Analiza danych z UC Irvine Machine Learning Repository

Wprowadzenie

Celem niniejszego projektu jest analiza danych z UC Irvine Machine Learning Repository przy użyciu różnych algorytmów klasyfikacji i klasteryzacji. Wykorzystane zostały zestawy danych o identyfikatorach 17, 53 oraz 850. Projekt obejmuje pobieranie danych, wstępne przetwarzanie, a następnie zastosowanie różnych algorytmów uczenia maszynowego w celu oceny ich skuteczności.

Wymagania

- Pobieranie danych z UC Irvine Machine Learning Repository.
- Wstępne przetwarzanie danych, w tym skalowanie i podział na zestawy treningowe i testowe.
- Implementacja i ocena różnych algorytmów klasyfikacji takich jak • Klasyfikator wielowarstwowy perceptronowy (MLPClassifier), • Regresja logistyczna (LogisticRegression), • Drzewo decyzyjne (DecisionTreeClassifier), • Liniowa klasyfikacja metodą wektorów nośnych (LinearSVC), • Klasyfikacja metodą stochastycznego gradientu (SGDClassifier), • Gaussowski Naive Bayes (GaussianNB).
- Zapisanie przetworzonych danych do pliku.
- Generowanie raportów klasyfikacji i dokładności dla każdego algorytmu.

Metoda

Pobieranie danych

Dane są pobierane z UC Irvine Machine Learning Repository za pomocą funkcji `fetch_ucirepo` z pakietu `ucimlrepo`. Jeśli dane zostały już pobrane wcześniej, są one wczytywane z pliku.

Wstępne przetwarzanie

Dane są skalowane do zakresu [-1, 1] za pomocą `MinMaxScaler`. Następnie dane są dzielone na zestawy treningowe i testowe z użyciem funkcji `train_test_split`.

Implementacja algorytmów klasyfikacji

Zaimplementowano i oceniono następujące algorytmy klasyfikacji:

Rodzaj klasyfikacji	Klasyfikator wielowarstwowy perceptronowy	Regresja logistyczna	Klasyfikacja metodą stochastycznego gradientu
Klasa scikit-learn	sklearn.neural_network.MLPClassifier	sklearn.linear_model.LogisticRegression	sklearn.naive_bayes.GaussianNB
Parametry	hidden_layer_sizes = (10, 10, 10) activation = relu solver = adam alpha = 0.0001 batch_size = auto learning_rate = constant learning_rate_init = 0.001 power_t = 0.5 max_iter = 1000 shuffle = True random_state = None tol = 0.0001 verbose = True warm_start = False momentum = 0.9 nesterovs_momentum = True early_stopping = False validation_fraction = 0.1 beta_1 = 0.9 beta_2 = 0.999 epsilon = 1e-08 n_iter_no_change = 10 max_fun = 15000	penalty = l2 dual = False tol = 0.0001 C = 1.0 fit_intercept = True intercept_scaling = 1 class_weight = None random_state = None solver = newton-cholesky max_iter = 100 multi_class = auto verbose = True warm_start = False n_jobs = None l1_ratio = None	loss = hinge penalty = l2 alpha = 0.0001 l1_ratio = 0.15 fit_intercept = True max_iter = 1000 tol = 0.001 shuffle = True verbose = True epsilon = 0.1 n_jobs = None random_state = None learning_rate = optimal eta0 = 0.0 power_t = 0.5 early_stopping = False validation_fraction = 0.1 n_iter_no_change = 5 class_weight = None warm_start = False average = False

Rodzaj klasyfikacji	Drzewo decyzyjne	Liniowa klasyfikacja metodą wektorów nośnych	Gaussian Naive Bayes
Klasa scikit-learn	sklearn.tree.DecisionTreeClassifier	sklearn.svm.LinearSVC	sklearn.linear_model.SGDClassifier
Parametry	criterion = gini splitter = best max_depth = None min_samples_split = 2 min_samples_leaf = 1 min_weight_fraction_leaf = 0.0 max_features = None random_state = None max_leaf_nodes = None min_impurity_decrease = 0.0 class_weight = None ccp_alpha = 0.0	penalty = l2 loss = squared_hinge dual = auto tol = 0.0001 C = 1.0 multi_class = ovr fit_intercept = True intercept_scaling = 1 class_weight = None verbose = 0 random_state = None max_iter = 1000	priors = None var_smoothing = 1e-09

Dla każdego algorytmu wyświetlono parametry, czas treningu oraz raport klasyfikacji zawierający dokładność i inne metryki.

Implementacja

1. Pobranie lub wczytanie danych:

```
dataset = fetch_ucirepo(id=dataset_id)
```

2. Skalowanie danych:

```
scaler = MinMaxScaler(feature_range=(-1, 1))
for column in features.columns:
    if features[column].dtype in ['int64', 'float64'] and features[column].nunique() > 1:
        features.loc[:, column] = scaler.fit_transform(features[column].values.reshape(-1, 1))
```

3. Podział danych na zestawy treningowe i testowe:

```
X_train, X_test, y_train, y_test = train_test_split(features, targets, test_size=0.25, stratify=ds['targets'])
```

4. Implementacja i trening algorytmów klasyfikacji:

```
mlpclassifier = MLPClassifier(hidden_layer_sizes=(10,10,10), verbose=True)
mlpclassifier.fit(X_train, y_train)
```

Analogicznie dla pozostałych algorytmów.

5. Testowanie algorytmów i generowanie raportów:

```
y_pred = mlpclassifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred, zero_division=0)
```

Analogicznie dla pozostałych algorytmów.

Zbieranie wyników

Ocena wyników każdego algorytmu obejmuje:

- Czas wykonania

- Dokładność (accuracy)
- Raport klasyfikacji (classification report) zawierający metryki takie jak precision, recall, F1-score.

Wyniki te pozwalają na porównanie skuteczności różnych algorytmów w kontekście analizy danych z wybranego zestawu.

Użyteczność wyników

Wyniki są specjalnie wypisywane w języku angielskim, aby zapewnić większą użyteczność. Programiści na całym świecie mogą w ten sposób z łatwością zrozumieć raporty i wyniki analizy, co sprzyja współpracy międzynarodowej i wymianie wiedzy.

Program generuje raport w formacie MD (Markdown). Markdown to lekki język znaczników, który umożliwia tworzenie tekstów sformatowanych w sposób czytelny zarówno dla ludzi, jak i dla maszyn. Najprostszym sposobem wyświetlania i edycji plików Markdown jest użycie Visual Studio Code (VS Code). Visual Studio Code to popularne, darmowe środowisko programistyczne, które oferuje wsparcie dla Markdown poprzez podgląd w czasie rzeczywistym (Ctrl + K, V) oraz liczne rozszerzenia, które ułatwiają pracę z tym formatem.

Przykładowe wyniki wykonania

Dataset	Dataset 17	Dataset 53	Dataset 850
Abstract	Diagnostic Wisconsin Breast Cancer Database.	A small classic dataset from Fisher, 1936. One of the earliest known datasets used for evaluating classification methods.	Images of the Kecimen and Besni raisin varieties were obtained with CVS. A total of 900 raisins were used, including 450 from both varieties, and 7 morphological features were extracted.
Summary	Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. A few of the images can be found at http://www.cs.wisc.edu/~street/images/ ...	This is one of the earliest datasets used in the literature on classification methods and widely used in statistics and machine learning. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are not linearly separable from each other. ...	Images of Kecimen and Besni raisin varieties grown in Turkey were obtained with CVS. A total of 900 raisin grains were used, including 450 pieces from both varieties. These images were subjected to various stages of pre-processing and 7 morphological features were extracted. These features have been classified using three different artificial intelligence techniques.
Data URL	https://archive.ics.uci.edu/static/public/17/data.csv	https://archive.ics.uci.edu/static/public/53/data.csv	https://archive.ics.uci.edu/static/public/850/data.csv

<table>
<tr><th>Dataset</th><th>Dataset 17</th><th>Dataset 53</th><th>Dataset 850</th></tr>
<tr><td>Variables</td>
<td><pre>   name   role   type
1   Diagnosis   Target Categorical
2   radius1 Feature   Continuous
3   texture1 Feature   Continuous
4   perimeter1 Feature   Continuous
5   area1 Feature   Continuous
6   smoothness1 Feature   Continuous
7   compactness1 Feature   Continuous
8   concavity1 Feature   Continuous
9   concave_points1 Feature   Continuous
10   symmetry1 Feature   Continuous
11 fractal_dimension1 Feature   Continuous
12   radius2 Feature   Continuous
13   texture2 Feature   Continuous
14   perimeter2 Feature   Continuous
15   area2 Feature   Continuous
16   smoothness2 Feature   Continuous
17   compactness2 Feature   Continuous
18   concavity2 Feature   Continuous
19   concave_points2 Feature   Continuous
20   symmetry2 Feature   Continuous
21 fractal_dimension2 Feature   Continuous
22   radius3 Feature   Continuous
23   texture3 Feature   Continuous
24   perimeter3 Feature   Continuous
25   area3 Feature   Continuous
26   smoothness3 Feature   Continuous
27   compactness3 Feature   Continuous
28   concavity3 Feature   Continuous
29   concave_points3 Feature   Continuous
30   symmetry3 Feature   Continuous
31 fractal_dimension3 Feature   Continuous</pre></td>
<td><pre>   name   role
0 sepal length Feature
1   sepal width Feature
2 petal length Feature
3   petal width Feature
4   class   Target</pre></td>
<td><pre>   name   role
0   Area Feature
1 MajorAxisLength Feature
2 MinorAxisLength Feature
3   Eccentricity Feature
4   ConvexArea Feature
5   Extent Feature
6   Perimeter Feature
7   Class   Target</pre></td>
</tr>
</table>

Dataset	Dataset 17				
		precision	recall	f1-score	support
MLPClassifier	B	0.98	0.97	0.97	90
	M	0.94	0.96	0.95	53
	accuracy			0.97	143
	macro avg	0.96	0.96	0.96	143
	weighted avg	0.97	0.97	0.97	143
LogisticRegression	B	0.98	0.99	0.98	90
	M	0.98	0.96	0.97	53
	accuracy			0.98	143
	macro avg	0.98	0.98	0.98	143
	weighted avg	0.98	0.98	0.98	143
DecisionTreeClassifier	B	0.97	0.96	0.96	90
	M	0.93	0.94	0.93	53
	accuracy			0.95	143
	macro avg	0.95	0.95	0.95	143
	weighted avg	0.95	0.95	0.95	143

Dataset	Dataset 53				
		precision	recall	f1-score	support
MLPClassifier	Iris-setosa	1.00	1.00	1.00	13
	Iris-versicolor	1.00	0.92	0.96	13
	Iris-virginica	0.92	1.00	0.96	12
	accuracy			0.97	38
	macro avg	0.97	0.97	0.97	38
	weighted avg	0.98	0.97	0.97	38
LogisticRegression	Iris-setosa	1.00	1.00	1.00	13
	Iris-versicolor	1.00	0.92	0.96	13
	Iris-virginica	0.92	1.00	0.96	12
	accuracy			0.97	38
	macro avg	0.97	0.97	0.97	38
	weighted avg	0.98	0.97	0.97	38
DecisionTreeClassifier	Iris-setosa	1.00	1.00	1.00	13
	Iris-versicolor	1.00	0.92	0.96	13
	Iris-virginica	0.92	1.00	0.96	12
	accuracy			0.97	38
	macro avg	0.97	0.97	0.97	38
	weighted avg	0.98	0.97	0.97	38

