Data Science Lab in Bioscience

An Integrated Approach for Data Analysis of Drawings and Articles Related to Parkinson's Patients

Anna Maria Teodori (889903), Ardalan Mehraram(856228), Michele Gazzola (825267)

Agenda

- The Object of Analysis
- Project Goals
- Literature Analysis
- Text Mining
- Case Study

The Object of Analysis

“Parkinson disease is a neurodegenerative disorder that mostly presents in later life with generalized slowing of movements (bradykinesia) and at least one other symptom of resting tremor or rigidity.”

(Parkinson Disease)

“Spiral drawing is commonly used to visually rate tremor intensity”

(Drawing Analysis)

“Tremor severity is estimated through visual rating of the drawings by movement disorders experts”

(Action Tremor Quantification)

Project Goals

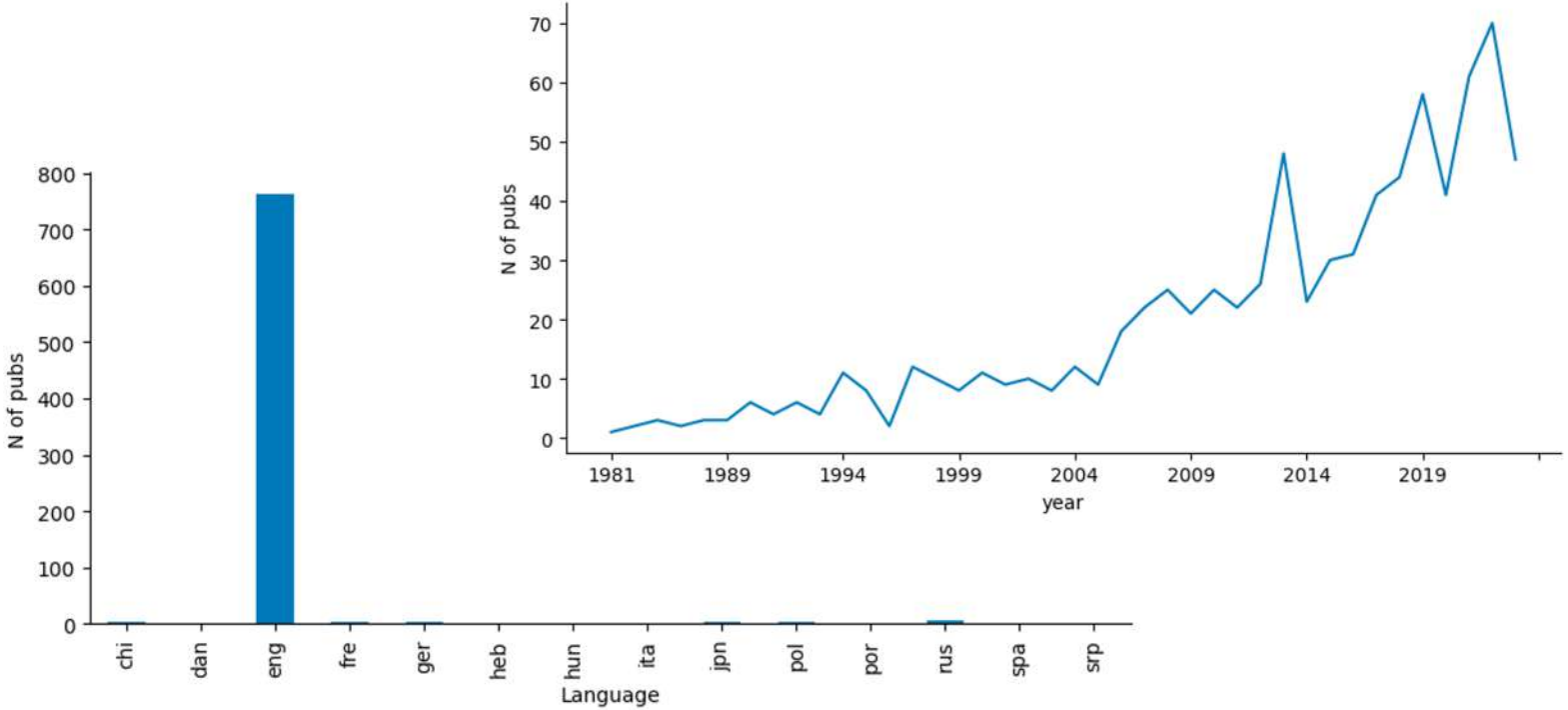
- Applying text mining analysis for evaluate the literature around the use of patient drawings as a metric of disease severity

- Verify how the use of Machine Learning Models can help for this purpose

Literature Analysis

N. of articles: 797

Source: PubMed



Text Mining

Data Acquisition

Topic Modeling

Preprocessing

Text Mining

Data Acquisition

```
from Bio import Entrez
```

```
pubmed_record = Entrez.efetch(db="pubmed", id=pubmed_id, retmode="xml")
```

PubmedID	ArticleTitle	PubDate	Abstract	Language	date
37695259	Visuospatial memory profile of patients with P...	2023-09-11	In Parkinson's Disease (PD) cognitive impairme...	eng	2023 Sep 11
37626522	Lipid Metabolism Disorder in Cerebrospinal Flu...	2023-08-04	Abnormal accumulation of lipids is found in do...	eng	2023 Aug 4
37625589	Endogenous Rab38 regulates LRRK2's membrane re...	2023-08-23	Point mutations in leucine-rich repeat kinase ...	eng	2023 Aug 23
37603024	An introduction to directed acyclic graphs in ...	1970-01-01	Directed acyclic graphs (DAGs) are visual repr...	eng	2023 Sep
37553934	Effects of Exercise-based Management on Motor ...	1970-01-01	Parkinson's disease (PD) is the second most co...	eng	2023 Aug
...	...	...	...	...	...
6348585	Comparison of pergolide and bromocriptine ther...	1970-01-01	Twenty-four parkinsonian patients compared per...	eng	1983 Aug

Text Mining

Preprocessing

1. Drop Nulls
2. Remove Numbers
3. Remove punctuation
4. Remove Stopwords
5. Tokenization
6. Bigrams
7. Lemmatization

```
df["preprocessed_text"].head(4)
```

```
0    [parkinson, disease, disease, cognitive, cogni...
1    [abnormal, accumulation, accumulation, lipid, ...
2    [point, mutation, mutation, leucine, leucine, ...
3    [directed, acyclic, acyclic, graph, graph, dag...
Name: preprocessed_text, dtype: object
```

