pixels build using convex hull image
- Perimeter

Tumor volume was calculated by following formula $V_pix = \sum_{i=0}^{slices} A_i * h_i$. Where A is an area of slice and h is height calculated based on difference in z position. using the values provided in *pixdim* header of NIFI file (explained in 1.7), I was able to calculate volume in mm^3.

2.7.2 Attributes evaluation and selection

We have multiple methods available for the feature selection process. This process is designed to select the subset of features. It is done to simplify the model, improve the generalization (by over-fitting reduction) and shorten the time of training.

I had 59 features extracted in the previous section. To reduce that number, I used variance threshold. It removes all the features, that have a variance below given threshold. In my case, I removed features that had the same value in more than 90% of samples. I end-up with 29 out of 59 features.

Another evaluation that is used for the feature selection is to measure linear correlation for the pairs of variables. Pearson correlation coefficient gives a value between $+1$ and -1, where $+1$ is a perfect positive correlation, -1 is a negative correlation. The issue with Pearson correlation is that it is not sensitive to a non-linear relationship. Moreover correlation value can be misleading (i.e. one outliner can produce correlation, even though relation is not linear).

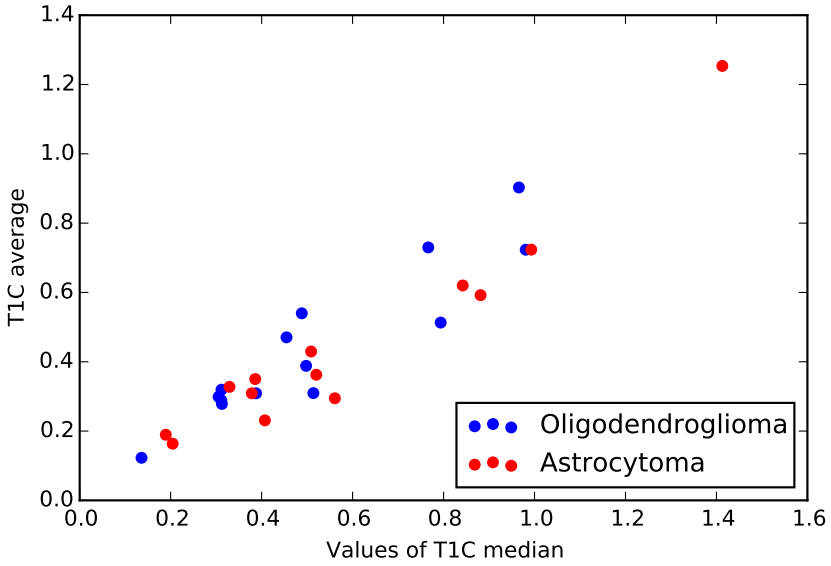

FIGURE 2.25: Scatter plot of t1c average and t1c median intensity values

The attributes that had high correlation, in most cases were obvious one (i.e average and median - shown in Figure 2.25). The average value and median of T1C scans had the Pearson correlation coefficient of 0.943. In the presented chart we have 27 samples, Oligodendroglioma marked blue, Astrocytoma marked red.

Another worth mentioning result of the correlation is 5-th and 4-th bin of FLAIR histogram. The Pearson correlation coefficient of 0.735. Based on the plot (Figure 2.26) it is also clear, that the split of this parameters will bring good results.

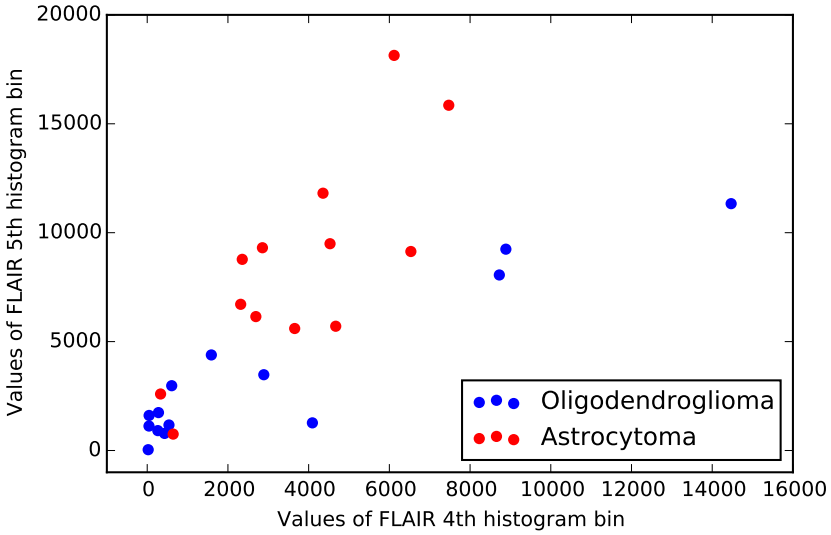

FIGURE 2.26: Selected features extracted from data set

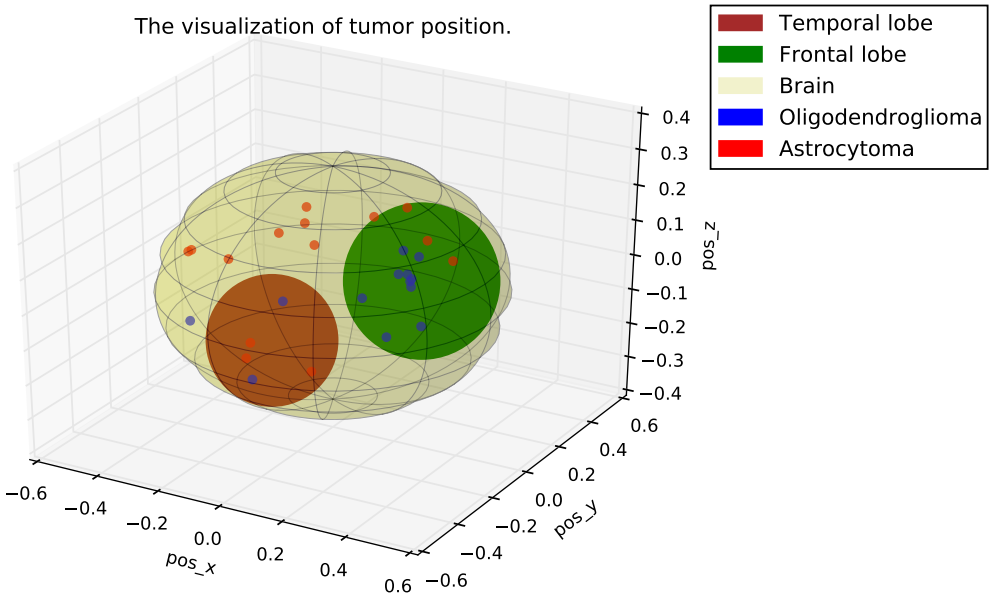

FIGURE 2.27: Tumor positional features

Last one that I paid attention to was tumor position *(pos_x, pos_y, pos_z)*. It was the only clue I had from the theoretical point of view. Oligodendroglioma (marked blue) should more often occur in frontal (marked as green) and temporal lobes (marked as brown). As shown in Figure 2.27 it seems like most of the Oligodendroglioma cases is present in the frontal and temporal lobes. The position of the tumor was calculated based on the middle of the brain. Therefore regardless of the brain size, position is normalized. Position of the tumor was calculated as a geometrical center of the tumor figure.