Dataset	Dataset 850				
		precision	recall	f1-score	support
MLPClassifier	Besni	0.91	0.81	0.86	112
	Kecimen	0.83	0.92	0.87	113
	accuracy			0.87	225
	macro avg	0.87	0.87	0.87	225
	weighted avg	0.87	0.87	0.87	225
LogisticRegression	Besni	0.89	0.84	0.86	112
	Kecimen	0.85	0.89	0.87	113
	accuracy			0.87	225
	macro avg	0.87	0.87	0.87	225
	weighted avg	0.87	0.87	0.87	225
DecisionTreeClassifier	Besni	0.79	0.76	0.77	112
	Kecimen	0.77	0.80	0.78	113
	accuracy			0.78	225
	macro avg	0.78	0.78	0.78	225
	weighted avg	0.78	0.78	0.78	225

<table>
<tr><th>Dataset</th><th>Dataset 17</th><th>Dataset 53</th><th>Dataset 850</th></tr>
<tr><td>LinearSVC</td>
<td><table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>B</td><td>0.95</td><td>1.00</td><td>0.97</td><td>90</td></tr>
<tr><td>M</td><td>1.00</td><td>0.91</td><td>0.95</td><td>53</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.97</td><td>143</td></tr>
<tr><td>macro avg</td><td>0.97</td><td>0.95</td><td>0.96</td><td>143</td></tr>
<tr><td>weighted avg</td><td>0.97</td><td>0.97</td><td>0.96</td><td>143</td></tr>
</table></td>
<td><table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>Iris-setosa</td><td>1.00</td><td>1.00</td><td>1.00</td><td>13</td></tr>
<tr><td>Iris-versicolor</td><td>1.00</td><td>0.92</td><td>0.96</td><td>13</td></tr>
<tr><td>Iris-virginica</td><td>0.92</td><td>1.00</td><td>0.96</td><td>12</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.97</td><td>38</td></tr>
<tr><td>macro avg</td><td>0.97</td><td>0.97</td><td>0.97</td><td>38</td></tr>
<tr><td>weighted avg</td><td>0.98</td><td>0.97</td><td>0.97</td><td>38</td></tr>
</table></td>
<td><table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>Besni</td><td>0.89</td><td>0.83</td><td>0.86</td><td>112</td></tr>
<tr><td>Kecimen</td><td>0.84</td><td>0.89</td><td>0.87</td><td>113</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.86</td><td>225</td></tr>
<tr><td>macro avg</td><td>0.86</td><td>0.86</td><td>0.86</td><td>225</td></tr>
<tr><td>weighted avg</td><td>0.86</td><td>0.86</td><td>0.86</td><td>225</td></tr>
</table></td></tr>
<tr><td>SGDClassifier</td>
<td><table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>B</td><td>0.98</td><td>0.98</td><td>0.98</td><td>90</td></tr>
<tr><td>M</td><td>0.96</td><td>0.96</td><td>0.96</td><td>53</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.97</td><td>143</td></tr>
<tr><td>macro avg</td><td>0.97</td><td>0.97</td><td>0.97</td><td>143</td></tr>
<tr><td>weighted avg</td><td>0.97</td><td>0.97</td><td>0.97</td><td>143</td></tr>
</table></td>
<td><table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>Iris-setosa</td><td>1.00</td><td>1.00</td><td>1.00</td><td>13</td></tr>
<tr><td>Iris-versicolor</td><td>1.00</td><td>0.92</td><td>0.96</td><td>13</td></tr>
<tr><td>Iris-virginica</td><td>0.92</td><td>1.00</td><td>0.96</td><td>12</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.97</td><td>38</td></tr>
<tr><td>macro avg</td><td>0.97</td><td>0.97</td><td>0.97</td><td>38</td></tr>
<tr><td>weighted avg</td><td>0.98</td><td>0.97</td><td>0.97</td><td>38</td></tr>
</table></td>
<td><table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>Besni</td><td>0.91</td><td>0.80</td><td>0.85</td><td>112</td></tr>
<tr><td>Kecimen</td><td>0.83</td><td>0.92</td><td>0.87</td><td>113</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.86</td><td>225</td></tr>
<tr><td>macro avg</td><td>0.87</td><td>0.86</td><td>0.86</td><td>225</td></tr>
<tr><td>weighted avg</td><td>0.87</td><td>0.86</td><td>0.86</td><td>225</td></tr>
</table></td></tr>
<tr><td>GaussianNB</td>
<td><table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>B</td><td>0.96</td><td>0.97</td><td>0.96</td><td>90</td></tr>
<tr><td>M</td><td>0.94</td><td>0.92</td><td>0.93</td><td>53</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.95</td><td>143</td></tr>
<tr><td>macro avg</td><td>0.95</td><td>0.95</td><td>0.95</td><td>143</td></tr>
<tr><td>weighted avg</td><td>0.95</td><td>0.95</td><td>0.95</td><td>143</td></tr>
</table></td>
<td><table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>Iris-setosa</td><td>1.00</td><td>1.00</td><td>1.00</td><td>13</td></tr>
<tr><td>Iris-versicolor</td><td>1.00</td><td>1.00</td><td>1.00</td><td>13</td></tr>
<tr><td>Iris-virginica</td><td>1.00</td><td>1.00</td><td>1.00</td><td>12</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>1.00</td><td>38</td></tr>
<tr><td>macro avg</td><td>1.00</td><td>1.00</td><td>1.00</td><td>38</td></tr>
<tr><td>weighted avg</td><td>1.00</td><td>1.00</td><td>1.00</td><td>38</td></tr>
</table></td>
<td><table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>Besni</td><td>0.95</td><td>0.79</td><td>0.86</td><td>112</td></tr>
<tr><td>Kecimen</td><td>0.82</td><td>0.96</td><td>0.88</td><td>113</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.87</td><td>225</td></tr>
<tr><td>macro avg</td><td>0.88</td><td>0.87</td><td>0.87</td><td>225</td></tr>
<tr><td>weighted avg</td><td>0.88</td><td>0.87</td><td>0.87</td><td>225</td></tr>
</table></td></tr>
</table>

Podsumowanie

W projekcie przeprowadzono analizę trzech zestawów danych z UC Irvine Machine Learning Repository, wykorzystując różne algorytmy klasyfikacji i klasteryzacji. Wykorzystane zestawy danych to: Diagnostic Wisconsin Breast Cancer Database, klasyczny zestaw danych od Fishera z 1936 roku oraz zestaw danych dotyczący rodzajów rodzynków (Kecimen i Besni).

Główne etapy projektu obejmowały:

- Pobieranie danych.
- Wstępne przetwarzanie danych.
- Implementacja i ocena algorytmów klasyfikacji.
- Generowanie raportów.

Projekt pokazał skuteczność różnych algorytmów klasyfikacji w analizie danych z UC Irvine Machine Learning Repository. Najlepsze wyniki w większości przypadków osiągnięto przy użyciu regresji logistycznej i klasyfikatora MLP. W przypadku niektórych zestawów danych, Gaussian Naive Bayes również wykazał bardzo dobrą skuteczność. Wyniki te mogą służyć jako baza do dalszych badań i optymalizacji algorytmów klasyfikacji w kontekście różnych zestawów danych.