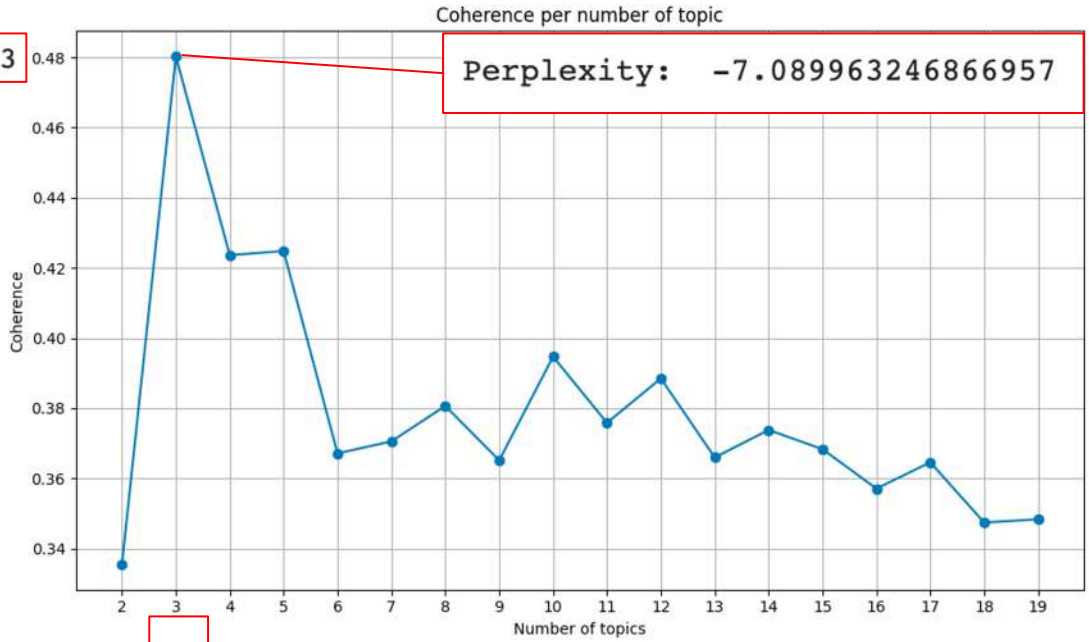
Text Mining

Topic Modeling

```
!pip install pyLDAvis
```

0.4802522782555423

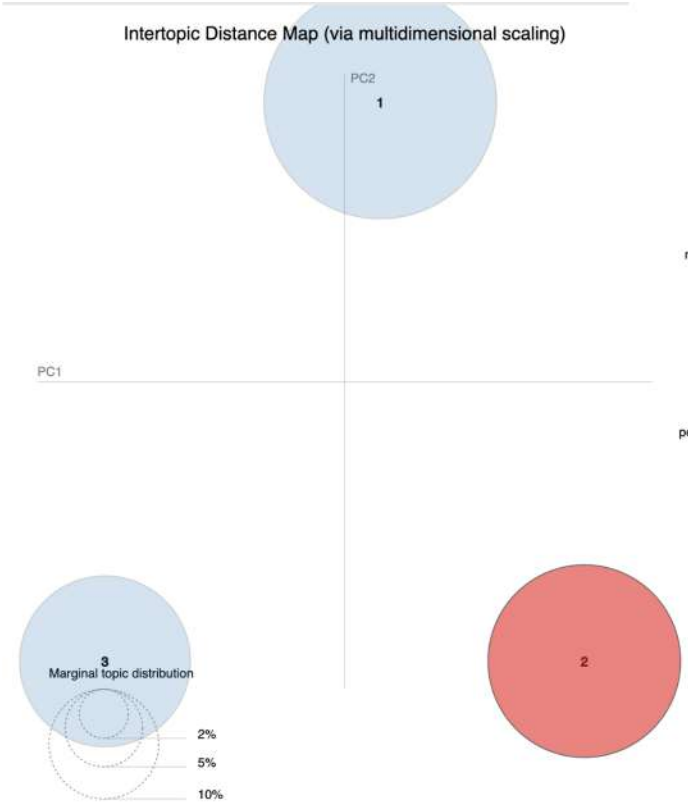
Perplexity: -7.089963246866957

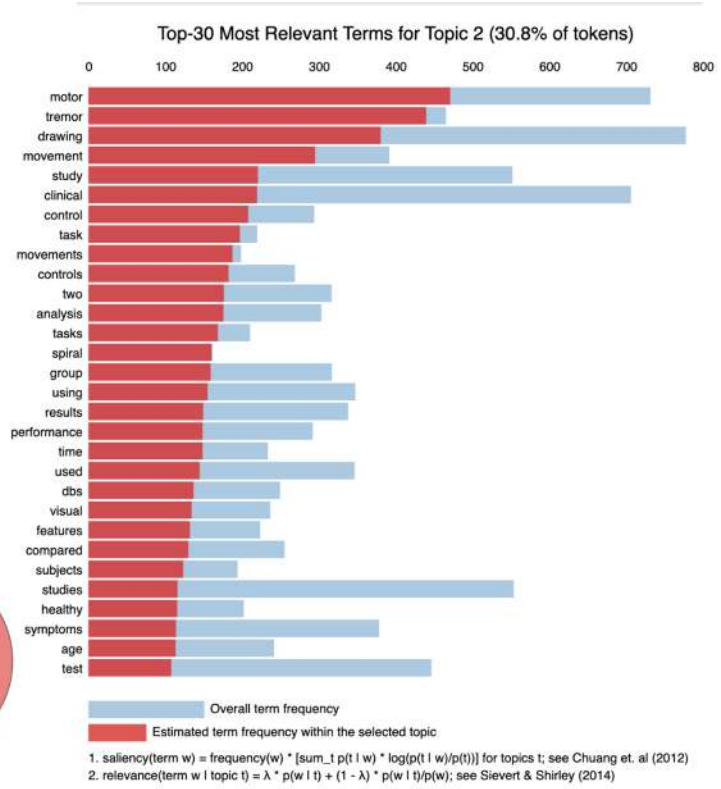


3

Text Mining

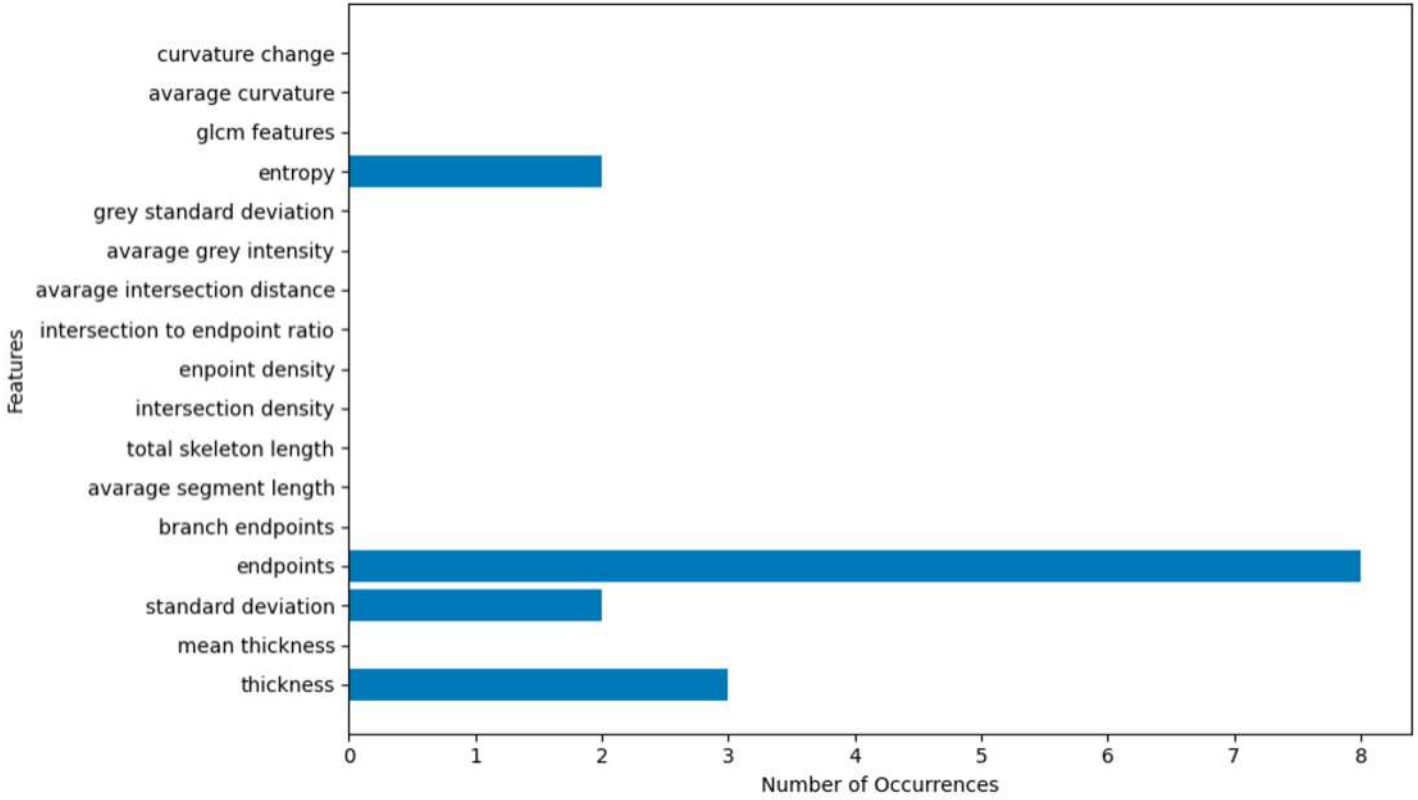
Topic Modeling





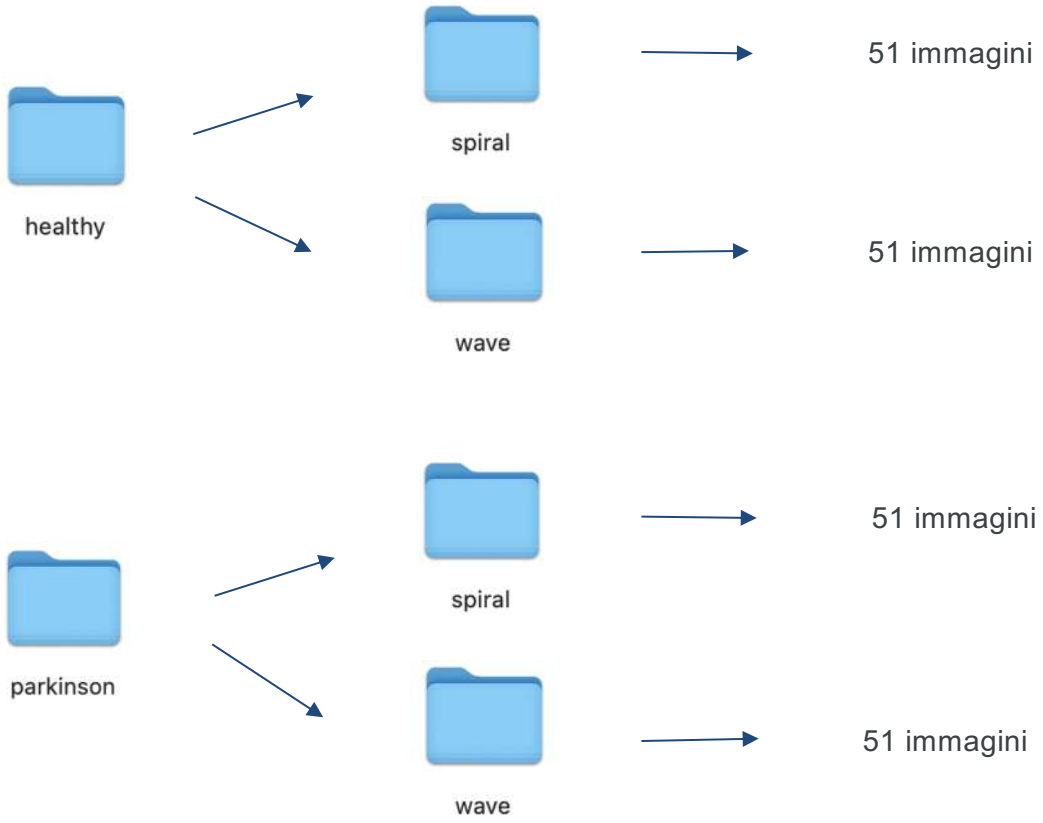
Text Mining

Searching for Features



Case Study

Dataset


healthy
spiral
wave
parkinson
spiral
wave
51 immagini
51 immagini
51 immagini
51 immagini