For the final attribute selection I used *CfsSubsetEval* and *BestFirst* form the WEKA application. Weka is a collection of machine learning algorithms for data mining tasks. I used Naive Bayers as a classifier, to estimate the accuracy of attributes. Naive Bayers considers all the features to contribute independently to the probability. The results are presented below (in the Listing 2.9).

LISTING 2.9: Attribute evaluation

```
=== Run information ===

Evaluator:
        weka.attributeSelection.ClassifierSubsetEval
        -B weka.classifiers.bayes.NaiveBayes
Search:
        weka.attributeSelection.BestFirst -D 2 -N 10
Instances:      27
Attributes:     45
Evaluation mode:5-fold cross-validation

= Attribute selection 5 fold cross-validation (stratified), seed: 1 =

number of folds (%)  attribute
  5(100 %)      3 flair_hist_4
  4( 80 %)      4 flair_hist_5
  3( 60 %)     32 eccentricity
  1( 20 %)     37 pos_z
  1( 20 %)     38 distance
  1( 20 %)     39 pos_x
  3( 60 %)     40 pos_y
```

Model build using selected in previous step attributes had an accuracy of 78.017% (std: 13.863).

2.7.3 Random Forests

Random Forests is a classification algorithm. To classify a input vector, Random Forest frows classification trees. Each tree gives a classification, final decision is based on all the votes. I'm using Random Forest due to its properties. Even tough, it does not have the best results, it does not overfit. In addition we can estimate importance of variables used in the classification process. I'm using *RandomForestClassifier* from the *sklearn.ensemble* package.

I have tested feature importance of the features selected in previous section using forests of trees. The plot (Figure 2.28) demonstrate the comparison of informative features (flair_hist_4 and flair_hist_5) to the, less informative (t2_hist_1).

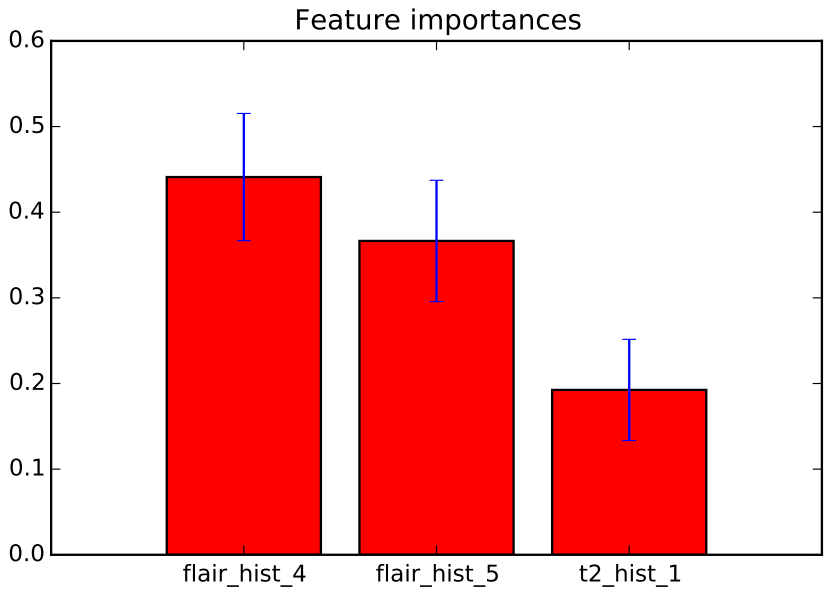

FIGURE 2.28: Selected features importance

Based on the feature ranking following features were selected:

- 4-th bin of FLAIR histogram
- 5-th bin of FLAIR histogram
- Tumor position (x,y,z)

The out-of-bag (OOB) error estimate is the error rate of the out-of-bag classifier on the training set. Random Forest is trained using bootstrap aggregation. We can use it to estimate parameters of Random Forest Classifier. In my case, I'm using it to estimate number of trees in the forest (*n_estimators*).

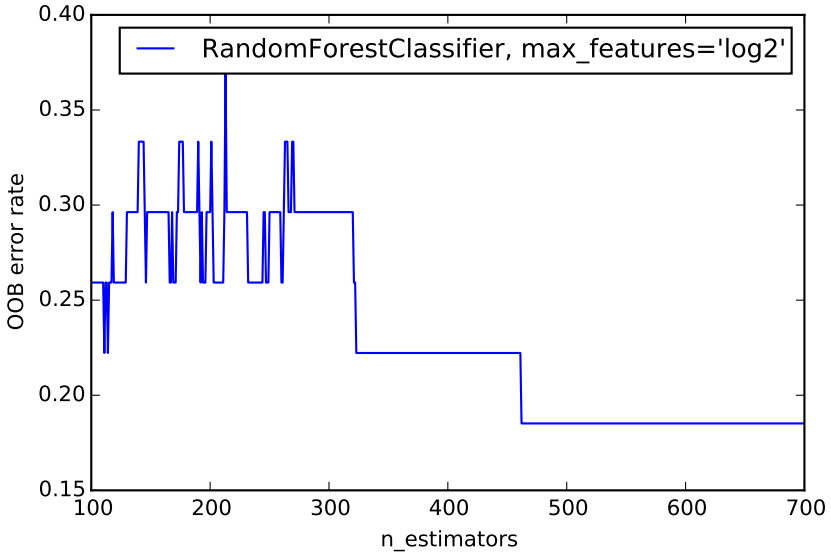

FIGURE 2.29: Selected features importance

Based on error trajectory during training we can estimate that 400-500 trees is a sufficient value.