Autorzy

- Ali Dávid - s26567@st.pans.nysa.pl
- Sieniawski Filip - s26594@st.pans.nysa.pl

Załączniki

1. Plik program.py

```
# -*- coding: utf-8 -*-
from ucimlrepo import fetch_ucirepo
from pprint import pprint
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
# from sklearn.cluster import KMeans
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import MinMaxScaler
import time
import sys
import pickle
from pathlib import Path
dataset_id = 53

f = open(str(dataset_id)+'.md', 'w')
sys.stdout = f
print(f"# Dataset {dataset_id}");
print("## Checking if file exists");
file_path = str(dataset_id)+".pkl"
path = Path(file_path)
if path.is_file():
    print(f' The file "{file_path}" exists.')
    print()
    print("## Loading dataset from file...");
    with open(file_path, 'rb') as file:
        ds = pickle.load(file)
    print(" Done.")
    print()
else:
    print(f' The file "{file_path}" does not exist.')
    print()
    print("## Downloading dataset...");
    start_time = time.time()
```

```
    dataset = fetch_ucirepo(id=dataset_id)
    end_time = time.time()
    execution_time = end_time - start_time
    print(f" Done in {execution_time} seconds.")
    print()
    ds = dict();
    ds["abstract"] = dataset.metadata.abstract #string
    ds["summary"] = dataset.metadata.additional_info.summary #string
    ds["url"] = dataset.metadata.data_url #string
    ds["variables"] = dataset.variables.copy() #DataFrame
    ds["features"] = dataset.data.features.copy() #DataFrame
    ds["targets"] = dataset.data.targets.copy() #DataFrame
    print("## Saving dataset to file...");
    # GENEROWANIE PLIKU CSV DO ANALIZY KORELACJI
    import pandas as pd
    alldata = pd.concat([ds["features"], ds["targets"]], axis=1)
    alldata.to_csv(str(dataset_id)+'.csv', index=False)
    with open(file_path, 'wb') as file:
        pickle.dump(ds, file)
    print(" Done.")
    print()
print("## Abstract")
print(f"{ds['abstract']}")
print()
print("## Summary")
print(f"<details>{ds['summary']}</details>")
print()
print("## Data URL")
print(f"{ds['url']}")
print()
print("## Variables")
print("```");
pprint(ds['variables'])
print("```")
print()
print("## Preprocessing");
print("### Scaling each column to fit [-1;1]...");
features = ds['features']
targets = ds['targets']
scaler = MinMaxScaler(feature_range=(-1, 1))
for column in features.columns:
    if features[column].dtype in ['int64', 'float64'] and features[column].nunique() > 1:
        features.loc[:, column] = scaler.fit_transform(features[column].values.reshape(-1, 1))
print(" Done.")
print()
```

```
print("### Splitting data into training and testing sets...");
X_train, X_test, y_train, y_test = train_test_split(features, targets, test_size=0.25, stratify=ds['targets'])
y_train = y_train.values.ravel()
y_test = y_test.values.ravel()
print(" Done.")
print()
print("## Results");
print("### Neural network");
print("#### Creating MLPClassifier...");
mlpclassifier = MLPClassifier(hidden_layer_sizes=(10,10,10), verbose=True, max_iter=1000)
print(" Done.")
print()
print("##### Parameters");
print("```")
print(" hidden_layer_sizes =", mlpclassifier.hidden_layer_sizes)
print("         activation =", mlpclassifier.activation)
print("             solver =", mlpclassifier.solver)
print("              alpha =", mlpclassifier.alpha)
print("         batch_size =", mlpclassifier.batch_size)
print("      learning_rate =", mlpclassifier.learning_rate)
print(" learning_rate_init =", mlpclassifier.learning_rate_init)
print("            power_t =", mlpclassifier.power_t)
print("           max_iter =", mlpclassifier.max_iter)
print("            shuffle =", mlpclassifier.shuffle)
print("       random_state =", mlpclassifier.random_state)
print("                tol =", mlpclassifier.tol)
print("            verbose =", mlpclassifier.verbose)
print("         warm_start =", mlpclassifier.warm_start)
print("           momentum =", mlpclassifier.momentum)
print(" nesterovs_momentum =", mlpclassifier.nesterovs_momentum)
print("     early_stopping =", mlpclassifier.early_stopping)
print("validation_fraction =", mlpclassifier.validation_fraction)
print("             beta_1 =", mlpclassifier.beta_1)
print("             beta_2 =", mlpclassifier.beta_2)
print("            epsilon =", mlpclassifier.epsilon)
print("   n_iter_no_change =", mlpclassifier.n_iter_no_change)
print("            max_fun =", mlpclassifier.max_fun)
print("```")
print()
print("#### Training...");
if(mlpclassifier.verbose):
    print("<details><pre>")
start_time = time.time()
mlpclassifier.fit(X_train, y_train)
end_time = time.time()
```

```
execution_time = end_time - start_time
if(mlpclassifier.verbose):
    print("</pre>
    </details>")
    print(f" Done in {execution_time} seconds.")
    print()
    print("#### Testing...");
    y_pred = mlpclassifier.predict(X_test)
    print(" Done.");
    print()
    print("#### Accuracy")
    accuracy = accuracy_score(y_test, y_pred)
    print(f" {accuracy}")
    print()
    print("#### Classification report")
    classification_rep = classification_report(y_test, y_pred, zero_division=0)
    classification_rep_dict = classification_report(y_test, y_pred, zero_division=0, output_dict=True)
    print("```")
    print(f"{classification_rep}")
    print("```")
    print()

    print("### Logistic regression");
    print("#### Creating LogisticRegression...");
    logisticregression = LogisticRegression(verbose=True, solver='newton-cholesky')
    print(" Done.")
    print()
    print("##### Parameters");
    print("```")
    print(" penalty =", logisticregression.penalty)
    print(" dual =", logisticregression.dual)
    print(" tol =", logisticregression.tol)
    print(" C =", logisticregression.C)
    print(" fit_intercept =", logisticregression.fit_intercept)
    print("intercept_scaling =", logisticregression.intercept_scaling)
    print(" class_weight =", logisticregression.class_weight)
    print(" random_state =", logisticregression.random_state)
    print(" solver =", logisticregression.solver)
    print(" max_iter =", logisticregression.max_iter)
    print(" multi_class =", logisticregression.multi_class)
    print(" verbose =", logisticregression.verbose)
    print(" warm_start =", logisticregression.warm_start)
    print(" n_jobs =", logisticregression.n_jobs)
    print(" l1_ratio =", logisticregression.l1_ratio)
    print("```")
```

```
    print()
    print("#### Training...");
    if(logisticregression.verbose):
    print("<details>
        <pre>")
start_time = time.time()
logisticregression.fit(X_train, y_train)
end_time = time.time()
execution_time = end_time - start_time
if(logisticregression.verbose):
    print("</pre>
    </details>")
    print(f" Done in {execution_time} seconds.")
    print()
    print("#### Testing...");
    y_pred = logisticregression.predict(X_test)
    print(" Done.");
    print()
    print("#### Accuracy")
    accuracy = accuracy_score(y_test, y_pred)
    print(f" {accuracy}")
    print()
    print("#### Classification report")
    classification_rep = classification_report(y_test, y_pred, zero_division=0)
    classification_rep_dict = classification_report(y_test, y_pred, zero_division=0, output_dict=True)
    print("```")
    print(f"{classification_rep}")
    print("```")
    print()

    print("### Decision tree classifier");
    print("#### Creating DecisionTreeClassifier...");
    decisiontreeclassifier = DecisionTreeClassifier()
    print(" Done.")
    print()
    print("##### Parameters");
    print("```")
    print(" criterion =", decisiontreeclassifier.criterion)
    print(" splitter =", decisiontreeclassifier.splitter)
    print(" max_depth =", decisiontreeclassifier.max_depth)
    print(" min_samples_split =", decisiontreeclassifier.min_samples_split)
    print(" min_samples_leaf =", decisiontreeclassifier.min_samples_leaf)
    print("min_weight_fraction_leaf =", decisiontreeclassifier.min_weight_fraction_leaf)
    print(" max_features =", decisiontreeclassifier.max_features)
    print(" random_state =", decisiontreeclassifier.random_state)
```

```
print(" max_leaf_nodes =", decisiontreeclassifier.max_leaf_nodes)
print(" min_impurity_decrease =", decisiontreeclassifier.min_impurity_decrease)
print(" class_weight =", decisiontreeclassifier.class_weight)
print(" ccp_alpha =", decisiontreeclassifier.ccp_alpha)
print("```")
print()
print("#### Training...");
start_time = time.time()
decisiontreeclassifier.fit(X_train, y_train)
end_time = time.time()
execution_time = end_time - start_time
print(f" Done in {execution_time} seconds.")
print()
print("#### Testing...");
y_pred = decisiontreeclassifier.predict(X_test)
print(" Done.");
print()
print("#### Accuracy")
accuracy = accuracy_score(y_test, y_pred)
print(f" {accuracy}")
print()
print("#### Classification report")
classification_rep = classification_report(y_test, y_pred, zero_division=0)
classification_rep_dict = classification_report(y_test, y_pred, zero_division=0, output_dict=True)
print("```")
print(f"{classification_rep}")
print("```")
print()

# print("### KMeans");
# print("#### Creating KMeans...");
# kmeans = KMeans(n_clusters = 2, verbose=True, n_init='auto')
# print(" Done.")
# print()
# print("##### Parameters");
# print("```")
# print(" n_clusters =", kmeans.n_clusters)
# print(" init =", kmeans.init)
# print(" n_init =", kmeans.n_init)
# print(" max_iter =", kmeans.max_iter)
# print(" tol =", kmeans.tol)
# print(" verbose =", kmeans.verbose)
# print("random_state =", kmeans.random_state)
# print(" copy_x =", kmeans.copy_x)
# print(" algorithm =", kmeans.algorithm)
```

```
# print("```")
# print()
# print("#### Training...");
# if(kmeans.verbose):
# print("```")
# start_time = time.time()
# kmeans.fit(X_train, y_train)
# end_time = time.time()
# execution_time = end_time - start_time
# if(kmeans.verbose):
# print("```")
# print(f" Done in {execution_time} seconds.")
# print()
# print("#### Testing...");
# y_pred = kmeans.predict(X_test)
# print(" Done.");
# print()
# print("#### Accuracy")
# accuracy = accuracy_score(y_test, y_pred)
# print(f" {accuracy}")
# print()
# print("#### Classification report")
# classification_rep = classification_report(y_test, y_pred, zero_division=0)
# classification_rep_dict = classification_report(y_test, y_pred, zero_division=0, output_dict=True)
# print("```")
# print(f"{classification_rep}")
# print("```")
# print()


print("### Linear Support Vector Classification");
print("#### Creating LinearSVC...");
svc = LinearSVC(dual='auto')
print(" Done.")
print()
print("##### Parameters");
print("```")
print(" penalty =", svc.penalty)
print(" loss =", svc.loss)
print(" dual =", svc.dual)
print(" tol =", svc.tol)
print(" C =", svc.C)
print(" multi_class =", svc.multi_class)
print(" fit_intercept =", svc.fit_intercept)
print("intercept_scaling =", svc.intercept_scaling)
```

```
    print(" class_weight =", svc.class_weight)
    print(" verbose =", svc.verbose)
    print(" random_state =", svc.random_state)
    print(" max_iter =", svc.max_iter)
    print("```")
    print()
    print("#### Training...");

    if(svc.verbose):
    print("<details>
        <pre>")
start_time = time.time()
svc.fit(X_train, y_train)
end_time = time.time()
execution_time = end_time - start_time
if(svc.verbose):
    print("</pre>
    </details>")
    print(f" Done in {execution_time} seconds.")
    print()
    print("#### Testing...");
    y_pred = svc.predict(X_test)
    print(" Done.");
    print()
    print("#### Accuracy")
    accuracy = accuracy_score(y_test, y_pred)
    print(f" {accuracy}")
    print()
    print("#### Classification report")
    classification_rep = classification_report(y_test, y_pred, zero_division=0)
    classification_rep_dict = classification_report(y_test, y_pred, zero_division=0, output_dict=True)
    print("```")
    print(f"{classification_rep}")
    print("```")
    print()

    print("### Stochastic Gradient Descent Classification");
    print("#### Creating SGDClassifier...");
    sgd = SGDClassifier(verbose=True)
    print(" Done.")
    print()
    print("##### Parameters");
    print("```")
    print(" loss =", sgd.loss)
    print(" penalty =", sgd.penalty)
```

```
    print(" alpha =", sgd.alpha)
    print(" l1_ratio =", sgd.l1_ratio)
    print(" fit_intercept =", sgd.fit_intercept)
    print(" max_iter =", sgd.max_iter)
    print(" tol =", sgd.tol)
    print(" shuffle =", sgd.shuffle)
    print(" verbose =", sgd.verbose)
    print(" epsilon =", sgd.epsilon)
    print(" n_jobs =", sgd.n_jobs)
    print(" random_state =", sgd.random_state)
    print(" learning_rate =", sgd.learning_rate)
    print(" eta0 =", sgd.eta0)
    print(" power_t =", sgd.power_t)
    print(" early_stopping =", sgd.early_stopping)
    print("validation_fraction =", sgd.validation_fraction)
    print(" n_iter_no_change =", sgd.n_iter_no_change)
    print(" class_weight =", sgd.class_weight)
    print(" warm_start =", sgd.warm_start)
    print(" average =", sgd.average)
    print("```")
    print()
    print("#### Training...");

    if(sgd.verbose):
    print("<details>
        <pre>")
start_time = time.time()
sgd.fit(X_train, y_train)
end_time = time.time()
execution_time = end_time - start_time
if(sgd.verbose):
    print("</pre>
    </details>")
    print(f" Done in {execution_time} seconds.")
    print()
    print("#### Testing...");
    y_pred = sgd.predict(X_test)
    print(" Done.");
    print()
    print("#### Accuracy")
    accuracy = accuracy_score(y_test, y_pred)
    print(f" {accuracy}")
    print()
    print("#### Classification report")
    classification_rep = classification_report(y_test, y_pred, zero_division=0)
```

```
classification_rep_dict = classification_report(y_test, y_pred, zero_division=0, output_dict=True)
print("```")
print(f"{classification_rep}")
print("```")
print()

print("### Gaussian Naive Bayes");
print("#### Creating GaussianNB...");
gnb = GaussianNB()
print(" Done.")
print()
print("##### Parameters");
print("```")
print(" priors =", gnb.priors)
print("var_smoothing =", gnb.var_smoothing)
print("```")
print()
print("#### Training...");
start_time = time.time()
gnb.fit(X_train, y_train)
end_time = time.time()
execution_time = end_time - start_time
print(f" Done in {execution_time} seconds.")
print()
print("#### Testing...");
y_pred = gnb.predict(X_test)
print(" Done.");
print()
print("#### Accuracy")
accuracy = accuracy_score(y_test, y_pred)
print(f" {accuracy}")
print()
print("#### Classification report")
classification_rep = classification_report(y_test, y_pred, zero_division=0)
classification_rep_dict = classification_report(y_test, y_pred, zero_division=0, output_dict=True)
print("```")
print(f"{classification_rep}")
print("```")
print()

f.close()
sys.stdout = sys.__stdout__
```