Data Exploration

Image Shape: 256x256

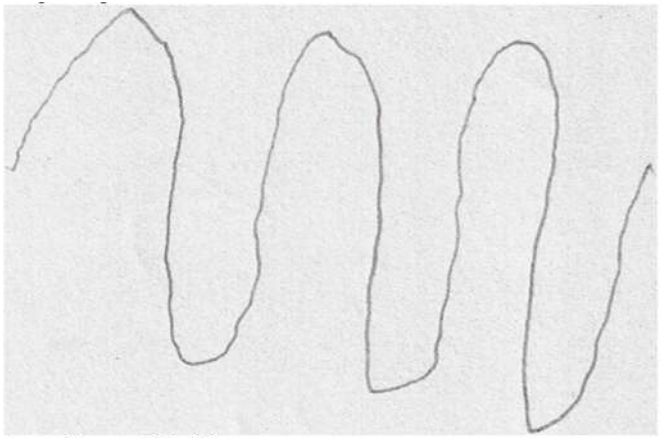


Image Shape: 512x393

Image Shape: 256x256

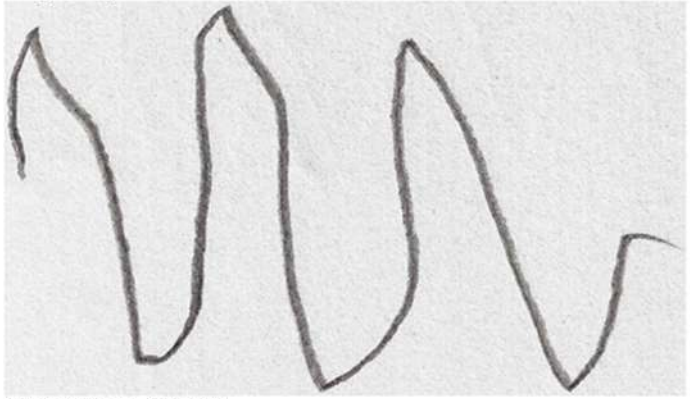


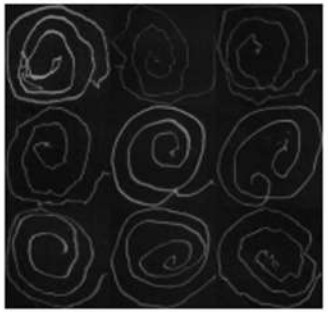
Image Shape: 512x297

Data Preprocessing

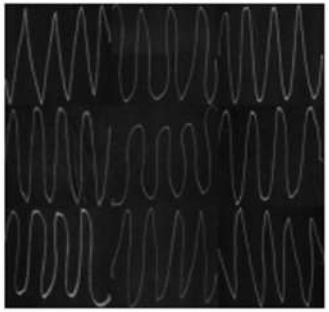
Reading and Resizing

spiral healthy

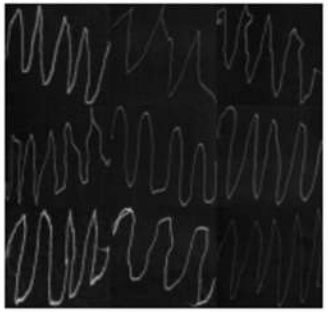
spiral parkinson



wave healthy



wave parkinson

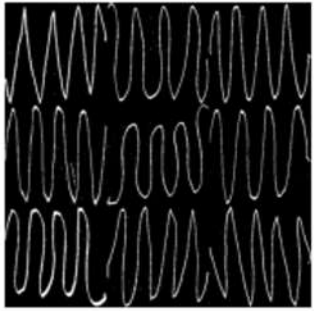


Generates Binary Image Based on Thresholding

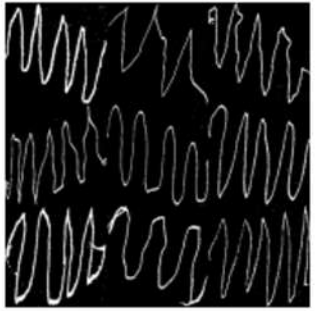
spiral healthy

spiral parkinson

wave healthy

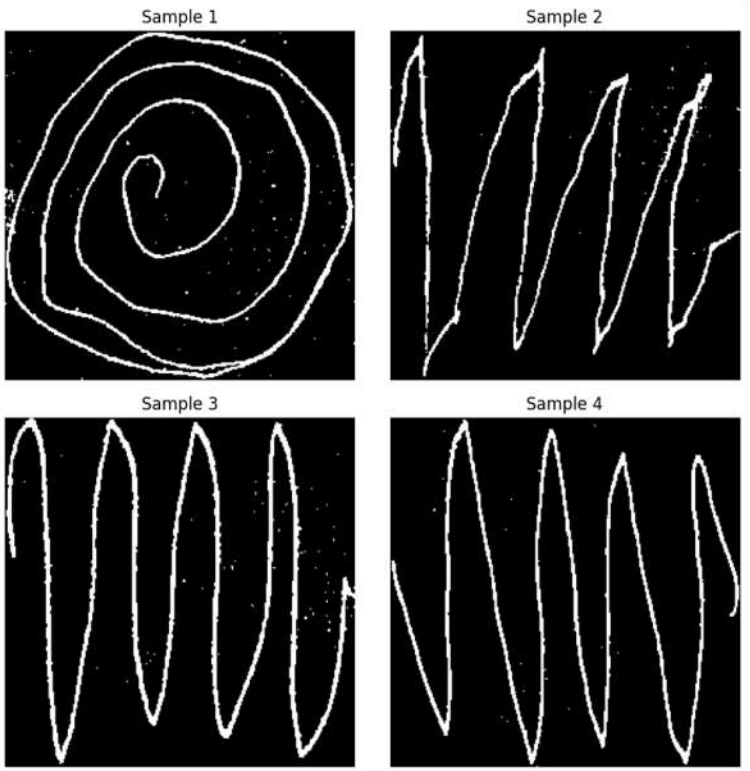


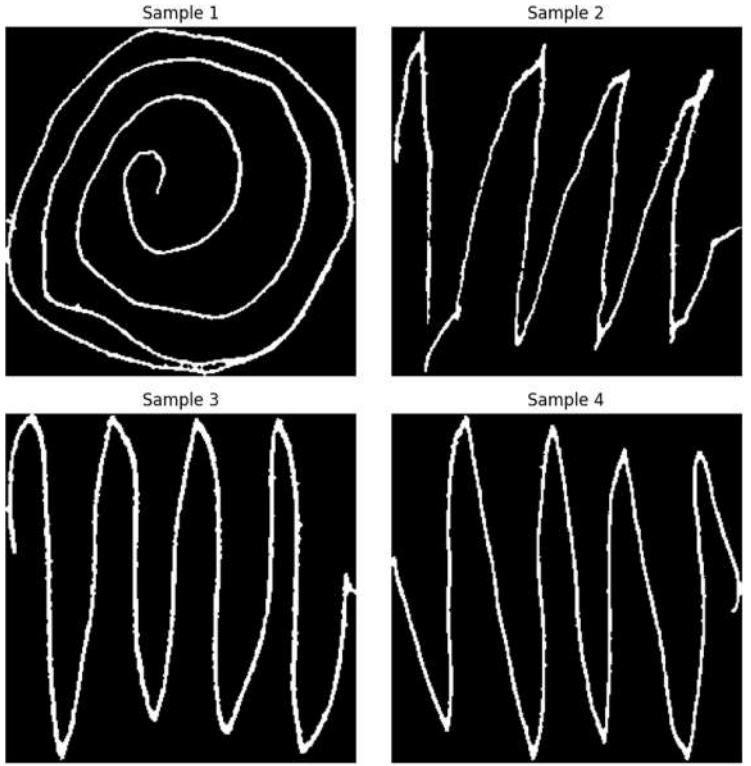
wave parkinson



Data Preprocessing

Image Labeling and Object Filtering for Improved Image Analysis



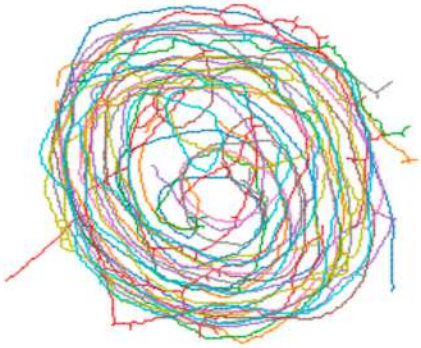


Data Visualization

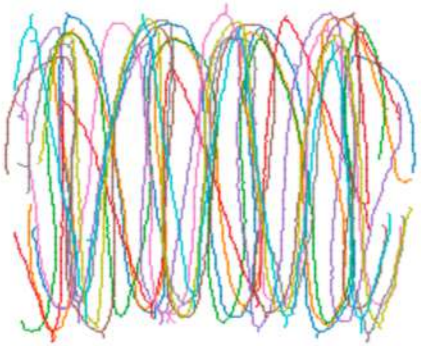
Visualization of Skeletonized Image Data and Analysis of Image Differences