LISTING 2.10: Final estimator

```
results = []
# X - input vector (pos_x, pos_y, pos_z, flair_hist_4, flair_hist_5)
# Y - expected class

# repeat 100 times
for _ in range(100):
        rf = RandomForestClassifier(n_estimators=500,
                                    max_features='log2')
    cores = cross_validation.cross_val_score(rf, X, Y, cv=5)

    results.extend(cores)

# print results
print np.average(results)*100, np.std(results)*100
```

In 5-fold cross-validation, that was run 100 times, I got average accuracy of **87.0%** (std: 12.991).

2.7.4 Texture Features with GLCM

Texture is another feature that can be helpful in image based classification. Features that were extracted in previous chapter described entire tumor as a whole. In this section I will look into statistical properties

of smaller areas. I am looking for similarities in structures, that are repeated within the tumor class. The GLCM (Grey Tone Spatial Dependency Matrix) is a relationship texture calculation. It is a tabulation of different combinations of brightness frequency. In my case it is a histogram of co-occurring values in selected patches.

First step was to extract the patches for each sample. Based on tumor segmentation performed in section 2.4.1, I was able to extract the position of points that belongs to the tumor area. Points were filtered to only those that does not contain the outer parts of the brain (taking the size of patch into the consideration). Next, 30 positions at random was picked. Patches of 20x20 was extracted from the T2/FLAIR image for each sample. As a result I had 810 tumor patches from Oligodendroglioma and Astrocytoma samples. Each patch had an information about sample number and cancer type assigned. It will be used in following chapter, for the cross-validation. Entire process was evaluated on T2 and then of FLAIR scans.

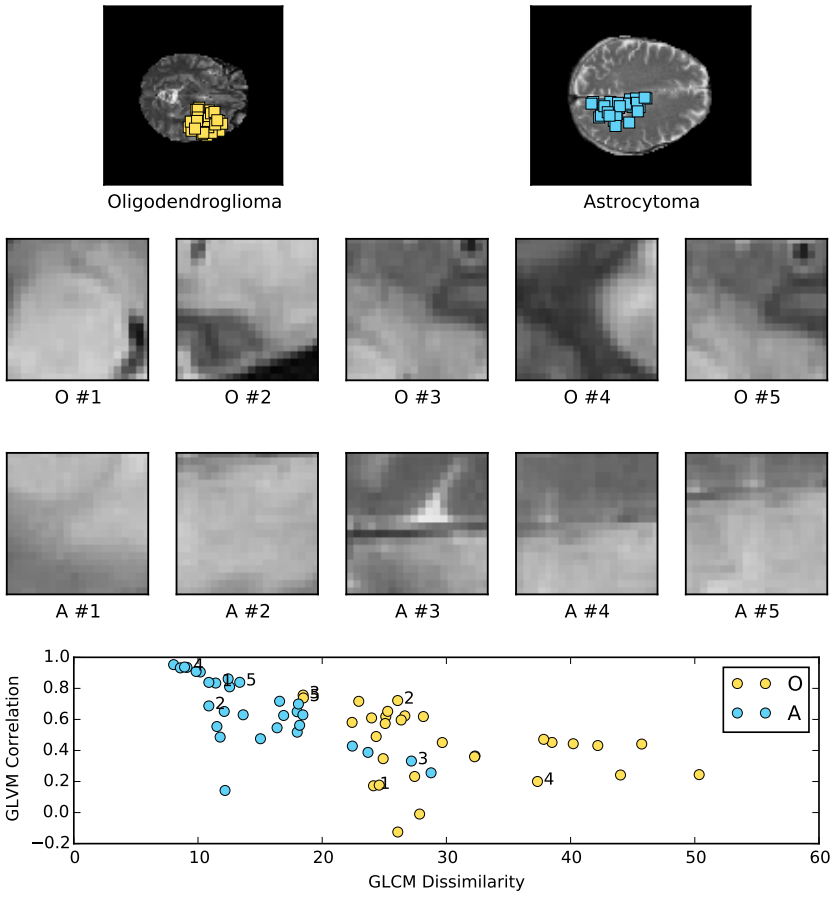

FIGURE 2.30: Co-occurence matrix features for Oligodendroglioma and Astrocytoma

To find out if there are texture features dissimilarities in the different types of tumors, I used two samples of each tumor type (Figure 2.30). In the Figure 2.30 only first 5 features were shown, although from the plot below them it is clear that there is correlation between them. We could simply distinguish them using linear function dissimilarity attributes were calculated.

For the implementation of GLCM, I used *skimage.feature* package. Function *greycomatrix* that calculates the co-occurence matrix was used with offset of 5 pixels, in addition symmetric occurrences were accepted. Using greycoprops function correlation (2.2) and dissimilarity (2.3) were calculated. P - co-occurence histogram, i - gray level i, j - gray level j. In addition contrast, homogeneity, and energy were calculated and extracted for classification model.

$$\sum_{i,j=0}^{255} P_{i,j}[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{\sigma_i^2\sigma_j^2}}] \tag{2.2}$$

$$\sum_{i,j=0}^{255} P_{i,j}(i-j)^2 \tag{2.3}$$

Using properties calculated in this section (contrast, dissimilarity, homogeneity, energy and correlation).For the model fitting I used 5-fold with non-overlapping labels. Therefore patches of patient that was used for training was excluded from testing. Otherwise, I would end up with false results, because patches from the same patient could be used in training and testing process. As for the model, I used SVM, Gaussian Naive Bayes, and Logistic Regression.

In 100 runs of Gaussian Naive Bayes (5-fold), I got average accuracy of 42% (std: 10). I was unable to achieve scores better than a coin toss. There is no indication in the Figure 2.31 of learning whatsoever.