2. Plik wynikowy 17.md

```
# Dataset 17
## Checking if file exists
 The file "17.pkl" exists.

## Loading dataset from file...
 Done.

## Abstract
Diagnostic Wisconsin Breast Cancer Database.

## Summary
Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass.
They describe characteristics of the cell nuclei present in the image. A few of the images can
be found at http://www.cs.wisc.edu/~street/images/



Separating plane described above was obtained using Multisurface Method-Tree (MSM-T)
[K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a
classification method which uses linear programming to construct a decision tree.  Relevant
features were selected using an exhaustive search in the space of 1-4 features and 1-3
separating planes.



The actual linear program used to obtain the separating plane in the 3-dimensional space is that
described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of
Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].



This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu

cd math-prog/cpo-dataset/machine-learn/WDBC/

## Data URL
https://archive.ics.uci.edu/static/public/17/data.csv

## Variables
```

```
                 name     role         type  ... description units missing_values
0                  ID       ID  Categorical  ...        None  None             no
1           Diagnosis   Target  Categorical  ...        None  None             no
2             radius1  Feature   Continuous  ...        None  None             no
3            texture1  Feature   Continuous  ...        None  None             no
4          perimeter1  Feature   Continuous  ...        None  None             no
5               area1  Feature   Continuous  ...        None  None             no
6         smoothness1  Feature   Continuous  ...        None  None             no
7        compactness1  Feature   Continuous  ...        None  None             no
8          concavity1  Feature   Continuous  ...        None  None             no
9     concave_points1  Feature   Continuous  ...        None  None             no
10          symmetry1  Feature   Continuous  ...        None  None             no
11  fractal_dimension1  Feature   Continuous  ...        None  None             no
12            radius2  Feature   Continuous  ...        None  None             no
13           texture2  Feature   Continuous  ...        None  None             no
14         perimeter2  Feature   Continuous  ...        None  None             no
15              area2  Feature   Continuous  ...        None  None             no
16        smoothness2  Feature   Continuous  ...        None  None             no
17       compactness2  Feature   Continuous  ...        None  None             no
18         concavity2  Feature   Continuous  ...        None  None             no
19    concave_points2  Feature   Continuous  ...        None  None             no
20          symmetry2  Feature   Continuous  ...        None  None             no
21  fractal_dimension2  Feature   Continuous  ...        None  None             no
22            radius3  Feature   Continuous  ...        None  None             no
23           texture3  Feature   Continuous  ...        None  None             no
24         perimeter3  Feature   Continuous  ...        None  None             no
25              area3  Feature   Continuous  ...        None  None             no
26        smoothness3  Feature   Continuous  ...        None  None             no
27       compactness3  Feature   Continuous  ...        None  None             no
28         concavity3  Feature   Continuous  ...        None  None             no
29    concave_points3  Feature   Continuous  ...        None  None             no
30          symmetry3  Feature   Continuous  ...        None  None             no
31  fractal_dimension3  Feature   Continuous  ...        None  None             no

[32 rows x 7 columns]
```

Preprocessing
Scaling each column to fit [-1;1]...
 Done.

Splitting data into training and testing sets...
 Done.

Results
Neural network
Creating MLPClassifier...
Done.

Parameters
```
 hidden_layer_sizes = (10, 10, 10)
         activation = relu
             solver = adam
              alpha = 0.0001
         batch_size = auto
      learning_rate = constant
 learning_rate_init = 0.001
            power_t = 0.5
           max_iter = 1000
            shuffle = True
       random_state = None
                tol = 0.0001
            verbose = True
         warm_start = False
           momentum = 0.9
 nesterovs_momentum = True
     early_stopping = False
validation_fraction = 0.1
             beta_1 = 0.9
             beta_2 = 0.999
            epsilon = 1e-08
   n_iter_no_change = 10
            max_fun = 15000
```

Training...
<details><pre>
Iteration 1, loss = 0.79318970
Iteration 2, loss = 0.77281687
Iteration 3, loss = 0.75485578
Iteration 4, loss = 0.73813877
Iteration 5, loss = 0.72321656
Iteration 6, loss = 0.70984249
Iteration 7, loss = 0.69673938
Iteration 8, loss = 0.68457113
Iteration 9, loss = 0.67243592
Iteration 10, loss = 0.66170762
Iteration 11, loss = 0.65124676

```
Iteration 12, loss = 0.64114736
Iteration 13, loss = 0.63134251
Iteration 14, loss = 0.62141141
Iteration 15, loss = 0.61213324
Iteration 16, loss = 0.60300229
Iteration 17, loss = 0.59397621
Iteration 18, loss = 0.58533275
Iteration 19, loss = 0.57705325
Iteration 20, loss = 0.56903845
Iteration 21, loss = 0.56063071
Iteration 22, loss = 0.55194550
Iteration 23, loss = 0.54309666
Iteration 24, loss = 0.53434064
Iteration 25, loss = 0.52552460
Iteration 26, loss = 0.51630970
Iteration 27, loss = 0.50702941
Iteration 28, loss = 0.49761035
Iteration 29, loss = 0.48826054
Iteration 30, loss = 0.47845634
Iteration 31, loss = 0.46893992
Iteration 32, loss = 0.45936869
Iteration 33, loss = 0.44988533
Iteration 34, loss = 0.44041468
Iteration 35, loss = 0.43076213
Iteration 36, loss = 0.42122982
Iteration 37, loss = 0.41114946
Iteration 38, loss = 0.40085622
Iteration 39, loss = 0.39046047
Iteration 40, loss = 0.37890189
Iteration 41, loss = 0.36695597
Iteration 42, loss = 0.35412181
Iteration 43, loss = 0.34153287
Iteration 44, loss = 0.32744894
Iteration 45, loss = 0.31353895
Iteration 46, loss = 0.30035616
Iteration 47, loss = 0.28916427
Iteration 48, loss = 0.27792905
Iteration 49, loss = 0.26727737
Iteration 50, loss = 0.25749651
Iteration 51, loss = 0.24754797
Iteration 52, loss = 0.23789711
Iteration 53, loss = 0.22945171
Iteration 54, loss = 0.22159411
Iteration 55, loss = 0.21373340
Iteration 56, loss = 0.20522993
```

```
Iteration 57, loss = 0.19758095
Iteration 58, loss = 0.19105494
Iteration 59, loss = 0.18448303
Iteration 60, loss = 0.17876516
Iteration 61, loss = 0.17359365
Iteration 62, loss = 0.16886336
Iteration 63, loss = 0.16513594
Iteration 64, loss = 0.16063431
Iteration 65, loss = 0.15571612
Iteration 66, loss = 0.15216140
Iteration 67, loss = 0.15061282
Iteration 68, loss = 0.14767618
Iteration 69, loss = 0.14293160
Iteration 70, loss = 0.13926787
Iteration 71, loss = 0.13735682
Iteration 72, loss = 0.13520618
Iteration 73, loss = 0.13288871
Iteration 74, loss = 0.13056193
Iteration 75, loss = 0.12855350
Iteration 76, loss = 0.12637963
Iteration 77, loss = 0.12469616
Iteration 78, loss = 0.12255715
Iteration 79, loss = 0.12305219
Iteration 80, loss = 0.12289718
Iteration 81, loss = 0.12150943
Iteration 82, loss = 0.11915397
Iteration 83, loss = 0.11700827
Iteration 84, loss = 0.11620707
Iteration 85, loss = 0.11534167
Iteration 86, loss = 0.11423876
Iteration 87, loss = 0.11333104
Iteration 88, loss = 0.11238549
Iteration 89, loss = 0.11160090
Iteration 90, loss = 0.11093097
Iteration 91, loss = 0.11056216
Iteration 92, loss = 0.11066434
Iteration 93, loss = 0.11048897
Iteration 94, loss = 0.10934572
Iteration 95, loss = 0.10770557
Iteration 96, loss = 0.10761087
Iteration 97, loss = 0.10602068
Iteration 98, loss = 0.10530316
Iteration 99, loss = 0.10706604
Iteration 100, loss = 0.10765472
Iteration 101, loss = 0.10712533
```

```
Iteration 102, loss = 0.10572441
Iteration 103, loss = 0.10353249
Iteration 104, loss = 0.10241405
Iteration 105, loss = 0.10199687
Iteration 106, loss = 0.10070828
Iteration 107, loss = 0.10169653
Iteration 108, loss = 0.10441552
Iteration 109, loss = 0.10147786
Iteration 110, loss = 0.09989709
Iteration 111, loss = 0.09991496
Iteration 112, loss = 0.10043760
Iteration 113, loss = 0.10007845
Iteration 114, loss = 0.09825951
Iteration 115, loss = 0.09903275
Iteration 116, loss = 0.09924464
Iteration 117, loss = 0.09834711
Iteration 118, loss = 0.09739717
Iteration 119, loss = 0.09680360
Iteration 120, loss = 0.09650406
Iteration 121, loss = 0.09616322
Iteration 122, loss = 0.09619639
Iteration 123, loss = 0.09596816
Iteration 124, loss = 0.09528125
Iteration 125, loss = 0.09505061
Iteration 126, loss = 0.09611868
Iteration 127, loss = 0.09570917
Iteration 128, loss = 0.09447389
Iteration 129, loss = 0.09414875
Iteration 130, loss = 0.09489077
Iteration 131, loss = 0.09454027
Iteration 132, loss = 0.09412389
Iteration 133, loss = 0.09377990
Iteration 134, loss = 0.09350420
Iteration 135, loss = 0.09366088
Iteration 136, loss = 0.09349472
Iteration 137, loss = 0.09301752
Iteration 138, loss = 0.09239404
Iteration 139, loss = 0.09224480
Iteration 140, loss = 0.09202357
Iteration 141, loss = 0.09178104
Iteration 142, loss = 0.09206358
Iteration 143, loss = 0.09194765
Iteration 144, loss = 0.09189124
Iteration 145, loss = 0.09180111
Iteration 146, loss = 0.09202747
```

```
Iteration 147, loss = 0.09156602
Iteration 148, loss = 0.09208365
Iteration 149, loss = 0.09130392
Iteration 150, loss = 0.09120224
Iteration 151, loss = 0.09113770
Iteration 152, loss = 0.09064151
Iteration 153, loss = 0.09047276
Iteration 154, loss = 0.09001393
Iteration 155, loss = 0.09031675
Iteration 156, loss = 0.09060417
Iteration 157, loss = 0.08997928
Iteration 158, loss = 0.08948944
Iteration 159, loss = 0.08957174
Iteration 160, loss = 0.08959321
Iteration 161, loss = 0.08956188
Iteration 162, loss = 0.08979138
Iteration 163, loss = 0.08957678
Iteration 164, loss = 0.08901666
Iteration 165, loss = 0.08879994
Iteration 166, loss = 0.08857367
Iteration 167, loss = 0.08857415
Iteration 168, loss = 0.08871169
Iteration 169, loss = 0.08884043
Iteration 170, loss = 0.08888056
Iteration 171, loss = 0.08859412
Iteration 172, loss = 0.08845107
Iteration 173, loss = 0.08805374
Iteration 174, loss = 0.08741686
Iteration 175, loss = 0.08859733
Iteration 176, loss = 0.08751436
Iteration 177, loss = 0.08898637
Iteration 178, loss = 0.09071206
Iteration 179, loss = 0.08887258
Iteration 180, loss = 0.08778475
Iteration 181, loss = 0.08731942
Iteration 182, loss = 0.08696337
Iteration 183, loss = 0.08672466
Iteration 184, loss = 0.08716972
Iteration 185, loss = 0.08701399
Iteration 186, loss = 0.08684757
Iteration 187, loss = 0.08689697
Iteration 188, loss = 0.08775157
Iteration 189, loss = 0.08726986
Iteration 190, loss = 0.08581942
Iteration 191, loss = 0.08853682
```

Iteration 192, loss = 0.09026800
Iteration 193, loss = 0.08854992
Iteration 194, loss = 0.08619627
Iteration 195, loss = 0.08576170
Iteration 196, loss = 0.08609124
Iteration 197, loss = 0.08593356
Iteration 198, loss = 0.08556383
Iteration 199, loss = 0.08531930
Iteration 200, loss = 0.08526001
Iteration 201, loss = 0.08518595
Iteration 202, loss = 0.08511926
Iteration 203, loss = 0.08522304
Iteration 204, loss = 0.08507998
Iteration 205, loss = 0.08499476
Iteration 206, loss = 0.08500778
Iteration 207, loss = 0.08506146
Iteration 208, loss = 0.08520803
Iteration 209, loss = 0.08586896
Iteration 210, loss = 0.08493516
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
</pre></details>
 Done in 0.2954387664794922 seconds.