spiral healthy

spiral parkinson



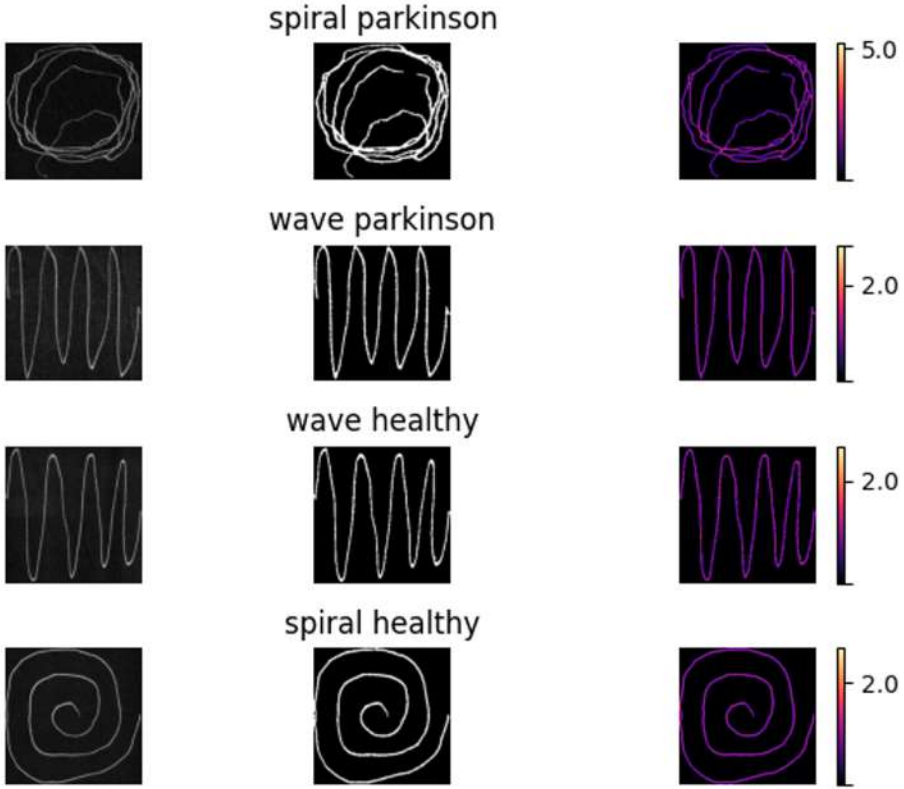
wave healthy



wave parkinson

Feature Extraction

Stroke Thickness Analysis and Calculation of Mean and Standard Deviation



Range of Standard Deviation of Stroke Thickness

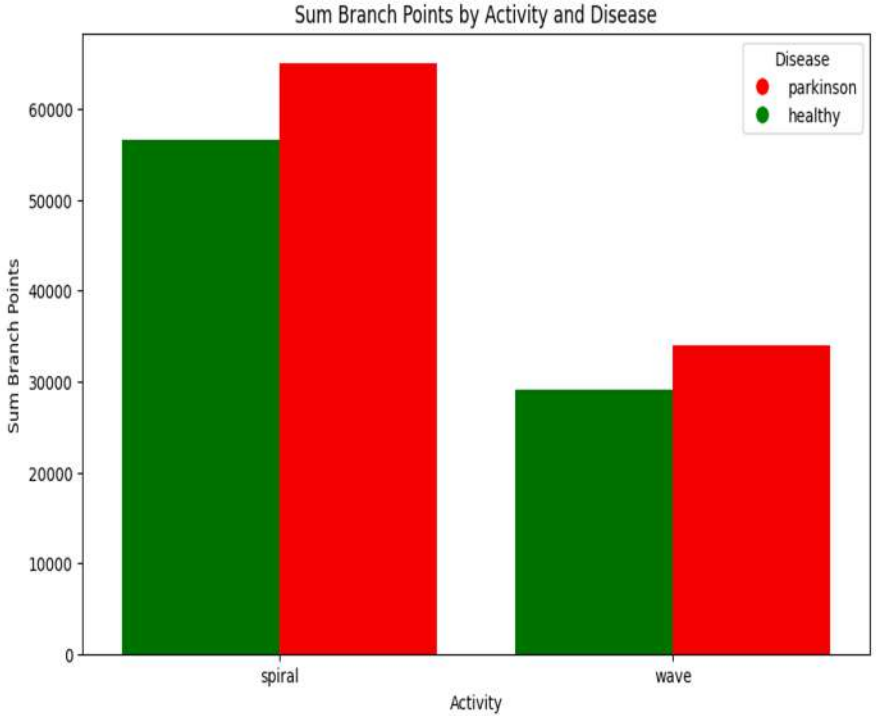
activity	disease	
spiral	healthy	0.451398
	parkinson	1.682739
wave	healthy	0.380710
	parkinson	0.753593

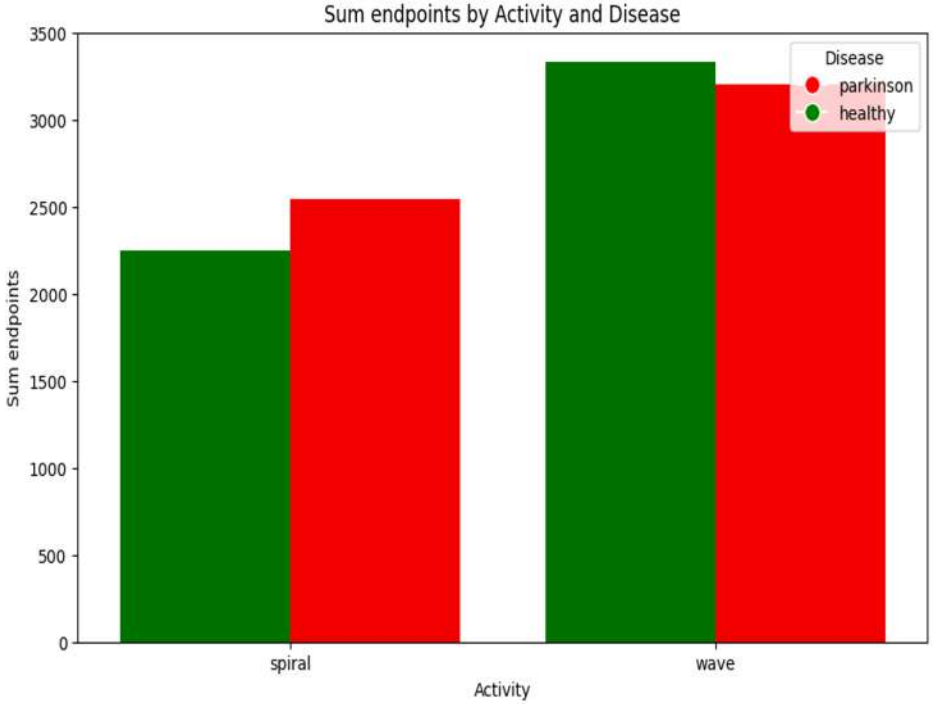
Range of Mean of Stroke Thickness

activity	disease	
spiral	healthy	1.741688
	parkinson	2.662872
wave	healthy	1.800135
	parkinson	1.738006

Feature Extraction

Identification of Endpoints and Branch Points in Image Data





Feature Extraction

Average Segment Length and Total Skeleton Path Length Intersection and Point Density and Endpoint Density

activity	disease	Median Avg Segment Length	Total Skeleton Length
spiral	healthy	2.506122	71432
spiral	parkinson	2.548837	81769
wave	healthy	6.143836	86318
wave	parkinson	5.059524	80723

activity	disease	Mean Intersection Density	Mean Endpoint Density
spiral	healthy	0.581415	0.424406
spiral	parkinson	0.588297	0.432951
wave	healthy	0.855157	0.638555
wave	parkinson	0.739997	0.571322

Feature Extraction

Intersection-to-Endpoint Ratio and Mean Gray Levels and Standard Deviation of Gray Levels and Entropy

activity	disease	Range Intersection-to-Endpoint Ratio
spiral	healthy	0.277254
spiral	parkinson	0.282005
wave	healthy	0.100876
wave	parkinson	0.208010

activity	disease	Range Avg Gray Intensity	Range Gray Std Dev
spiral	healthy	0.087540	0.128834
spiral	parkinson	0.166809	0.192642
wave	healthy	0.106415	0.138724
wave	parkinson	0.143921	0.192057

activity	disease	Mean Entropy
spiral	healthy	0.411622
spiral	parkinson	0.446789
wave	healthy	0.410521
wave	parkinson	0.434582