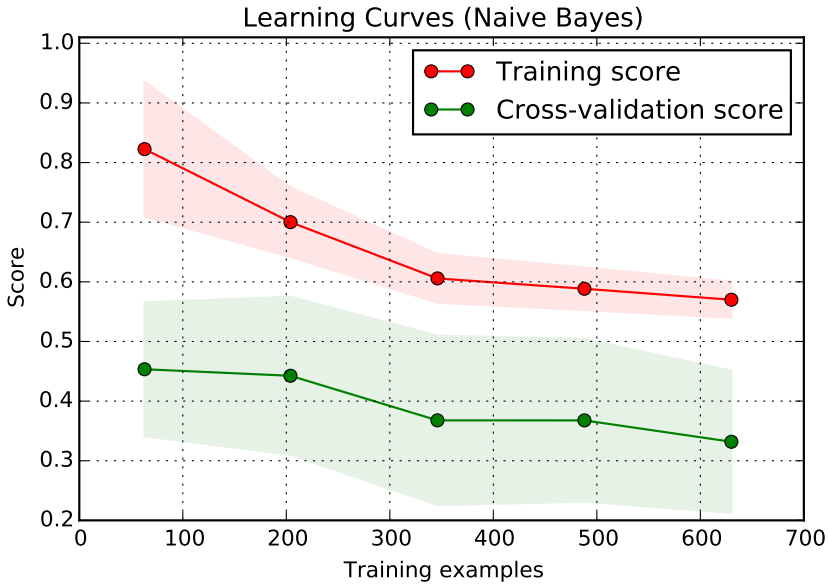

FIGURE 2.31: Learing Curves for Naive Bayes

2.7.5 Local Binary Pattern - texture classification

In previous subsection, I was unable to use GLCM attributes for the classification process. I decided to give it one more try, this time using Local Binary Pattern. This method was successfully used in breast cancer classification with 800 cases [PEC12].

Local Binary Pattern (LBP) is another texture classification technique. LBP works only on gray-scale images, in our case we have an intensity, that also has only one channel. It is a iterative algorithm that tests surrounding points of a central point. Boolean results are assigned based on inequality of surrounding points in comparison to central point.

I used *local_binary_pattern* function from *skimage.feature* package. LBP is used to detect the texture, histogram to look at the distribution. At the end, input data is normalize. Implementation is shown in Listing 2.11.

LISTING 2.11: Calculate LBP histogram from patches of selected patient

```
inputv = []

# patches - patches of one patient
for patch in patches:
    radius = 4
    no_points = 8 * radius
```

```
# LBP image (32 points, radius 4)
lbp = local_binary_pattern(patch, no_points, radius, method='uniform')

# calculate histogram with 33 bins
(hist, _) = np.histogram(lbp.ravel(),
        bins=np.arange(0, no_points + 2),
        range=(0, no_points + 1))
hist = hist.astype("float")

# normalize data
hist /= hist.sum()

# append histogram to the input vector
inputv.append(hist)
```

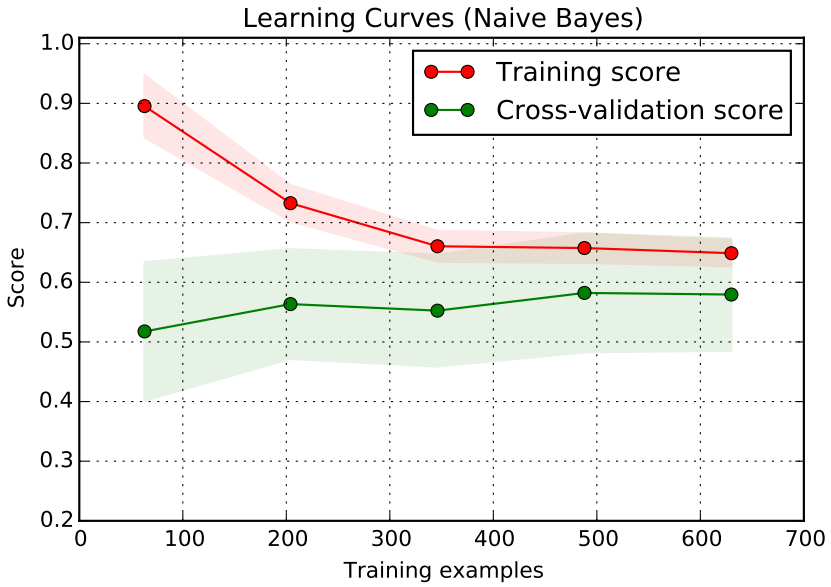

FIGURE 2.32: Learning Curves for Naive Bayes

Once again, I used Naive Bayes to check learning curves for this method. Those are expected characteristics of Naive Bayes. The accuracy score benefits as we increase training data.

For the final classification, I used Logistic Regression fitted in the 5-fold cross-validation. Patches of patient that was used for training was excluded from testing in cross-valid iteration. In 100 runs of the model I got average accuracy of 68% (std: 8).

2.7.6 Summary

In the pre-process MRI slices shape was unified. Based on the head estimation, position of skulls were adjusted. Median filter was used to reduce the noise. The pre-processing was essential for the segmentation and classification processes. Without it I would not be able to combine different scanning types (T2, FLAIR).

K-Means, K-Means with position in the input vector, Mini Batch K-Means, and Agglomerative clustering algorithms were tested for the primary segmentation. Symmetry analysis algorithm was implemented to detect the symmetry disorder between the hemispheres. Combination of symmetry analysis and primary segmentation lead to the accuracy of 89.027% (std: 5.408). Accuracy estimation was performed on the BraTS data set, that had an additional ground truth tables. Original data set, that was used for the classification does not have grand truth. The accuracy was calculated with a similarity measure between truth table and extracted tumor.

Features extracted from the tumor area (selected in the segmentation process), were used in the binary classification between Oligodendroglioma and Astrocytoma. Total 59 features were extracted and evaluated. Final evaluation showed that there is a relation between the tumor position and corresponding class. Oligodendroglioma occurred mainly in frontal (and temporal lobes. The final accuracy score was mainly affected by the intensity distribution in the FLAIR image. Textural patterns proves to be less effective, the average accuracy of the tumor classification was 68% (std: 8). The final Random Forest classification model estimated in 100 5-fold cross validations scored average 87.0% accuracy (std: 12.991).

Chapter 3

Pathology imaging

3.1 Origin of the process

Pathology-imaging based classifier is a separate model, that was built on the same data set. In each iteration of cross validation Oligodendroglioma estimates from the testing set were calculated. Results in form of the probability of sample belonging to Oligodendroglioma class were calculated (Appendix B).

Pathology imaging part was implemented for MICCAI 2015, International Conference on Medical Image Computing and Computer Assisted Intervention. It was carried out by Jakub Czakon, Piotr Giedziun, Grzegorz Żurek, Piotr Krajewski, and Witold Dyrka. The progress and results were published as a Conference Papers [Cza+15] and [Żu+15]. This chapter will cover essential concepts, that are required in my thesis.

3.2 Data set

For pathology imaging, we were provided with the data of the same patients I used in Chapter 2. Instead of brain MRI we got 32 cases of pathology images (Figure 3.1). Images were stored in TIFF format with several different magnification levels. Samples had an information about cancer type corresponding to them. Images of the same class varied substantially (coloring, intensity and size).