Testing...
 Done.

Accuracy
 0.965034965034965

Classification report
```
              precision    recall  f1-score   support

           B       0.98      0.97      0.97        90
           M       0.94      0.96      0.95        53

    accuracy                           0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.97      0.97      0.97       143

```

Logistic regression
Creating LogisticRegression...
 Done.

Parameters
```
          penalty = l2
             dual = False
              tol = 0.0001
                C = 1.0
    fit_intercept = True
intercept_scaling = 1
     class_weight = None
     random_state = None
           solver = newton-cholesky
         max_iter = 100
      multi_class = auto
          verbose = True
       warm_start = False
           n_jobs = None
         l1_ratio = None
```

Training...
<details><pre>
Newton iter=1
  Check Convergence
    1. max |gradient| 0.051320700030266835 <= 0.0001
Newton iter=2
  Check Convergence
    1. max |gradient| 0.01797867964189787 <= 0.0001
Newton iter=3
  Check Convergence
    1. max |gradient| 0.005787159940580239 <= 0.0001
Newton iter=4
  Check Convergence
    1. max |gradient| 0.001258582138516263 <= 0.0001
Newton iter=5
  Check Convergence
    1. max |gradient| 8.650434266559062e-05 <= 0.0001
    2. Newton decrement 0.0004388692050945829 <= 0.0001
Newton iter=6
  Check Convergence
    1. max |gradient| 4.2551199829973796e-07 <= 0.0001
    2. Newton decrement 2.0381159484134227e-06 <= 0.0001
  Solver did converge at loss = 0.13249993946021482.
</pre></details>
 Done in 0.010075092315673828 seconds.

Testing...
 Done.

Accuracy
 0.9790209790209791

Classification report
```
              precision    recall  f1-score   support

           B       0.98      0.99      0.98        90
           M       0.98      0.96      0.97        53

    accuracy                           0.98       143
   macro avg       0.98      0.98      0.98       143
weighted avg       0.98      0.98      0.98       143

```

Decision tree classifier
Creating DecisionTreeClassifier...
 Done.

Parameters
```
               criterion = gini
                splitter = best
               max_depth = None
       min_samples_split = 2
        min_samples_leaf = 1
min_weight_fraction_leaf = 0.0
            max_features = None
            random_state = None
          max_leaf_nodes = None
   min_impurity_decrease = 0.0
            class_weight = None
               ccp_alpha = 0.0
```

Training...
 Done in 0.021494626998901367 seconds.

Testing...
 Done.

Accuracy
 0.951048951048951

Classification report
```
              precision    recall  f1-score   support

           B       0.97      0.96      0.96        90
           M       0.93      0.94      0.93        53

    accuracy                           0.95       143
   macro avg       0.95      0.95      0.95       143
weighted avg       0.95      0.95      0.95       143

```

Linear Support Vector Classification
Creating LinearSVC...
 Done.

Parameters
```
          penalty = l2
             loss = squared_hinge
             dual = auto
              tol = 0.0001
                C = 1.0
      multi_class = ovr
    fit_intercept = True
intercept_scaling = 1
     class_weight = None
          verbose = 0
     random_state = None
         max_iter = 1000
```

Training...
 Done in 0.002929210662841797 seconds.

Testing...
 Done.

Accuracy
 0.965034965034965

Classification report
```
              precision    recall  f1-score   support

           B       0.95      1.00      0.97        90
           M       1.00      0.91      0.95        53

    accuracy                           0.97       143
   macro avg       0.97      0.95      0.96       143
weighted avg       0.97      0.97      0.96       143

```

Stochastic Gradient Descent Classification
Creating SGDClassifier...
 Done.

Parameters
```
               loss = hinge
            penalty = l2
              alpha = 0.0001
           l1_ratio = 0.15
      fit_intercept = True
           max_iter = 1000
                tol = 0.001
            shuffle = True
            verbose = True
            epsilon = 0.1
             n_jobs = None
       random_state = None
      learning_rate = optimal
               eta0 = 0.0
            power_t = 0.5
     early_stopping = False
validation_fraction = 0.1
   n_iter_no_change = 5
       class_weight = None
         warm_start = False
            average = False
```

Training...
<details><pre>

```
-- Epoch 1
Norm: 105.09, NNZs: 30, Bias: 35.804302, T: 426, Avg. loss: 3.861311
Total training time: 0.00 seconds.
-- Epoch 2
Norm: 100.13, NNZs: 30, Bias: 41.631351, T: 852, Avg. loss: 1.685037
Total training time: 0.00 seconds.
-- Epoch 3
Norm: 92.78, NNZs: 30, Bias: 46.557907, T: 1278, Avg. loss: 1.360758
Total training time: 0.00 seconds.
-- Epoch 4
Norm: 88.24, NNZs: 30, Bias: 46.708110, T: 1704, Avg. loss: 1.154795
Total training time: 0.00 seconds.
-- Epoch 5
Norm: 83.60, NNZs: 30, Bias: 50.405620, T: 2130, Avg. loss: 1.003877
Total training time: 0.00 seconds.
-- Epoch 6
Norm: 73.33, NNZs: 30, Bias: 50.414789, T: 2556, Avg. loss: 0.781050
Total training time: 0.00 seconds.
-- Epoch 7
Norm: 70.60, NNZs: 30, Bias: 50.398290, T: 2982, Avg. loss: 0.586121
Total training time: 0.00 seconds.
-- Epoch 8
Norm: 68.32, NNZs: 30, Bias: 50.491579, T: 3408, Avg. loss: 0.755739
Total training time: 0.00 seconds.
-- Epoch 9
Norm: 62.27, NNZs: 30, Bias: 52.632852, T: 3834, Avg. loss: 0.584968
Total training time: 0.00 seconds.
-- Epoch 10
Norm: 61.46, NNZs: 30, Bias: 52.622073, T: 4260, Avg. loss: 0.543819
Total training time: 0.00 seconds.
-- Epoch 11
Norm: 59.17, NNZs: 30, Bias: 52.628435, T: 4686, Avg. loss: 0.350682
Total training time: 0.00 seconds.
-- Epoch 12
Norm: 56.91, NNZs: 30, Bias: 50.964656, T: 5112, Avg. loss: 0.388474
Total training time: 0.00 seconds.
-- Epoch 13
Norm: 54.97, NNZs: 30, Bias: 51.037227, T: 5538, Avg. loss: 0.374892
Total training time: 0.00 seconds.
-- Epoch 14
Norm: 52.12, NNZs: 30, Bias: 52.554830, T: 5964, Avg. loss: 0.407384
Total training time: 0.00 seconds.
-- Epoch 15
Norm: 50.31, NNZs: 30, Bias: 52.557903, T: 6390, Avg. loss: 0.323786
Total training time: 0.00 seconds.
```

```
-- Epoch 16
Norm: 49.30, NNZs: 30, Bias: 51.262973, T: 6816, Avg. loss: 0.343624
Total training time: 0.00 seconds.
-- Epoch 17
Norm: 47.14, NNZs: 30, Bias: 52.533996, T: 7242, Avg. loss: 0.278711
Total training time: 0.00 seconds.
-- Epoch 18
Norm: 46.54, NNZs: 30, Bias: 51.367039, T: 7668, Avg. loss: 0.295385
Total training time: 0.00 seconds.
-- Epoch 19
Norm: 43.94, NNZs: 30, Bias: 52.496664, T: 8094, Avg. loss: 0.298351
Total training time: 0.00 seconds.
-- Epoch 20
Norm: 44.23, NNZs: 30, Bias: 51.416831, T: 8520, Avg. loss: 0.283665
Total training time: 0.00 seconds.
-- Epoch 21
Norm: 43.69, NNZs: 30, Bias: 51.439574, T: 8946, Avg. loss: 0.295662
Total training time: 0.00 seconds.
-- Epoch 22
Norm: 40.99, NNZs: 30, Bias: 52.422027, T: 9372, Avg. loss: 0.232508
Total training time: 0.00 seconds.
-- Epoch 23
Norm: 40.17, NNZs: 30, Bias: 52.397735, T: 9798, Avg. loss: 0.242462
Total training time: 0.00 seconds.
-- Epoch 24
Norm: 40.69, NNZs: 30, Bias: 51.501447, T: 10224, Avg. loss: 0.219125
Total training time: 0.00 seconds.
-- Epoch 25
Norm: 37.87, NNZs: 30, Bias: 53.239543, T: 10650, Avg. loss: 0.192681
Total training time: 0.00 seconds.
-- Epoch 26
Norm: 38.99, NNZs: 30, Bias: 51.538474, T: 11076, Avg. loss: 0.200018
Total training time: 0.00 seconds.
-- Epoch 27
Norm: 38.62, NNZs: 30, Bias: 51.527190, T: 11502, Avg. loss: 0.212885
Total training time: 0.00 seconds.
-- Epoch 28
Norm: 39.14, NNZs: 30, Bias: 50.742396, T: 11928, Avg. loss: 0.174657
Total training time: 0.00 seconds.
-- Epoch 29
Norm: 37.87, NNZs: 30, Bias: 50.739173, T: 12354, Avg. loss: 0.146064
Total training time: 0.00 seconds.
-- Epoch 30
Norm: 37.55, NNZs: 30, Bias: 50.731626, T: 12780, Avg. loss: 0.193404
Total training time: 0.00 seconds.
```

```
-- Epoch 31
Norm: 37.80, NNZs: 30, Bias: 50.028060, T: 13206, Avg. loss: 0.122128
Total training time: 0.00 seconds.
-- Epoch 32
Norm: 36.29, NNZs: 30, Bias: 50.726734, T: 13632, Avg. loss: 0.167694
Total training time: 0.00 seconds.
-- Epoch 33
Norm: 37.33, NNZs: 30, Bias: 49.381621, T: 14058, Avg. loss: 0.138491
Total training time: 0.00 seconds.
-- Epoch 34
Norm: 35.64, NNZs: 30, Bias: 50.046244, T: 14484, Avg. loss: 0.112943
Total training time: 0.00 seconds.
-- Epoch 35
Norm: 36.02, NNZs: 30, Bias: 49.418226, T: 14910, Avg. loss: 0.118134
Total training time: 0.00 seconds.
-- Epoch 36
Norm: 34.78, NNZs: 30, Bias: 50.039976, T: 15336, Avg. loss: 0.127287
Total training time: 0.00 seconds.
-- Epoch 37
Norm: 35.14, NNZs: 30, Bias: 49.438437, T: 15762, Avg. loss: 0.136513
Total training time: 0.00 seconds.
-- Epoch 38
Norm: 34.62, NNZs: 30, Bias: 49.441614, T: 16188, Avg. loss: 0.109847
Total training time: 0.00 seconds.
-- Epoch 39
Norm: 33.94, NNZs: 30, Bias: 49.441214, T: 16614, Avg. loss: 0.110828
Total training time: 0.00 seconds.
-- Epoch 40
Norm: 33.27, NNZs: 30, Bias: 49.439802, T: 17040, Avg. loss: 0.133479
Total training time: 0.00 seconds.
-- Epoch 41
Norm: 33.71, NNZs: 30, Bias: 48.889930, T: 17466, Avg. loss: 0.124680
Total training time: 0.00 seconds.
-- Epoch 42
Norm: 32.38, NNZs: 30, Bias: 49.420006, T: 17892, Avg. loss: 0.120488
Total training time: 0.00 seconds.
-- Epoch 43
Norm: 32.83, NNZs: 30, Bias: 48.887861, T: 18318, Avg. loss: 0.133354
Total training time: 0.00 seconds.
Convergence after 43 epochs took 0.00 seconds
</pre></details>
 Done in 0.0 seconds.

#### Testing...
 Done.
```

Accuracy
 0.972027972027972

Classification report
```
              precision    recall  f1-score   support

           B       0.98      0.98      0.98        90
           M       0.96      0.96      0.96        53

    accuracy                           0.97       143
   macro avg       0.97      0.97      0.97       143
weighted avg       0.97      0.97      0.97       143

```

Gaussian Naive Bayes
Creating GaussianNB...
 Done.