Feature Extraction

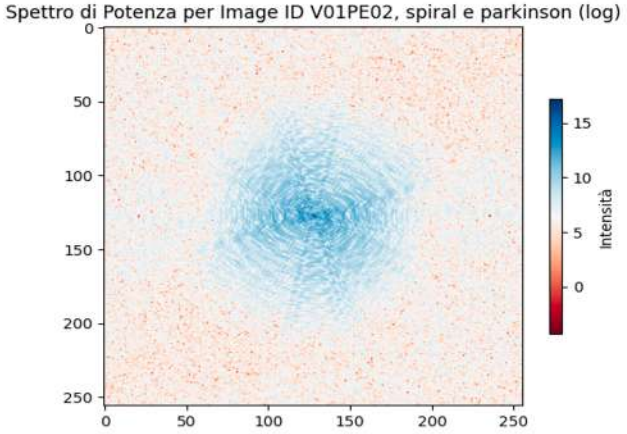
Texture Analysis using GLCM (Gray-Level Co-Occurrence Matrix), Average Curvature and Curvature Change

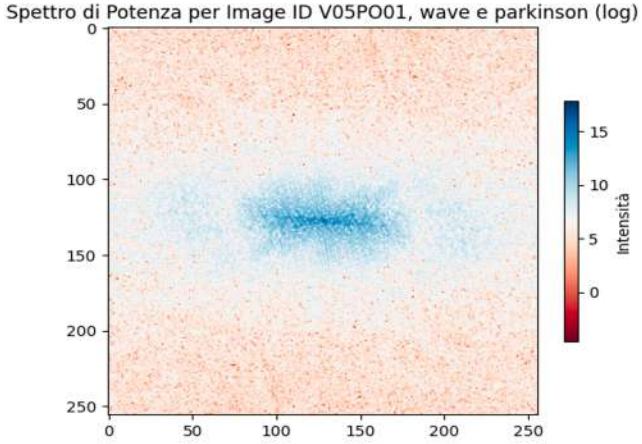
activity	disease	Range Contrast	Range Dissimilarity	Range Homogeneity	Range Energy	Range Correlation
spiral	healthy	0.013067	0.016086	0.013480	0.014195	0.013067
spiral	parkinson	0.034421	0.043183	0.032782	0.032880	0.034421
wave	healthy	0.021752	0.021807	0.004779	0.022391	0.021752
wave	parkinson	0.038664	0.040754	0.009712	0.036463	0.038664

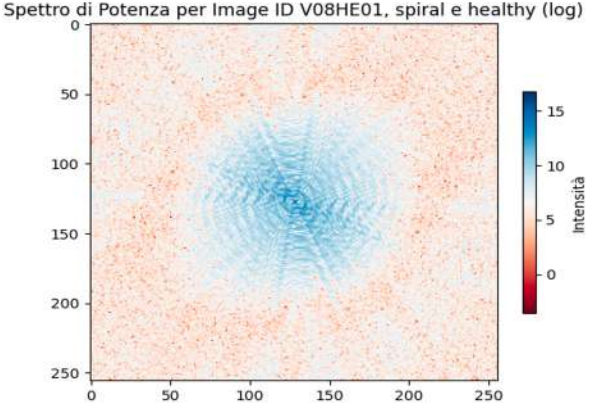
activity	disease	Range Curvature Change
spiral	healthy	1.533631e-17
spiral	parkinson	1.740346e-17
wave	healthy	6.749985e-18
wave	parkinson	1.171734e-17

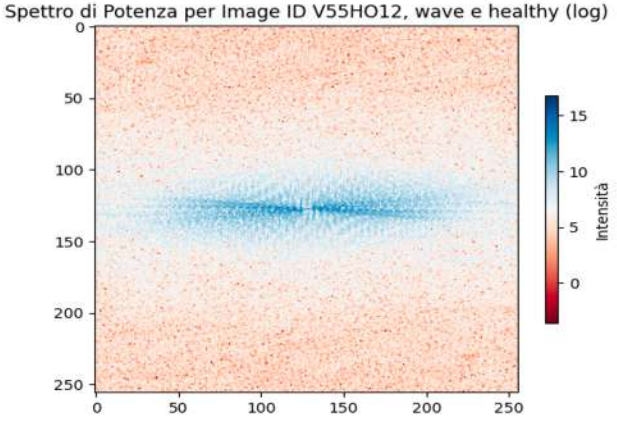
Feature Extraction

Power Spectrum









Model Development

Model Development

path	object
img_id	object
disease	object
activity	object
thresh_img	object
clean_img	object
x	float64
y	float64
thickness	object
mean_thickness	float64
std_thickness	float64
endpoints	int64
branch_points	int64
avg_segment_length	float64
total_skeleton_length	int64
intersection_density	float64
endpoint_density	float64
intersection_to_endpoint_ratio	float64
avg_gray_intensity	float64
gray_std_dev	float64
entropy	float64
contrast	float64
dissimilarity	float64
homogeneity	float64
energy	float64
correlation	float64
average_curvature	float64
curvature_change	float64
power_spectrum	object

```python
# Definisci una funzione per calcolare la media degli elementi in ciascun array
def average_power_spectrum(x):
    return np.mean(np.fromstring(x.strip('[]'), sep=' '))

# Applica la funzione di calcolo della media e assegna i risultati a una nuova colonna
df['average_power_spectrum'] = df['power_spectrum'].apply(average_power_spectrum)
```

```
df['average_power_spectrum']

0      525.281678
1      502.826132
2      378.742162
3      241.684848
4       53.748812
          ...
197     47.648581
198    118.387923
199    167.894973
200    248.670443
201     48.750880
Name: average_power_spectrum, Length: 202, dtype: float64
```