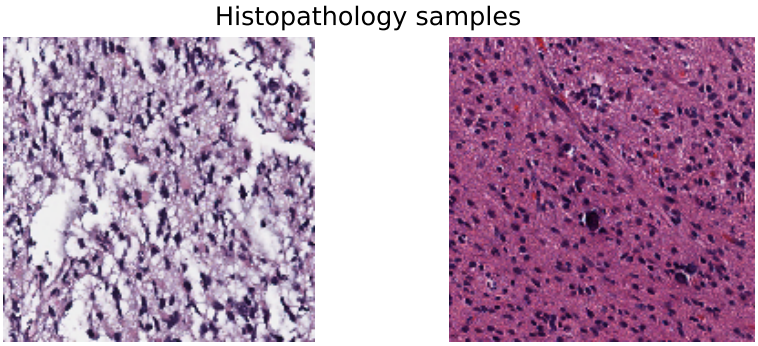

FIGURE 3.1: Histopatology samples

3.3 Process

The first step of algorithm was ROI selection. Selection condition was based on the density of cells. The targeted area must have a density above certain threshold. Then selected ROIs were spited into smaller

frames of 1000 by 1000 pixels each. Pre-processing was carried out by Reinhard's normalization. Each frame was segmented based on intensity levels. Segmentation was performed to extract and count number of cells in each frame. Based on that count, top 50 frames were selected.

Each of the cell from the selected frames along with the margin of fixed size was extracted. Features such as shape, density an others were used to describe extracted patches.

Logistic Regression (LR) and Linear Support Vector Classifier (SVC) were used for the classification. Classifier was train to check given cell was an oligodendrocyte.

Base on the cells probability distribution each frame was classified as Oligodendroglioma or Astrocytoma. This time Gaussian Naive Bayes (GNB) or Logistic Regression (LR) was used.

3.4 Results

Based on the process described in previous section in cross-validation procedure average accuracy reached a **73%**. The result was obtained in SVC+LR as well as LR+GNB.

Chapter 4

Project implementation

4.1 Project

First part of the project was an implementation. Vast majority of the computations were implemented manually. External application was used only for the attribute evaluation. Project can be splitted into three subtasks:

- segmentation and classification implementation
- segmentation and classification analysis
- web application used for the ROIs selection

4.2 Used technology

The project was fully written in *Python 2.7*. I decided to use *Python* due to its flexibility and simplicity. Moreover, *Python* offers wide spectrum of packages for machine learning, data analysis, and image manipulation. As oppose to *R*, *MATLAB* or others data analysis languages *Python* is a general-purpose language. Building a web interface for implemented segmentation and classification was straightforward.

The vast majority of algorithms used in my thesis was already implemented in *Python* packages (mainly scikit and scipy). It allowed me to perform a variety of experiments in a relatively fast paced.

Used technology:

- Python 2.7
- Jupyter Notebook - web-based interactive computing system
- NumPy package - fundamental scientific package
- scipy package - ecosystem for mathematics, science, and engineering
- scikit-learn package - machine learning
- scikit-image package - image processing
- nibabel package - for common medical and neuroimaging file formats
- Tornado Web Server package - WSGI server

4.3 Development

Development was performed on the virtual environment created for machine learning and data analysis related projects. One of the main requirements in case of machine learning is an ability to fit it in memory.

TABLE 4.1: Virtual Machine specs

SDRAM	32 GB DDR4
STORAGE	2TB in RAID 0
CPU	i7-4790K 4.4 GHz
GPU	GeForce GTX 970 4GB GDDR5

Otherwise, data has to be divided into batches and work around fitting procedure had to be created. In case of K-Means, it's required to maintain entire cluster volume in memory.

Then we have a CPU, that translates to overall calculation speed. Last but not least is a hard drive, its performance is also important. I had gigabytes of data that I had to read each time algorithm was executed. Disk speed (write and read) can be increased by the RADIO 0 (two drives with the same capability). Full specs of the machine I was using are shown in Table 4.1.

Most of the development work was done in Jupyer Notebook due to its easily accessible web interface. Then if the code concept was successful I have rewritten in using PyCharm. PyCharm is an IDE for professional development with more advanced features. Such as code analysis used to improve final code quality, debugger to analyze the execution and contextual code completion.

4.3.1 Jupyter Notebook

The IPython Notebook is an interactive computing environment. It allows to create interactive inline plots, live code separated by narrative text and more.

It has three main components:

- The notebook web application - interactive web page and IDE, allows to live code, manage files
- Kernels - processes started by the web application, runs users code
- Notebook documents - files where implementation and narrative text is stored

Each of them is bringing major features helpful scientific development.

The notebook web application allows to access the code from anywhere. As long as we have an internet connection. There is no environment setup required, only modern web browser. We can access full potential of server class machine on standard devices.

Kernels attached to notebook documents are persistence. This feature is one of the most helpful. Let's assume I have two steps - pre-process and then segmentation. Each time I did change something in segmentation there is no need to re-run pre-processing part. There is a workaround using pickle and flags, although it requires additional code. Jupyter is build around this concept, it allows to only execute selected part of the code, while entire scope is intact.

Another worth mentioning feature of notebook documents is that it save each change and its state. We can pick up from where we left, with all the outputs we had by the time we left. With all the plots rendered as we left them.

Screen shot of one of the notebooks created in project is shown in Figure 4.1.

4.4 Web application

The application is a middle ground between tool built for development tools and final product for end-user. The goal was to create a simplified interface to execute the project code (perform segmentation and classification on the provided image). During the development of the segmentation model, I was lacking the tool for interactive access to the output data. For the classification part, to speed up segmentation process I had to create an ROI selector. All of those requirements were met in this web application. I have chosen to implement it as a web application due to **following reasons**:

- There is no need for requirements installation on the user machine. There is quite a few of them, and they might me platform specific.
- Canvas element is suitable for image manipulations that were required