Parameters
```
       priors = None
var_smoothing = 1e-09
```

Training...
 Done in 0.0 seconds.

Testing...
 Done.

Accuracy
 0.951048951048951

Classification report
```
              precision    recall  f1-score   support

           B       0.96      0.97      0.96        90
           M       0.94      0.92      0.93        53

    accuracy                           0.95       143
   macro avg       0.95      0.95      0.95       143
```

```
weighted avg       0.95      0.95      0.95       143
```

3. Plik wynikowy 53.md

~~~~
# Dataset 53
## Checking if file exists
 The file "53.pkl" exists.

## Loading dataset from file...
 Done.

## Abstract
A small classic dataset from Fisher, 1936. One of the earliest known datasets used for
evaluating classification methods.


## Summary
<details>This is one of the earliest datasets used in the literature on classification
methods and widely used in statistics and machine learning.  The data set contains 3 classes
of 50 instances each, where each class refers to a type of iris plant.  One class is linearly
separable from the other 2; the latter are not linearly separable from each other.

Predicted attribute: class of iris plant.

This is an exceedingly simple domain.

This data differs from the data presented in Fishers article (identified by Steve Chadwick,
spchadwick@espeedaz.net ).  The 35th sample should be: 4.9,3.1,1.5,0.2,"Iris-setosa" where the
error is in the fourth feature. The 38th sample: 4.9,3.6,1.4,0.1,"Iris-setosa" where the errors
are in the second and third features.  </details>

## Data URL
https://archive.ics.uci.edu/static/public/53/data.csv

## Variables
```
           name     role  ... units missing_values
0  sepal length  Feature  ...    cm             no
1   sepal width  Feature  ...    cm             no
2  petal length  Feature  ...    cm             no
3   petal width  Feature  ...    cm             no
4         class   Target  ...  None             no

[5 rows x 7 columns]
```
~~~~

Preprocessing
Scaling each column to fit [-1;1]...
Done.

Splitting data into training and testing sets...
Done.

Results
Neural network
Creating MLPClassifier...
Done.

Parameters
```
 hidden_layer_sizes = (10, 10, 10)
         activation = relu
             solver = adam
              alpha = 0.0001
         batch_size = auto
      learning_rate = constant
 learning_rate_init = 0.001
            power_t = 0.5
           max_iter = 1000
            shuffle = True
       random_state = None
                tol = 0.0001
            verbose = True
         warm_start = False
           momentum = 0.9
 nesterovs_momentum = True
     early_stopping = False
validation_fraction = 0.1
             beta_1 = 0.9
             beta_2 = 0.999
            epsilon = 1e-08
   n_iter_no_change = 10
            max_fun = 15000
```