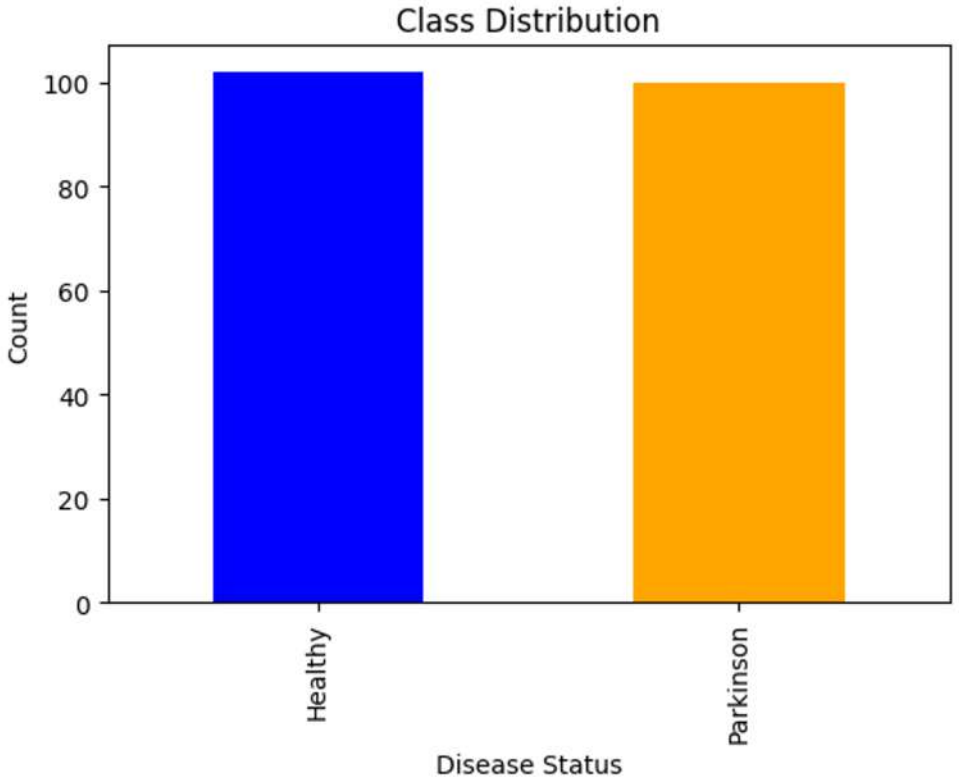
Model Development

disease to_binary

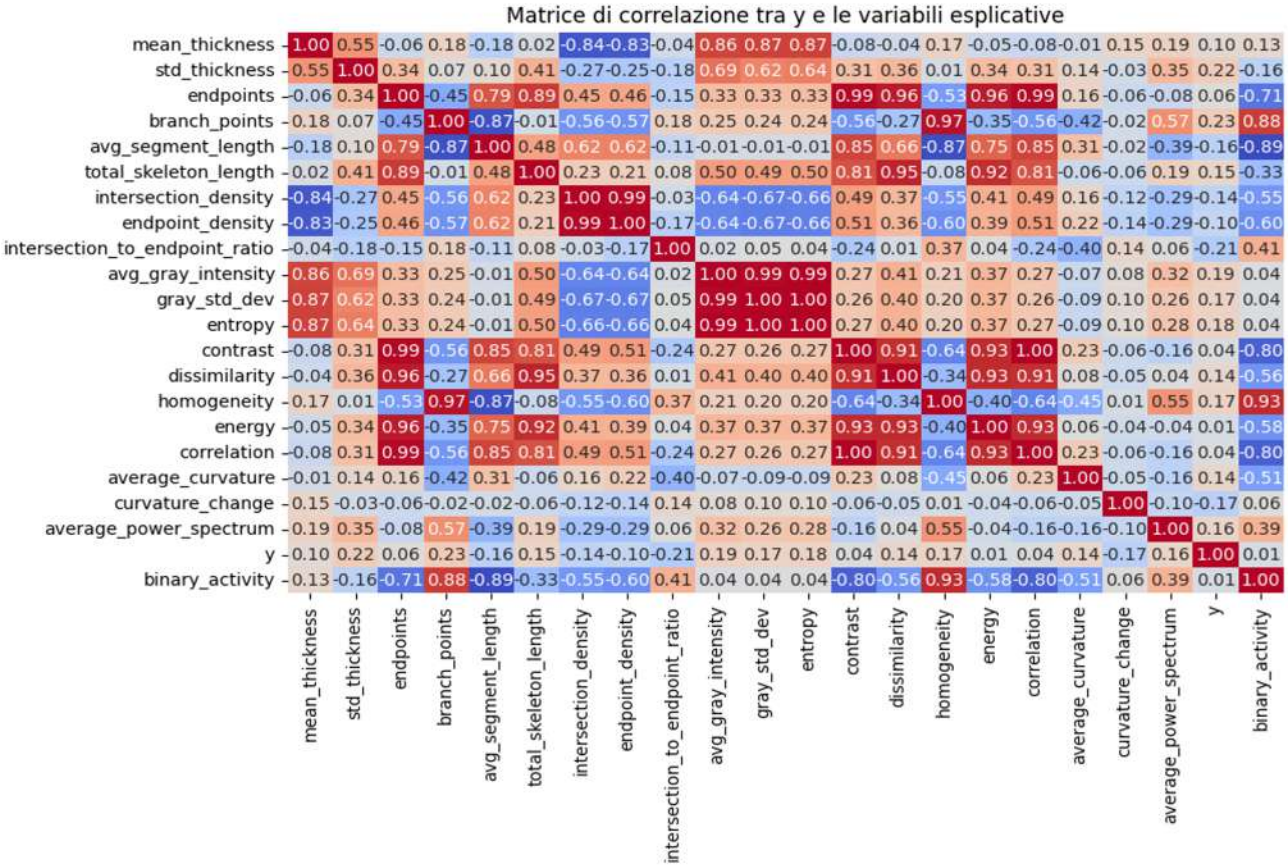
```python
# Define a function to convert disease to binary
def convert_to_binary(disease):
    if disease == 'parkinson':
        return 1
    else:
        return 0

# Apply the function to create the "disease_binary" column
df['y'] = df['disease'].apply(convert_to_binary)
y = df['y']
```

activity to_binary

```python
# Define a function to convert activity to binary
def convert_to_binary(activity):
    if activity == 'spiral':
        return 1
    else:
        return 0

# Apply the function to create the "disease_binary" column
df['binary_activity'] = df['activity'].apply(convert_to_binary)
binary_activity = df['binary_activity']
```