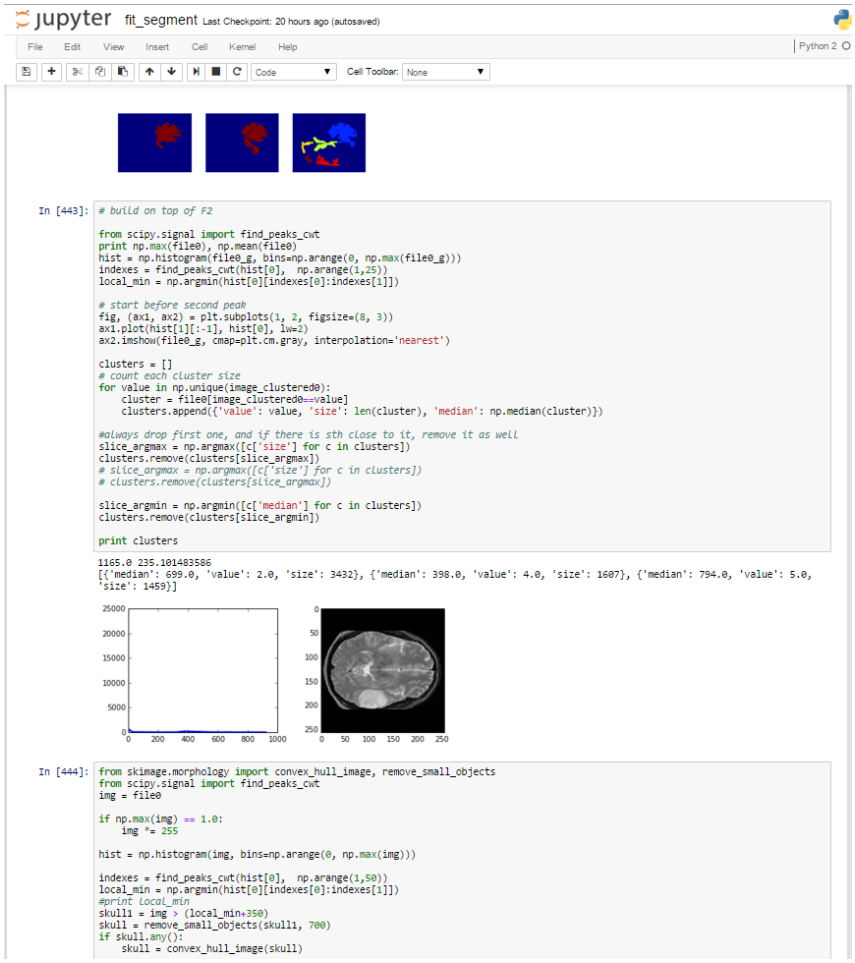

FIGURE 4.1: Screen shot of Jupyter environment

- It allows to use additional computational power, grater than average works station

The application (Figure 4.2) was written in Tornado Web Framework (server side) and pure JavaScript (front-end). It is server rich application, front-end is just a display tool. The application is an interface to features implemented in Python. The communication layer is implemented using WebSocket. It is a full-duplex TCP connection that supports binary transmission. It was an obvious choice due to heavy communication between client and server. The images are computed on the server side and then transferred through communication layer, to be displayed inside the canvas element.

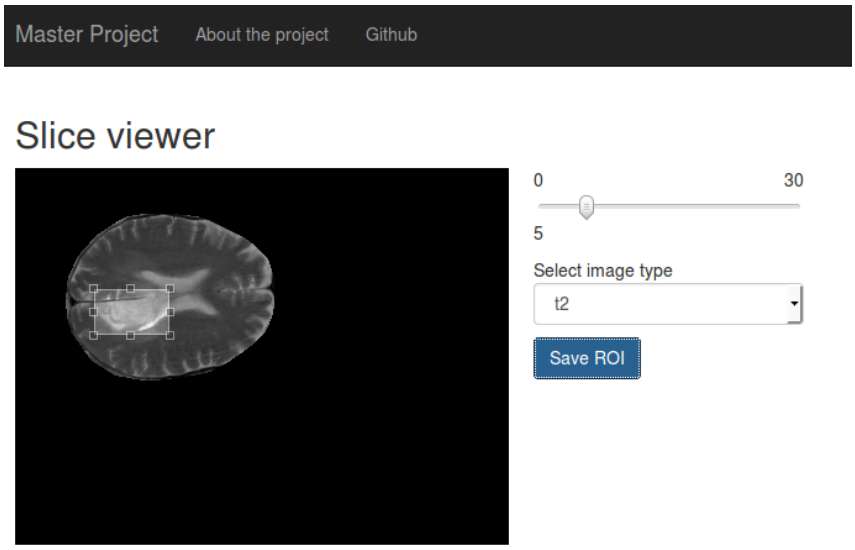

FIGURE 4.2: Application - slice viewer

Chapter 5

Results

5.1 Radiology imaging

Segmentation was carried out with two algorithms. Primary one was evaluated and tested against alternatives. K-Means had the best scores and computation efficiency. Secondary algorithm was based on the hemispheres symmetry analysis. Tumor segmentation algorithms were scored based to similarity measure between the scored tumor and corresponding truth table.

TABLE 5.1: Tumor segmentation (with all the samples)

METHOD	BEST SCORE
K-Means (5 clusters)	87.240% (std: 12.607)
K-Means with position (5 clusters)	86.141% (std: 10.351)
Mini Batch K-Means (5 clusters)	85.141% (std: 15.163)
Agglomerative Clustering	82.438% (std: 12.605)

TABLE 5.2: Tumor segmentation (without sample 5)

METHOD	BEST SCORE
Mini Batch K-Means (5 clusters)	89.027% (std: 5.408)
K-Means (5 clusters)	88.168% (std: 5.264)
K-Means with position (5 clusters)	86.026% (std: 5.282)
Agglomerative Clustering	88.956% (std: 10.632)

Segmentation was used to extract the tumor from the input images. For each sample (out of 27) attributes based on intensity, shape, position and textural features were extracted. Attributes were evaluated, irrelevant features (in terms of efficiency, impact on accuracy) were removed. Final set of attributes and corresponding classes was evaluated using Logistic Regression and Random Forest Classifier. The score of the evaluation is equal to the accuracy of classifier. Predictive performance of the model was measured in 5-fold cross validation.

TABLE 5.3: Cancer classification

METHOD	BEST SCORE
Random Forest Classifier (selected attributes)	87.0% (std: 12.991)
Logistic Regression (selected attributes)	81.297% (std: 5.744)
Logistic Regression (texture)	68.285% (std 0.082)

5.2 Combined Radiology and Pathology Classification

One of the goals of my thesis is to find out if combined pathology and radiology will impact the accuracy. I have limited options, because I do not have a direct access to the pathology model. I am limited to

the list of probabilities of Oligodendroglioma cancer for each sample (attached as Appendix B). There are two options to combine to combine the classifications. First one is to combine the probabilities of Oligodendroglioma (calculate by the pathology classifier) with the input vector of features extracted from MRI images. And then look for the correlation between features. The correlation with the MRI features was close to 0. Moreover, importance measured by the Random Forest in comparison to the informative features was negligible. Addition of the pathology data had negative impact on the final accuracy. The accuracy drop was in range of the standard deviation.