Training...
<details><pre>
Iteration 1, loss = 1.15852887
Iteration 2, loss = 1.15397585
Iteration 3, loss = 1.14950344
Iteration 4, loss = 1.14518846

```
Iteration 5, loss = 1.14101004
Iteration 6, loss = 1.13687861
Iteration 7, loss = 1.13283763
Iteration 8, loss = 1.12884800
Iteration 9, loss = 1.12487606
Iteration 10, loss = 1.12101031
Iteration 11, loss = 1.11723535
Iteration 12, loss = 1.11348245
Iteration 13, loss = 1.10974304
Iteration 14, loss = 1.10597852
Iteration 15, loss = 1.10224609
Iteration 16, loss = 1.09852983
Iteration 17, loss = 1.09483147
Iteration 18, loss = 1.09111304
Iteration 19, loss = 1.08733699
Iteration 20, loss = 1.08357627
Iteration 21, loss = 1.07978217
Iteration 22, loss = 1.07596304
Iteration 23, loss = 1.07214383
Iteration 24, loss = 1.06830322
Iteration 25, loss = 1.06448876
Iteration 26, loss = 1.06074725
Iteration 27, loss = 1.05710845
Iteration 28, loss = 1.05366393
Iteration 29, loss = 1.05026806
Iteration 30, loss = 1.04698411
Iteration 31, loss = 1.04388826
Iteration 32, loss = 1.04087379
Iteration 33, loss = 1.03799407
Iteration 34, loss = 1.03530845
Iteration 35, loss = 1.03282748
Iteration 36, loss = 1.03047666
Iteration 37, loss = 1.02829263
Iteration 38, loss = 1.02620825
Iteration 39, loss = 1.02415470
Iteration 40, loss = 1.02214117
Iteration 41, loss = 1.02014212
Iteration 42, loss = 1.01816615
Iteration 43, loss = 1.01620718
Iteration 44, loss = 1.01425441
Iteration 45, loss = 1.01231556
Iteration 46, loss = 1.01038187
Iteration 47, loss = 1.00843270
Iteration 48, loss = 1.00646470
Iteration 49, loss = 1.00447373
```

```
Iteration 50, loss = 1.00246562
Iteration 51, loss = 1.00044676
Iteration 52, loss = 0.99841611
Iteration 53, loss = 0.99637314
Iteration 54, loss = 0.99429867
Iteration 55, loss = 0.99219797
Iteration 56, loss = 0.99006878
Iteration 57, loss = 0.98792360
Iteration 58, loss = 0.98576769
Iteration 59, loss = 0.98358800
Iteration 60, loss = 0.98137507
Iteration 61, loss = 0.97912996
Iteration 62, loss = 0.97684577
Iteration 63, loss = 0.97450710
Iteration 64, loss = 0.97212630
Iteration 65, loss = 0.96969827
Iteration 66, loss = 0.96723151
Iteration 67, loss = 0.96472863
Iteration 68, loss = 0.96217910
Iteration 69, loss = 0.95957436
Iteration 70, loss = 0.95692521
Iteration 71, loss = 0.95423320
Iteration 72, loss = 0.95150289
Iteration 73, loss = 0.94872938
Iteration 74, loss = 0.94590939
Iteration 75, loss = 0.94304540
Iteration 76, loss = 0.94012419
Iteration 77, loss = 0.93714380
Iteration 78, loss = 0.93411705
Iteration 79, loss = 0.93103841
Iteration 80, loss = 0.92790942
Iteration 81, loss = 0.92471864
Iteration 82, loss = 0.92146967
Iteration 83, loss = 0.91812562
Iteration 84, loss = 0.91471905
Iteration 85, loss = 0.91125254
Iteration 86, loss = 0.90772991
Iteration 87, loss = 0.90415259
Iteration 88, loss = 0.90051762
Iteration 89, loss = 0.89684141
Iteration 90, loss = 0.89311380
Iteration 91, loss = 0.88933258
Iteration 92, loss = 0.88550445
Iteration 93, loss = 0.88161632
Iteration 94, loss = 0.87765143
```

```
Iteration 95, loss = 0.87359179
Iteration 96, loss = 0.86945930
Iteration 97, loss = 0.86525089
Iteration 98, loss = 0.86095941
Iteration 99, loss = 0.85659181
Iteration 100, loss = 0.85215627
Iteration 101, loss = 0.84764434
Iteration 102, loss = 0.84305904
Iteration 103, loss = 0.83840461
Iteration 104, loss = 0.83365202
Iteration 105, loss = 0.82881888
Iteration 106, loss = 0.82388806
Iteration 107, loss = 0.81890200
Iteration 108, loss = 0.81385944
Iteration 109, loss = 0.80875638
Iteration 110, loss = 0.80358177
Iteration 111, loss = 0.79830146
Iteration 112, loss = 0.79294665
Iteration 113, loss = 0.78751342
Iteration 114, loss = 0.78193931
Iteration 115, loss = 0.77629531
Iteration 116, loss = 0.77060058
Iteration 117, loss = 0.76487102
Iteration 118, loss = 0.75908509
Iteration 119, loss = 0.75324322
Iteration 120, loss = 0.74736975
Iteration 121, loss = 0.74143561
Iteration 122, loss = 0.73547109
Iteration 123, loss = 0.72946010
Iteration 124, loss = 0.72341895
Iteration 125, loss = 0.71731373
Iteration 126, loss = 0.71116032
Iteration 127, loss = 0.70498383
Iteration 128, loss = 0.69879612
Iteration 129, loss = 0.69256678
Iteration 130, loss = 0.68631631
Iteration 131, loss = 0.68005704
Iteration 132, loss = 0.67377800
Iteration 133, loss = 0.66751105
Iteration 134, loss = 0.66125523
Iteration 135, loss = 0.65499648
Iteration 136, loss = 0.64871475
Iteration 137, loss = 0.64243754
Iteration 138, loss = 0.63616312
Iteration 139, loss = 0.62988179
```

```
Iteration 140, loss = 0.62360113
Iteration 141, loss = 0.61732369
Iteration 142, loss = 0.61104133
Iteration 143, loss = 0.60474970
Iteration 144, loss = 0.59846278
Iteration 145, loss = 0.59218185
Iteration 146, loss = 0.58590824
Iteration 147, loss = 0.57963757
Iteration 148, loss = 0.57338741
Iteration 149, loss = 0.56715168
Iteration 150, loss = 0.56093038
Iteration 151, loss = 0.55472710
Iteration 152, loss = 0.54853493
Iteration 153, loss = 0.54235915
Iteration 154, loss = 0.53619950
Iteration 155, loss = 0.53006269
Iteration 156, loss = 0.52394253
Iteration 157, loss = 0.51784495
Iteration 158, loss = 0.51176270
Iteration 159, loss = 0.50569756
Iteration 160, loss = 0.49967641
Iteration 161, loss = 0.49367281
Iteration 162, loss = 0.48768070
Iteration 163, loss = 0.48169802
Iteration 164, loss = 0.47572748
Iteration 165, loss = 0.46976777
Iteration 166, loss = 0.46380754
Iteration 167, loss = 0.45786255
Iteration 168, loss = 0.45193170
Iteration 169, loss = 0.44601771
Iteration 170, loss = 0.44011918
Iteration 171, loss = 0.43426430
Iteration 172, loss = 0.42843805
Iteration 173, loss = 0.42263629
Iteration 174, loss = 0.41685027
Iteration 175, loss = 0.41109043
Iteration 176, loss = 0.40537914
Iteration 177, loss = 0.39969129
Iteration 178, loss = 0.39403210
Iteration 179, loss = 0.38837447
Iteration 180, loss = 0.38272555
Iteration 181, loss = 0.37710348
Iteration 182, loss = 0.37151092
Iteration 183, loss = 0.36595953
Iteration 184, loss = 0.36044875
```

```
Iteration 185, loss = 0.35497126
Iteration 186, loss = 0.34951779
Iteration 187, loss = 0.34412415
Iteration 188, loss = 0.33878475
Iteration 189, loss = 0.33346345
Iteration 190, loss = 0.32817734
Iteration 191, loss = 0.32295391
Iteration 192, loss = 0.31779197
Iteration 193, loss = 0.31269006
Iteration 194, loss = 0.30765173
Iteration 195, loss = 0.30268409
Iteration 196, loss = 0.29778073
Iteration 197, loss = 0.29294327
Iteration 198, loss = 0.28817492
Iteration 199, loss = 0.28348219
Iteration 200, loss = 0.27886280
Iteration 201, loss = 0.27431506
Iteration 202, loss = 0.26984246
Iteration 203, loss = 0.26544989
Iteration 204, loss = 0.26114142
Iteration 205, loss = 0.25691513
Iteration 206, loss = 0.25276379
Iteration 207, loss = 0.24870166
Iteration 208, loss = 0.24474870
Iteration 209, loss = 0.24089632
Iteration 210, loss = 0.23711412
Iteration 211, loss = 0.23339889
Iteration 212, loss = 0.22974514
Iteration 213, loss = 0.22614609
Iteration 214, loss = 0.22261034
Iteration 215, loss = 0.21914531
Iteration 216, loss = 0.21574618
Iteration 217, loss = 0.21241318
Iteration 218, loss = 0.20915038
Iteration 219, loss = 0.20595984
Iteration 220, loss = 0.20284097
Iteration 221, loss = 0.19979295
Iteration 222, loss = 0.19681184
Iteration 223, loss = 0.19389444
Iteration 224, loss = 0.19103709
Iteration 225, loss = 0.18823734
Iteration 226, loss = 0.18549443
Iteration 227, loss = 0.18280683
Iteration 228, loss = 0.18017439
Iteration 229, loss = 0.17759561
```

```
Iteration 230, loss = 0.17506963
Iteration 231, loss = 0.17259653
Iteration 232, loss = 0.17017614
Iteration 233, loss = 0.16781592
Iteration 234, loss = 0.16551423
Iteration 235, loss = 0.16326662
Iteration 236, loss = 0.16107097
Iteration 237, loss = 0.15892291
Iteration 238, loss = 0.15682111
Iteration 239, loss = 0.15476516
Iteration 240, loss = 0.15275336
Iteration 241, loss = 0.15078564
Iteration 242, loss = 0.14886113
Iteration 243, loss = 0.14697807
Iteration 244, loss = 0.14513510
Iteration 245, loss = 0.14333212
Iteration 246, loss = 0.14156850
Iteration 247, loss = 0.13984286
Iteration 248, loss = 0.13815362
Iteration 249, loss = 0.13649949
Iteration 250, loss = 0.13487706
Iteration 251, loss = 0.13326994
Iteration 252, loss = 0.13169261
Iteration 253, loss = 0.13014508
Iteration 254, loss = 0.12862898
Iteration 255, loss = 0.12714173
Iteration 256, loss = 0.12568299
Iteration 257, loss = 0.12425148
Iteration 258, loss = 0.12284676
Iteration 259, loss = 0.12146829
Iteration 260, loss = 0.12011558
Iteration 261, loss = 0.11878844
Iteration 262, loss = 0.11748762
Iteration 263, loss = 0.11621378
Iteration 264, loss = 0.11496397
Iteration 265, loss = 0.11373801
Iteration 266, loss = 0.11253492
Iteration 267, loss = 0.11135439
Iteration 268, loss = 0.11019586
Iteration 269, loss = 0.10905972
Iteration 270, loss = 0.10794463
Iteration 271, loss = 0.10685019
Iteration 272, loss = 0.10577612
Iteration 273, loss = 0.10472584
Iteration 274, loss = 0.10369666
```

```
Iteration 275, loss = 0.10268637
Iteration 276, loss = 0.10169443
Iteration 277, loss = 0.10072042
Iteration 278, loss = 0.09976413
Iteration 279, loss = 0.09882478
Iteration 280, loss = 0.09790219
Iteration 281, loss = 0.09699611
Iteration 282, loss = 0.09610609
Iteration 283, loss = 0.09523171
Iteration 284, loss = 0.09437273
Iteration 285, loss = 0.09352890
Iteration 286, loss = 0.09269971
Iteration 287, loss = 0.09188513
Iteration 288, loss = 0.09108483
Iteration 289, loss = 0.09029835
Iteration 290, loss = 0.08952585
Iteration 291, loss = 0.08876650
Iteration 292, loss = 0.08801999
Iteration 293, loss = 0.08728602
Iteration 294, loss = 0.08656493
Iteration 295, loss = 0.08585664
Iteration 296, loss = 0.08516004
Iteration 297, loss = 0.08447527
Iteration 298, loss = 0.08380184
Iteration 299, loss = 0.08313917
Iteration 300, loss = 0.08248728
Iteration 301, loss = 0.08184595
Iteration 302, loss = 0.08121482
Iteration 303, loss = 0.08059367
Iteration 304, loss = 0.07998223
Iteration 305, loss = 0.07938018
Iteration 306, loss = 0.07878761
Iteration 307, loss = 0.07820481
Iteration 308, loss = 0.07763085
Iteration 309, loss = 0.07706557
Iteration 310, loss = 0.07650903
Iteration 311, loss = 0.07596072
Iteration 312, loss = 0.07542139
Iteration 313, loss = 0.07489026
Iteration 314, loss = 0.07436731
Iteration 315, loss = 0.07385202
Iteration 316, loss = 0.07334432
Iteration 317, loss = 0.07284437
Iteration 318, loss = 0.07235164
Iteration 319, loss = 0.07186598
```

```
Iteration 320, loss = 0.07138728
Iteration 321, loss = 0.07091541
Iteration 322, loss = 0.07045018
Iteration 323, loss = 0.06999174
Iteration 324, loss = 0.06953957
Iteration 325, loss = 0.06909350
Iteration 326, loss = 0.06865358
Iteration 327, loss = 0.06821967
Iteration 328, loss = 0.06779161
Iteration 329, loss = 0.06736934
Iteration 330, loss = 0.06695292
Iteration 331, loss = 0.06654191
Iteration 332, loss = 0.06613618
Iteration 333, loss = 0.06573968
Iteration 334, loss = 0.06534912
Iteration 335, loss = 0.06496236
Iteration 336, loss = 0.06457935
Iteration 337, loss = 0.06420062
Iteration 338, loss = 0.06382618
Iteration 339, loss = 0.06345728
Iteration 340, loss = 0.06309630
Iteration 341, loss = 0.06273973
Iteration 342, loss = 0.06238759
Iteration 343, loss = 0.06203975
Iteration 344, loss = 0.06169605
Iteration 345, loss = 0.06135647
Iteration 346, loss = 0.06102093
Iteration 347, loss = 0.06068928
Iteration 348, loss = 0.06036153
Iteration 349, loss = 0.06003785
Iteration 350, loss = 0.05971982
Iteration 351, loss = 0.05940436
Iteration 352, loss = 0.05909296
Iteration 353, loss = 0.05878590
Iteration 354, loss = 0.05848222
Iteration 355, loss = 0.05818184
Iteration 356, loss = 0.05788521
Iteration 357, loss = 0.05759300
Iteration 358, loss = 0.05730410
Iteration 359, loss = 0.05701807
Iteration 360, loss = 0.05673604
Iteration 361, loss = 0.05645692
Iteration 362, loss = 0.05618150
Iteration 363, loss = 0.05591038
Iteration 364, loss = 0.05564221
```

```
Iteration 365, loss = 0.05537740
Iteration 366, loss = 0.05511523
Iteration 367, loss = 0.05485625
Iteration 368, loss = 0.05460073
Iteration 369, loss = 0.05434713
Iteration 370, loss = 0.05409761
Iteration 371, loss = 0.05385156
Iteration 372, loss = 0.05360804
Iteration 373, loss = 0.05336678
Iteration 374, loss = 0.05312801
Iteration 375, loss = 0.05289213
Iteration 376, loss = 0.05265936
Iteration 377, loss = 0.05242882
Iteration 378, loss = 0.05220079
Iteration 379, loss = 0.05197542
Iteration 380, loss = 0.05175179
Iteration 381, loss = 0.05153111
Iteration 382, loss = 0.05131276
Iteration 383, loss = 0.05109651
Iteration 384, loss = 0.05088192
Iteration 385, loss = 0.05067165
Iteration 386, loss = 0.05046198
Iteration 387, loss = 0.05025306
Iteration 388, loss = 0.05004878
Iteration 389, loss = 0.04984627
Iteration 390, loss = 0.04964505
Iteration 391, loss = 0.04944511
Iteration 392, loss = 0.04924649
Iteration 393, loss = 0.04905021
Iteration 394, loss = 0.04885720
Iteration 395, loss = 0.04866454
Iteration 396, loss = 0.04847504
Iteration 397, loss = 0.04828741
Iteration 398, loss = 0.04810065
Iteration 399, loss = 0.04791494
Iteration 400, loss = 0.04773240
Iteration 401, loss = 0.04755159
Iteration 402, loss = 0.04737103
Iteration 403, loss = 0.04719332
Iteration 404, loss = 0.04701768
Iteration 405, loss = 0.04684276
Iteration 406, loss = 0.04666859
Iteration 407, loss = 0.04649755
Iteration 408, loss = 0.04632795
Iteration 409, loss = 0.04615836
```

```
Iteration 410, loss = 0.04599097
Iteration 411, loss = 0.04582610
Iteration 412, loss = 0.04566173
Iteration 413, loss = 0.04549787
Iteration 414, loss = 0.04533877
Iteration 415, loss = 0.04517964
Iteration 416, loss = 0.04502033
Iteration 417, loss = 0.04486097
Iteration 418, loss = 0.04470730
Iteration 419, loss = 0.04455419
Iteration 420, loss = 0.04440114
Iteration 421, loss = 0.04424827
Iteration 422, loss = 0.04409568
Iteration 423, loss = 0.04394483
Iteration 424, loss = 0.04379784
Iteration 425, loss = 0.04365139
Iteration 426, loss = 0.04350500
Iteration 427, loss = 0.04336351
Iteration 428, loss = 0.04322202
Iteration 429, loss = 0.04308037
Iteration 430, loss = 0.04293867
Iteration 431, loss = 0.04279705
Iteration 432, loss = 0.04266047
Iteration 433, loss = 0.04252457
Iteration 434, loss = 0.04238804
Iteration 435, loss = 0.04225178
Iteration 436, loss = 0.04211648
Iteration 437, loss = 0.04198528
Iteration 438, loss = 0.04185378
Iteration 439, loss = 0.04172206
Iteration 440, loss = 0.04159449
Iteration 441, loss = 0.04146697
Iteration 442, loss = 0.04133878
Iteration 443, loss = 0.04120992
Iteration 444, loss = 0.04108710
Iteration 445, loss = 0.04096393
Iteration 446, loss = 0.04084052
Iteration 447, loss = 0.04071674
Iteration 448, loss = 0.04059288
Iteration 449, loss = 0.04047473
Iteration 450, loss = 0.04035724
Iteration 451, loss = 0.04023900
Iteration 452, loss = 0.04012026
Iteration 453, loss = 0.04000106
Iteration 454, loss = 0.03988688
```

Iteration 455, loss = 0.03977391
Iteration 456, loss = 0.03965991
Iteration 457, loss = 0.03954526
Iteration 458, loss = 0.03943033
Iteration 459, loss = 0.03932249
Iteration 460, loss = 0.03921455
Iteration 461, loss = 0.03910554
Iteration 462, loss = 0.03899564
Iteration 463, loss = 0.03888520
Iteration 464, loss = 0.03877786
Iteration 465, loss = 0.03867359
Iteration 466, loss = 0.03856814
Iteration 467, loss = 0.03846161
Iteration 468, loss = 0.03835578
Iteration 469, loss = 0.03825345
Iteration 470, loss = 0.03815021
Iteration 471, loss = 0.03804683
Iteration 472, loss = 0.03794535
Iteration 473, loss = 0.03784526
Iteration 474, loss = 0.03774483
Iteration 475, loss = 0.03764404
Iteration 476, loss = 0.03754464
Iteration 477, loss = 0.03744735
Iteration 478, loss = 0.03734872
Iteration 479, loss = 0.03725229
Iteration 480, loss = 0.03715594
Iteration 481, loss = 0.03705857
Iteration 482, loss = 0.03696360
Iteration 483, loss = 0.03686909
Iteration 484, loss = 0.03677305
Iteration 485, loss = 0.03668024
Iteration 486, loss = 0.03658783
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
</pre></details>
Done in 0.24985241889953613 seconds.