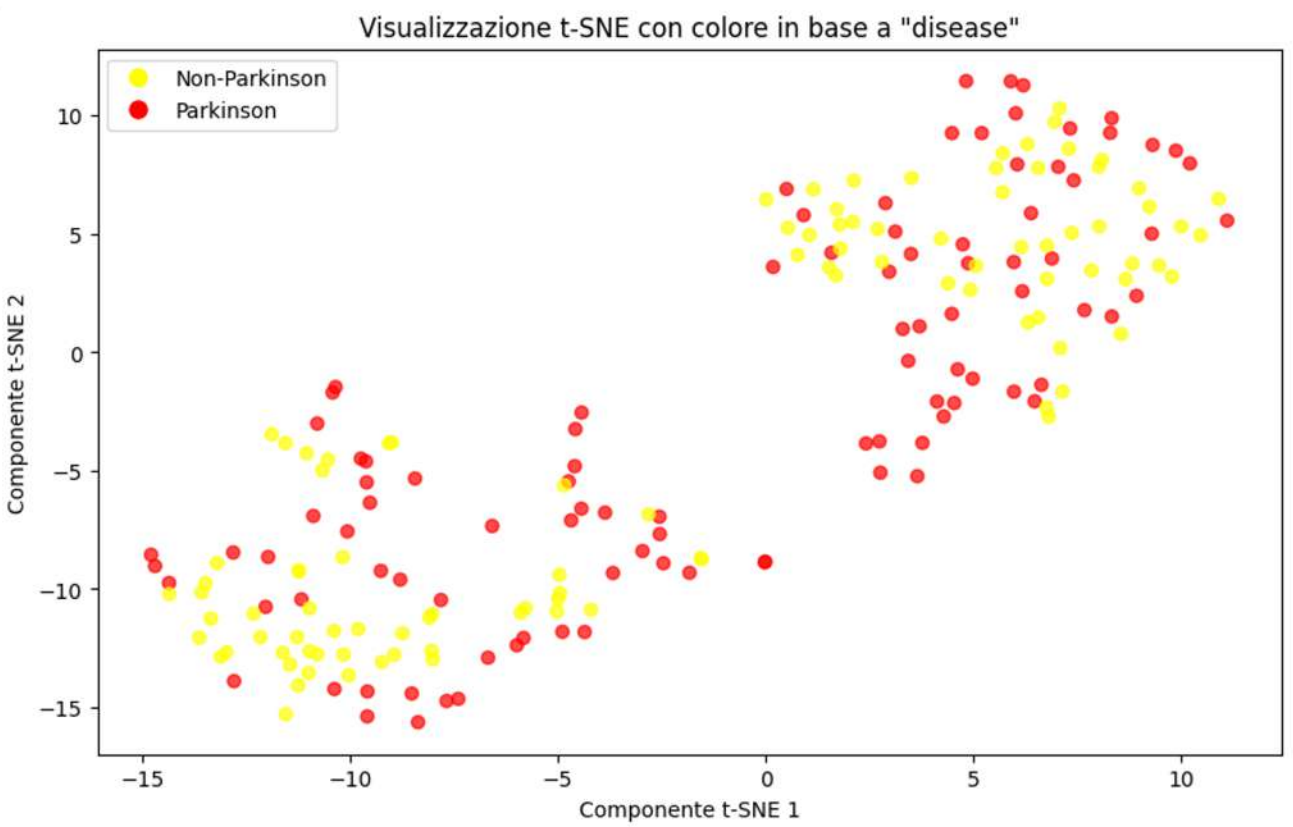


Model Development



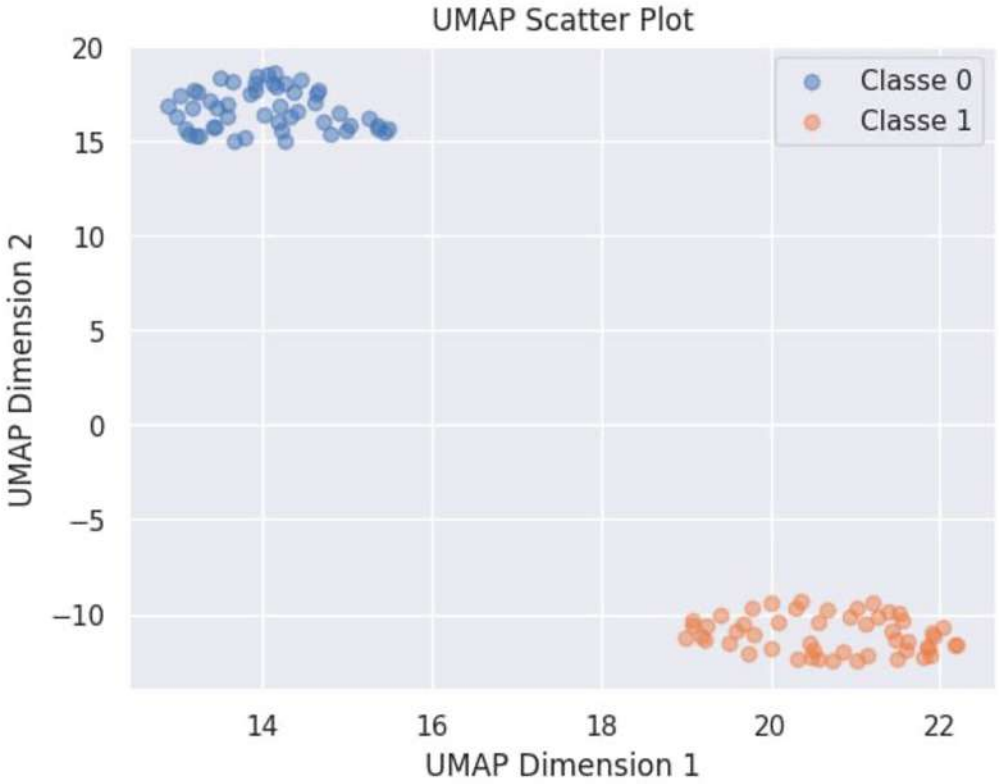
Model Development

complete dataset



Model Development

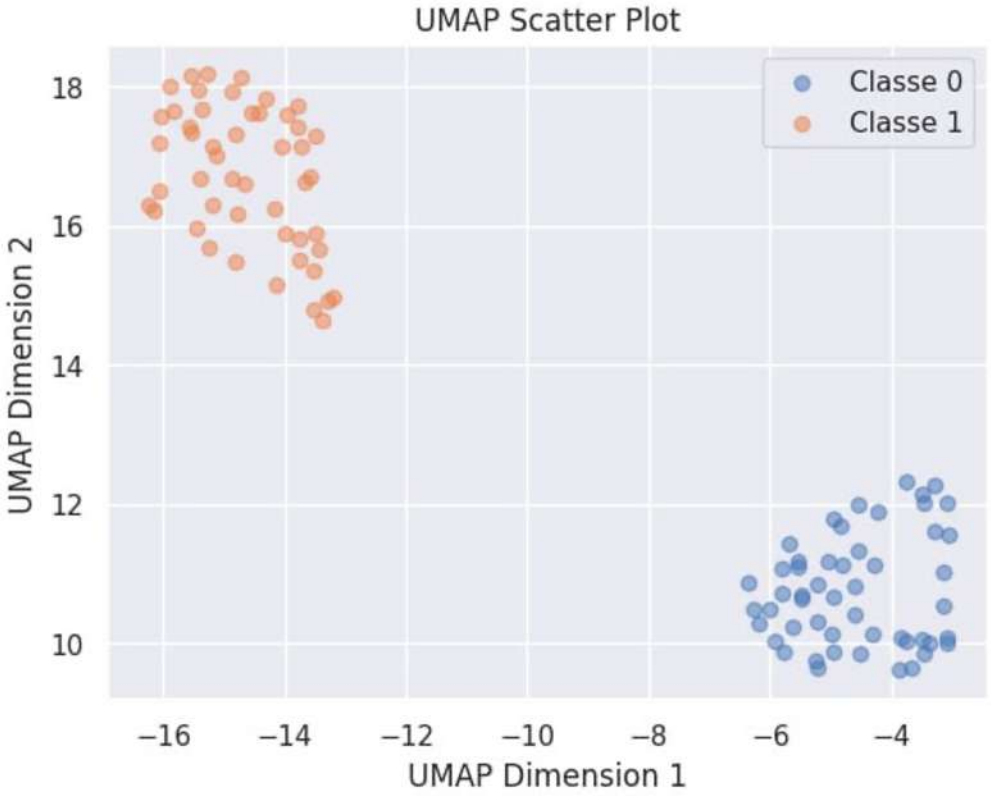
filtered by activity
spiral



Model Development

filtered by activity

wave



Model development

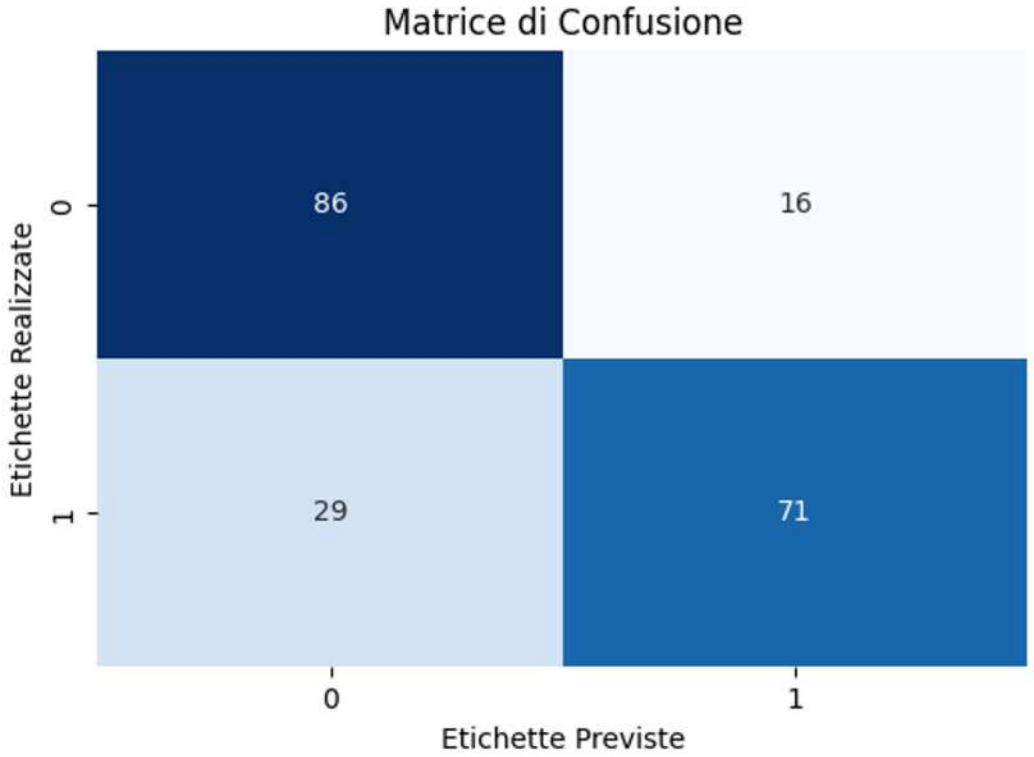
Logistic Regression

```
# Prepare the cross-validation procedure
cv = RepeatedKFold(n_splits=10, n_repeats=5, random_state=1)

# Create the logistic regression model
model_logistic = LogisticRegression()

# Define a grid of C values to search
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10]}

# Create GridSearchCV to search for the best C value
grid_search = GridSearchCV(estimator=model_logistic, param_grid=param_grid, scoring='accuracy', cv=cv, n_jobs=-1)

# Fit the model to the data using GridSearchCV
grid_search.fit(X, y)


# Print the best C value and corresponding accuracy
print("Best C value:", grid_search.best_params_['C'])
print("Best Accuracy:", grid_search.best_score_)

# Save the trained model to a file
joblib.dump(grid_search.best_estimator_, '/content/modello_logistic_regression.pkl')
```

```
Best C value: 10
Best Accuracy: 0.7494285714285716
```

Model development

Logistic Regression



Model Development

Logistic Regression

	Variable	Coefficient
0	mean_thickness	0.595021
1	std_thickness	0.033897
2	endpoints	-2.008272
3	branch_points	-0.425716
4	avg_segment_length	0.269593
5	total_skeleton_length	3.089507
6	intersection_density	-1.004600
7	endpoint_density	0.764931
8	intersection_to_endpoint_ratio	-1.827263
9	avg_gray_intensity	1.015616
10	gray_std_dev	-1.537367
11	entropy	-0.242810
12	contrast	-3.337074
13	dissimilarity	4.235216
14	homogeneity	2.085572
15	energy	0.586764
16	correlation	-3.337074
17	average_curvature	0.256519
18	curvature_change	-0.263073
19	average_power_spectrum	-0.292733
20	binary_activity	-3.407931

Model development

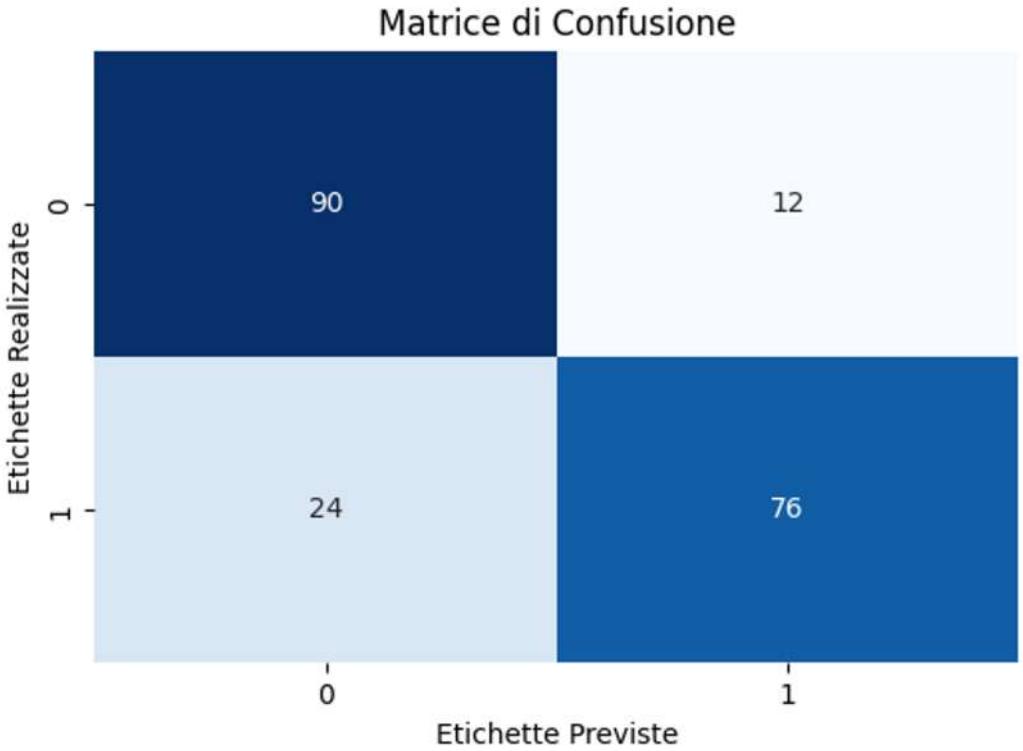
Ridge Regression

```
# Create the logistic regression model
model_ridge = LogisticRegression(penalty='l2', solver='liblinear')

# Define the parameter grid for Ridge regression (alpha values)
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100]}

# Prepare the cross-validation procedure
cv = RepeatedKFold(n_splits=10, n_repeats=5, random_state=1)

# Create the GridSearchCV object to find the best C value
grid_search = GridSearchCV(model_ridge, param_grid, scoring='accuracy', cv=cv, n_jobs=-1)

# Fit the model with cross-validation
grid_search.fit(X, y)

# Display the best C value and corresponding accuracy
print('Best C value: ', grid_search.best_params_['C'])
print('Best Accuracy: %.3f' % grid_search.best_score_)

# Save the trained model to a file
joblib.dump(grid_search.best_estimator_, '/content/model_ridge.pkl')
```

```
Best C value:  100
Best Accuracy: 0.763
```