I also tried to build a new model, based on combined probabilities. I extracted the probabilities of Oligodendroglioma cancer for each sample from the radiology-based model. The data set consisted of two features - probabilities of Oligodendroglioma cancer estimated by different models. I had almost the same results (within the range of standard deviation). Attribute evaluation using Naive Bayes showed that final model was 80% based on the radiology attribute.

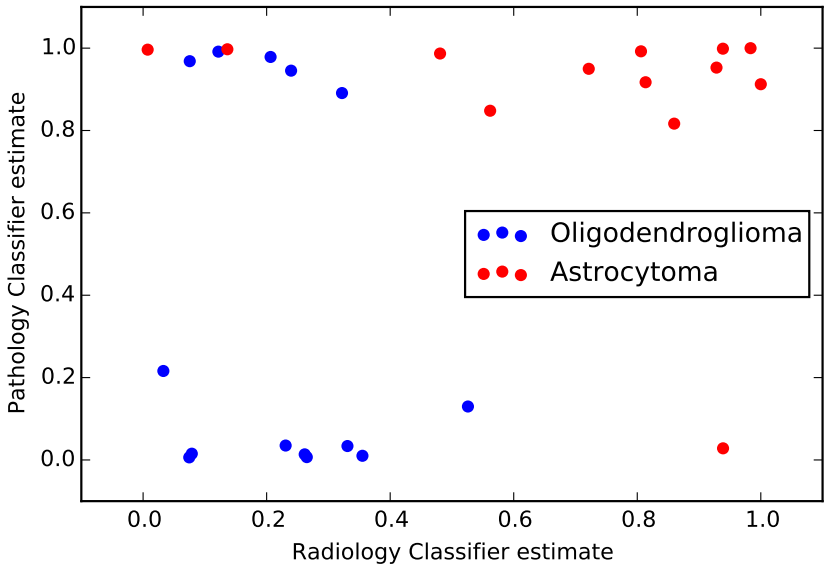

FIGURE 5.1: Comparison of Pathology and Radiology results (average estimations of Oligodendroglioma cancer for each sample)

Figure 5.1 shows distribution of Oligodendroglioma estimations in Pathology and Radiology imaging. Even though it looks like, the data separation is straightforward, those are specific cases. Based on the data I had, I was unable to generalize it, to such extend, that would result in the improved accuracy. The radiology model alone had the best results out of three (Radiology alone, combined Radiology and Pathology, Pathology alone).

5.3 Conclusions

Pre-processing of the input data is a hand-crafted process, that had to be performed. Radiology images of the same patient varied in terms of shape, average intensity level and head position. Both Otsu and adaptive method based on the histogram peaks (see Section 2.3) were able to extract binary mask of the head.

I was able to adopt the image segmentation techniques to execute on set of radiology images. K-Means had the best score (similarity measure to the grand truth, calculated with adjusted rand score) out of Mini Batch K-Means, K-Means with modified input vector (with position), and Agglomerative clustering. K-Means had the bast score with 5 clusters, based on the Silhouette analysis and supervised Grid Search. Differences in the clustering techniques were subtle. Agglomerative clustering was the cleanest (see Figure 2.23), it did not require to split the clusters into smaller areas, therefore it was the easiest to implement. Mini Batch K-Means proven to be almost as good as K-Means, with twice as fast execution time.

The challenging part was to select the proper cluster, that corresponds to the tumor area. I was unable to do that using only single (primary) segmentation algorithm. Symmetry analysis of the hemispheres proven to be versatile tool for the cluster classification (whether it is a tumor or not). In the areas that was not affected by tumor, the asymmetrical differences were subtle.

Intensity, shape, position and textural features were extracted from the T1, T1C, FLAIR and T2 scans (see Section 2.7.1). Based on evolution and selection of features tumor position and distribution of intensity in FLAIR image was most relevant. According to the literature Oligodendroglioma often occurs in frontal and temporal lobes [Ass16c], it proven to be truth in my data set. If I would have only the position of the tumor, cross-validation classifier would have an accuracy of 70% (std: 12). Distribution of intensity in FLAIR scan had high correlation and the highest importance. In the T2 scans lesions always had high intensity, however in FLAIR lesion were either partially bright or dark. Textural pattern based classification requires more samples to be effective.

Random Forest classifier validated with k-fold cross validation had average accuracy of 87.0% (std: 12.991). In my data set, model build on the features (FLAIR distribution in 4th and 5th histogram bin and tumor position) exceeded results of pathology classifier (that had 73% of accuracy, see Section 3.4).

My primary goal was to build a radiology classifier, that will have an accuracy in rage of 60-70%. Then, using additional data from the pathology classification, improve the accuracy. Radiology-based classification result exceeded my expectations. I was unable to improve the accuracy of the final radiology-based model with the pathology-based data.

5.4 Further Work

- The data set I am using is too small to speculate about the universal model. Due of nature of the data samples, that possess required medical data, are extremely hard to find. Some kind of cooperation with medical institution could result in data augmentation
- K-Means with custom vector (described in Section 2.6.3) requires further work. The approach seems promising based on its performance and achieved score. Euclidean distance might be problematic for straight coordinates. It could be resolved by modifying their coordinate system or by using different distance metric
- A supervised technique for features extraction could be implemented. I only manage to extract finite number of feature, that I came across in literature. The automatic method could extract new features. That potentially could increase classification accuracy
- With more samples that possess the truth tables convolutional neural network could be implemented for the segmentation part. It is a recent trend, that lead to significant advancements in segmentation.