Model development

Ridge Regression



Model development

Ridge Regression

	Variable	Coefficient
0	mean_thickness	1.295667
1	std_thickness	-0.284514
2	endpoints	-10.382955
3	branch_points	-1.825895
4	avg_segment_length	-0.114713
5	total_skeleton_length	12.013801
6	intersection_density	-4.084239
7	endpoint_density	3.637668
8	intersection_to_endpoint_ratio	-4.175698
9	avg_gray_intensity	1.345481
10	gray_std_dev	-5.119997
11	entropy	1.988745
12	contrast	-9.241060
13	dissimilarity	9.039770
14	homogeneity	-2.638754
15	energy	5.047114
16	correlation	-9.241060
17	average_curvature	0.150507
18	curvature_change	-0.264004
19	average_power_spectrum	-0.281926
20	binary_activity	-4.464066

Model development

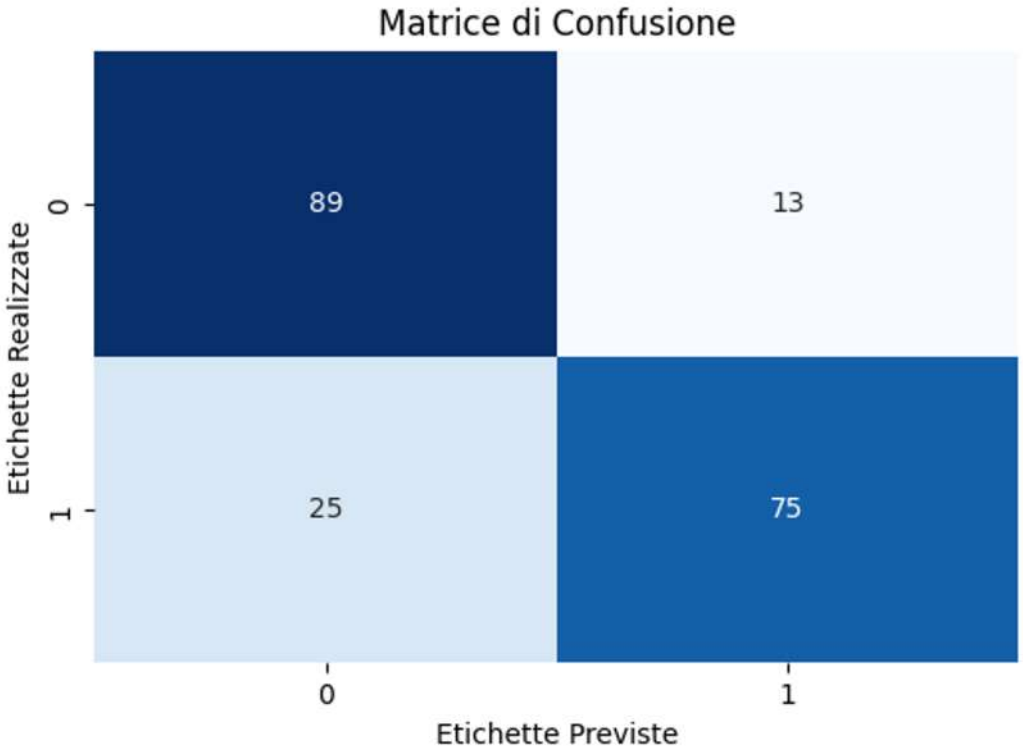
Lasso Regression

```
# Create a logistic regression model with L1 penalty (Lasso)
model_lasso = LogisticRegression(penalty='l1', solver='liblinear')

# Define a grid of C values for cross-validation
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10]}

# Prepare the cross-validation procedure
cv = RepeatedKFold(n_splits=10, n_repeats=5, random_state=1)

# Create GridSearchCV to search for the best C value
grid_search = GridSearchCV(estimator=model_lasso, param_grid=param_grid, scoring='accuracy', cv=cv, n_jobs=-1)

# Fit the model to the data using GridSearchCV
grid_search.fit(X, y)

# Print the best C value and corresponding accuracy
print("Best C value:", grid_search.best_params_['C'])
print("Best Accuracy:", grid_search.best_score_)

joblib.dump(grid_search.best_estimator_, '/content/model_lasso.pkl')
```

```
Best C value: 10
Best Accuracy: 0.7522380952380954
```

Model development

Lasso Regression



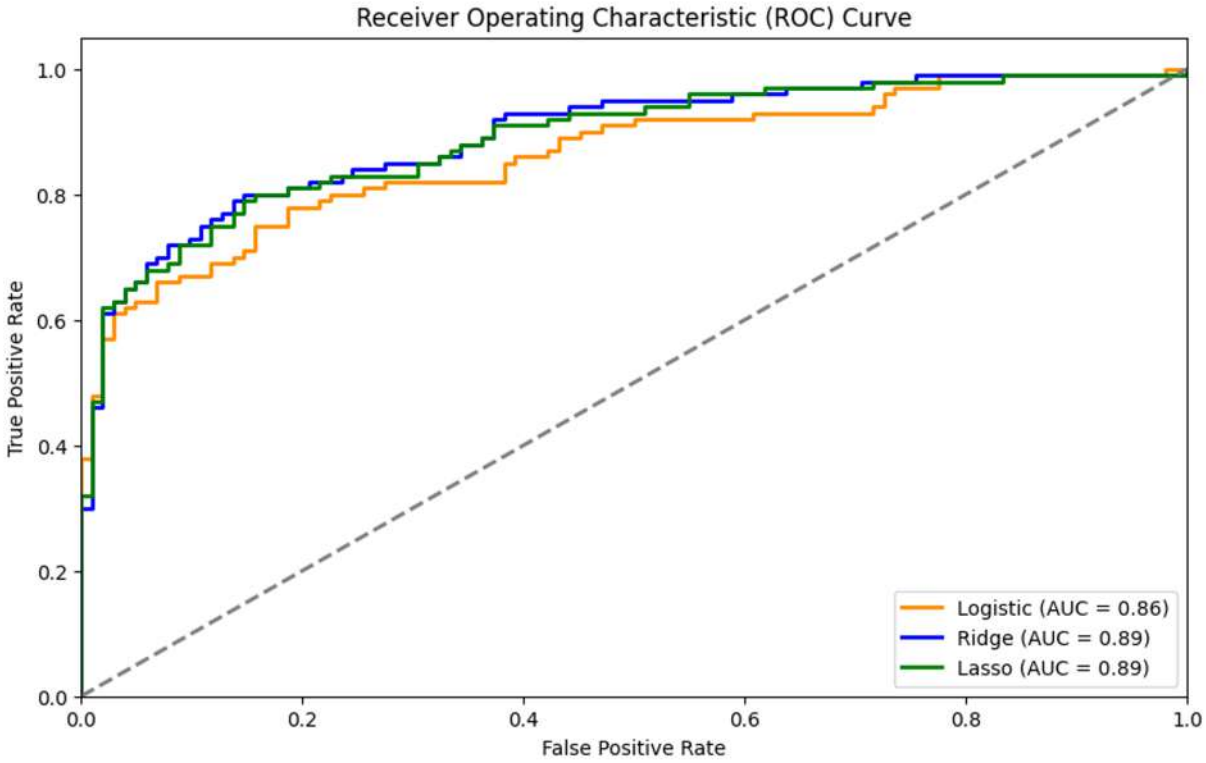
Model development

Lasso Regression

	Variable	Coefficient
0	mean_thickness	0.935361
1	std_thickness	-0.275028
2	endpoints	-8.833026
3	branch_points	-3.324016
4	avg_segment_length	0.000000
5	total_skeleton_length	18.569818
6	intersection_density	-0.072685
7	endpoint_density	0.000000
8	intersection_to_endpoint_ratio	-4.144266
9	avg_gray_intensity	0.979863
10	gray_std_dev	-1.997881
11	entropy	0.000000
12	contrast	-8.833215
13	dissimilarity	4.105749
14	homogeneity	-2.817610
15	energy	0.000000
16	correlation	-8.690074
17	average_curvature	0.175130
18	curvature_change	-0.261915
19	average_power_spectrum	-0.259653
20	binary_activity	-4.496131

Model Evaluation

Roc Curve



Improvements and Next Steps

Applying Data Augmentation

Improving hyperparameters optimization

carry out text mining and topic modeling on different sites

Improving performances

Incorporating additional data sources

Thank you!

Any questions?