Listings

List of Tables

Appendix A

Original data set

TABLE A.1: List of samples (1-20)

PATIENT No.	MRI SCANS	DIAGNOSIS
1	T1, T1C	A
2	FLAIR, T2, T1, T1C	A
3	FLAIR, T2, T1, T1C	O
4	FLAIR, T2, T1, T1C	O
5	FLAIR, T2, T1, T1C	O
6	FLAIR, T2, T1, T1C	O
7	FLAIR, T2, T1, T1C	A
8	FLAIR, T2, T1, T1C	A
9	T2, T1, T1C	A
10	FLAIR, T2, T1, T1C	O
11	FLAIR, T2, T1, T1C	A
12	FLAIR, T2, T1, T1C	O
13	FLAIR, T2, T1	O
14	FLAIR, T2, T1	A
15	FLAIR, T2, T1, T1C	A
16	FLAIR, T2, T1, T1C	O
17	FLAIR, T2, T1, T1C	A
18	T2, T1	A
19	FLAIR, T2, T1, T1C	A
20	FLAIR, T2, T1C	O

TABLE A.2: List of samples (21-32)

PATIENT No.	MRI SCANS	DIAGNOSIS
21	FLAIR, T2, T1, T1C	O
22	FLAIR, T2, T1, T1C	O
23	FLAIR, T2	A
24	FLAIR, T1, T1C	O
25	FLAIR, T2, T1, T1C	O
26	T2, T1	A
27	FLAIR, T2, T1, T1C	O
28	FLAIR, T2, T1, T1C	A
29	FLAIR, T1, T1C	A
30	-	O
31	FLAIR, T2, T1, T1C	A
32	FLAIR, T1, T1C	O

Appendix B

Pathology results

This is the data I got from the Pathology classification. It's a table of probability estimates of Oligodendroglioma cancer. Average accuracy of this model reached a 73%.

TABLE B.1: List of samples (1-20)

PATIENT No.	PROBABILITY OF OLIGODENDROGLIOMA
1	0.00
2	0.22
3	1.00
4	1.00
5	0.99
6	1.00
7	0.02
8	0.89
9	0.01
10	0.85
11	0.04
12	0.95
13	0.03
14	0.01
15	0.97
16	1.00
17	0.01
18	0.03
19	0.13
20	0.99

TABLE B.2: List of samples (21-32)

PATIENT No.	PROBABILITY OF OLIGODENDROGLIOMA
21	0.92
22	0.91
23	0.98
24	1.00
25	0.82
26	0.95
27	0.95
28	0.01
29	1.00
30	0.99
31	0.99
32	0.88

Bibliography

[AMEAA15] Eman Abdel-Maksouda, Mohammed Elmogyb, and Rashid Al-Awadic. "Brain tumor segmentation based on a hybrid clustering technique, Computed Tomography, Ultrasonography, Magnetic Resonance Imaging, and Scintigraphy". In: (2015).

[Ass16a] The American Brain Tumor Association. *Astrocytoma*. 2016. URL: http://www.abta.org/brain-tumor-information/types-of-tumors/astrocytoma.html (visited on 01/01/2016).

[Ass16b] The American Brain Tumor Association. *Malignant and Benign Brain Tumors*. 2016. URL: http://www.abta.org/brain-tumor-information/diagnosis/malignant-benign-brain-tumors.html (visited on 01/01/2016).

[Ass16c] The American Brain Tumor Association. *Oligodendroglioma*. 2016. URL: http://www.abta.org/brain-tumor-information/types-of-tumors/oligodendroglioma.html (visited on 01/01/2016).

[Ass16d] The American Brain Tumor Association. *Types of Tumors*. 2016. URL: http://www.abta.org/brain-tumor-information/types-of-tumors/ (visited on 01/01/2016).

[CMK10] S. Allin Christe, K. Malathy, and A. Kandaswamy. "IMPROVED HYBRID SEGMENTATION OF BRAIN MRI TISSUE AND TUMOR USING STATISTICAL FEATURES". In: (2010).

[Cza+15] Jakub Czakon et al. "An ensemble algorithm for the nuclei segmentation in the histological images". In: (2015).

[ECM15] MD Edward Claro Mader. *What is the difference between T1 and T2 imaging in MRI*. 2015. URL: https://www.quora.com/What-is-the-difference-between-T1-and-T2-imaging-in-MRI (visited on 01/13/2015).

[Fes+13] Joana Festa et al. "Automatic Brain Tumor Segmentation of Multi-sequence MR images using Random Decision Forests". In: *Proceedings of NCI-MICCAI BRATS 2013*. Nagoya, Japan, 2013.

[Han09] Lars G. Hanson. "Introduction to Magnetic Resonance Imaging Techniques". In: (2009). URL: http://eprints.drcmr.dk/37/1/MRI_English_a4.pdf.

[HHS04] U.S. Department of Health & Human Services. *NIfTI-1 Data Format*. 2004. URL: http://nifti.nimh.nih.gov/nifti-1 (visited on 01/01/2016).

[JRH16] FACR John R. Hesselink MD. *BASIC PRINCIPLES OF MR IMAGING*. 2016. URL: http://spinwarp.ucsd.edu/neuroweb/Text/br-100.htm (visited on 01/01/2016).

[KCB07] Hassan Khotanlou, Olivier Colliot, and Isabelle Bloch. "Automatic brain tumor segmentation using symmetry analysis and deformable models". In: (2007).

[M.D16] Benjamin Kennedy M.D. *Astrocytoma*. 2016. URL: http://emedicine.medscape.com/article/283453-overview (visited on 01/01/2016).

[Moh+13] Punithavathy Mohan et al. "Intelligent Based Brain Tumor Detection Using ACO". In: (2013).

[NMB13] Proceedings of NCI-MICCAI BRATS 2013. "Multimodal Brain Tumor Segmentation". In: (2013).

[PEC12] Eanes Torres Pereira, Sidney Pimentel Eleuterio, and Joao Marques de Carvalho. "Local Binary Patterns Applied to Breast Cancer Classification in Mammographies". In: (2012).

[PEP09] Carlos Pineda, Rolando Espinosa, and Angelica Pena. "Radiographic Imaging in Osteomyelitis: The Role of Plain Radiography, Computed Tomography, Ultrasonography, Magnetic Resonance Imaging, and Scintigraphy". In: (2009). URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2884903/f.

[RD12] V.P.Gladis Pushpa Rathi and Dr.S.Palani. "BRAIN TUMOR MRI IMAGE CLASSIFICATION WITH FEATURE SELECTION AND EXTRACTION USING LINEAR DISCRIMINANT ANALYSIS". In: (2012).

[Tus+10] N. J. Tustison et al. "N4ITK: Improved N3 Bias Correction". In: (2010).

[ZP12] Nitish Zulpe and Vrushsen Pawar. "GLCM Textural Features for Brain Tumor Classification". In: (2012).

[Żu+15] Grzegorz Żurek et al. "Brain tumor classification using logistic regression and linear support vector classifier". In: Computational Brain Tumor Cluster of Events (CBTC), At Munich, Germany, 2015.