Testing...
Done.

Accuracy
0.9473684210526315

Classification report
```
                 precision    recall  f1-score   support
```

    Iris-setosa   1.00   1.00   1.00   13
Iris-versicolor   0.92   0.92   0.92   13
 Iris-virginica   0.92   0.92   0.92   12

       accuracy   0.95   38
      macro avg   0.95   0.95   0.95   38
   weighted avg   0.95   0.95   0.95   38

```

### Logistic regression
#### Creating LogisticRegression...
 Done.

##### Parameters
```
          penalty = l2
             dual = False
              tol = 0.0001
                C = 1.0
    fit_intercept = True
intercept_scaling = 1
     class_weight = None
     random_state = None
           solver = newton-cholesky
         max_iter = 100
      multi_class = auto
          verbose = True
       warm_start = False
           n_jobs = None
         l1_ratio = None
```

#### Training...
<details><pre>
Newton iter=1
  Check Convergence
    1. max |gradient| 0.06031055998606637 <= 0.0001
Newton iter=2
  Check Convergence
    1. max |gradient| 0.018318948267569576 <= 0.0001
Newton iter=3
  Check Convergence
    1. max |gradient| 0.005851429988762207 <= 0.0001
```

```
Newton iter=4
  Check Convergence
    1. max |gradient| 0.0007925775117513846 <= 0.0001
Newton iter=5
  Check Convergence
    1. max |gradient| 1.3744563722827962e-05 <= 0.0001
    2. Newton decrement 2.0789501993220552e-05 <= 0.0001
  Solver did converge at loss = 0.10988913038482634.
Newton iter=1
  Check Convergence
    1. max |gradient| 0.024199175348732525 <= 0.0001
Newton iter=2
  Check Convergence
    1. max |gradient| 0.0023217339189821745 <= 0.0001
Newton iter=3
  Check Convergence
    1. max |gradient| 3.07431217252737e-05 <= 0.0001
    2. Newton decrement 6.084752897064173e-05 <= 0.0001
  Solver did converge at loss = 0.5239634736548608.
Newton iter=1
  Check Convergence
    1. max |gradient| 0.05842299393629161 <= 0.0001
Newton iter=2
  Check Convergence
    1. max |gradient| 0.020094790680868925 <= 0.0001
Newton iter=3
  Check Convergence
    1. max |gradient| 0.006048048404590795 <= 0.0001
Newton iter=4
  Check Convergence
    1. max |gradient| 0.000643705490211862 <= 0.0001
Newton iter=5
  Check Convergence
    1. max |gradient| 6.61935841182737e-06 <= 0.0001
    2. Newton decrement 3.338858481976473e-05 <= 0.0001
  Solver did converge at loss = 0.24819524728060488.
</pre></details>
 Done in 0.0021212100982666016 seconds.

#### Testing...
 Done.

#### Accuracy
 0.9210526315789473
```

Classification report
```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        13
Iris-versicolor       0.92      0.85      0.88        13
 Iris-virginica       0.85      0.92      0.88        12

       accuracy                           0.92        38
      macro avg       0.92      0.92      0.92        38
   weighted avg       0.92      0.92      0.92        38

```

Decision tree classifier
Creating DecisionTreeClassifier...
 Done.

Parameters
```
               criterion = gini
                splitter = best
               max_depth = None
       min_samples_split = 2
        min_samples_leaf = 1
min_weight_fraction_leaf = 0.0
            max_features = None
            random_state = None
          max_leaf_nodes = None
   min_impurity_decrease = 0.0
            class_weight = None
               ccp_alpha = 0.0
```

Training...
 Done in 0.008053064346313477 seconds.

Testing...
 Done.

Accuracy
 0.9736842105263158

Classification report
```
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        13
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.92      1.00      0.96        12

       accuracy                           0.97        38
      macro avg       0.97      0.97      0.97        38
   weighted avg       0.98      0.97      0.97        38

```

Linear Support Vector Classification
Creating LinearSVC...
 Done.

Parameters
```
          penalty = l2
             loss = squared_hinge
             dual = auto
              tol = 0.0001
                C = 1.0
      multi_class = ovr
    fit_intercept = True
intercept_scaling = 1
     class_weight = None
          verbose = 0
     random_state = None
         max_iter = 1000
```

Training...
 Done in 0.0 seconds.

Testing...
 Done.

Accuracy
 0.9473684210526315

Classification report
```
                 precision    recall  f1-score   support
```

    Iris-setosa   1.00   1.00   1.00   13
Iris-versicolor   0.92   0.92   0.92   13
 Iris-virginica   0.92   0.92   0.92   12

       accuracy   0.95   38
      macro avg   0.95   0.95   0.95   38
   weighted avg   0.95   0.95   0.95   38

```

### Stochastic Gradient Descent Classification
#### Creating SGDClassifier...
 Done.

##### Parameters
```
               loss = hinge
            penalty = l2
              alpha = 0.0001
           l1_ratio = 0.15
      fit_intercept = True
           max_iter = 1000
                tol = 0.001
            shuffle = True
            verbose = True
            epsilon = 0.1
             n_jobs = None
       random_state = None
      learning_rate = optimal
               eta0 = 0.0
            power_t = 0.5
     early_stopping = False
validation_fraction = 0.1
   n_iter_no_change = 5
       class_weight = None
         warm_start = False
            average = False
```

#### Training...
<details><pre>
-- Epoch 1
Norm: 15.41, NNZs: 4, Bias: -9.793001, T: 112, Avg. loss: 0.103351
Total training time: 0.00 seconds.
-- Epoch 2
```

```
Norm: 20.50, NNZs: 4, Bias: -18.278017, T: 224, Avg. loss: 0.045699
Total training time: 0.00 seconds.
-- Epoch 3
Norm: 24.44, NNZs: 4, Bias: -18.215820, T: 336, Avg. loss: 0.069730
Total training time: 0.00 seconds.
-- Epoch 4
Norm: 22.55, NNZs: 4, Bias: -18.215820, T: 448, Avg. loss: 0.000000
Total training time: 0.00 seconds.
-- Epoch 5
Norm: 20.93, NNZs: 4, Bias: -18.215820, T: 560, Avg. loss: 0.000000
Total training time: 0.00 seconds.
-- Epoch 6
Norm: 19.53, NNZs: 4, Bias: -18.215820, T: 672, Avg. loss: 0.000000
Total training time: 0.00 seconds.
-- Epoch 7
Norm: 23.17, NNZs: 4, Bias: -18.198529, T: 784, Avg. loss: 0.003851
Total training time: 0.00 seconds.
-- Epoch 8
Norm: 21.80, NNZs: 4, Bias: -18.198529, T: 896, Avg. loss: 0.000000
Total training time: 0.00 seconds.
-- Epoch 9
Norm: 20.58, NNZs: 4, Bias: -18.198529, T: 1008, Avg. loss: 0.000000
Total training time: 0.00 seconds.
Convergence after 9 epochs took 0.00 seconds
-- Epoch 1
Norm: 33.41, NNZs: 4, Bias: -0.968141, T: 112, Avg. loss: 3.609922
Total training time: 0.00 seconds.
-- Epoch 2
Norm: 32.58, NNZs: 4, Bias: -9.787298, T: 224, Avg. loss: 2.743476
Total training time: 0.00 seconds.
-- Epoch 3
Norm: 23.44, NNZs: 4, Bias: -17.132772, T: 336, Avg. loss: 2.765791
Total training time: 0.00 seconds.
-- Epoch 4
Norm: 29.61, NNZs: 4, Bias: -2.498514, T: 448, Avg. loss: 2.785209
Total training time: 0.00 seconds.
-- Epoch 5
Norm: 30.17, NNZs: 4, Bias: -9.222483, T: 560, Avg. loss: 1.850997
Total training time: 0.00 seconds.
-- Epoch 6
Norm: 27.93, NNZs: 4, Bias: -3.330254, T: 672, Avg. loss: 2.754469
Total training time: 0.00 seconds.
-- Epoch 7
Norm: 20.01, NNZs: 4, Bias: -9.096078, T: 784, Avg. loss: 2.282979
Total training time: 0.00 seconds.
```

```
-- Epoch 8
Norm: 21.85, NNZs: 4, Bias: -14.145746, T: 896, Avg. loss: 1.911473
Total training time: 0.00 seconds.
-- Epoch 9
Norm: 20.66, NNZs: 4, Bias: -3.779083, T: 1008, Avg. loss: 2.246304
Total training time: 0.00 seconds.
-- Epoch 10
Norm: 27.63, NNZs: 4, Bias: -3.860687, T: 1120, Avg. loss: 1.882126
Total training time: 0.00 seconds.
Convergence after 10 epochs took 0.00 seconds
-- Epoch 1
Norm: 35.14, NNZs: 4, Bias: -9.832273, T: 112, Avg. loss: 0.989802
Total training time: 0.00 seconds.
-- Epoch 2
Norm: 37.77, NNZs: 4, Bias: -10.339661, T: 224, Avg. loss: 0.136237
Total training time: 0.00 seconds.
-- Epoch 3
Norm: 33.23, NNZs: 4, Bias: -25.991898, T: 336, Avg. loss: 0.338796
Total training time: 0.00 seconds.
-- Epoch 4
Norm: 37.40, NNZs: 4, Bias: -18.578286, T: 448, Avg. loss: 0.285007
Total training time: 0.00 seconds.
-- Epoch 5
Norm: 39.28, NNZs: 4, Bias: -12.135068, T: 560, Avg. loss: 0.162513
Total training time: 0.00 seconds.
-- Epoch 6
Norm: 40.79, NNZs: 4, Bias: -18.620477, T: 672, Avg. loss: 0.204594
Total training time: 0.00 seconds.
-- Epoch 7
Norm: 40.98, NNZs: 4, Bias: -18.643216, T: 784, Avg. loss: 0.125246
Total training time: 0.00 seconds.
-- Epoch 8
Norm: 40.24, NNZs: 4, Bias: -18.660894, T: 896, Avg. loss: 0.098929
Total training time: 0.00 seconds.
-- Epoch 9
Norm: 39.83, NNZs: 4, Bias: -18.727414, T: 1008, Avg. loss: 0.122693
Total training time: 0.00 seconds.
-- Epoch 10
Norm: 38.51, NNZs: 4, Bias: -18.783379, T: 1120, Avg. loss: 0.110739
Total training time: 0.00 seconds.
-- Epoch 11
Norm: 40.25, NNZs: 4, Bias: -14.381678, T: 1232, Avg. loss: 0.079530
Total training time: 0.00 seconds.
-- Epoch 12
Norm: 37.20, NNZs: 4, Bias: -18.780000, T: 1344, Avg. loss: 0.138565
```

Total training time: 0.00 seconds.
-- Epoch 13
Norm: 36.65, NNZs: 4, Bias: -18.737272, T: 1456, Avg. loss: 0.129684
Total training time: 0.00 seconds.
-- Epoch 14
Norm: 37.52, NNZs: 4, Bias: -14.781564, T: 1568, Avg. loss: 0.110343
Total training time: 0.00 seconds.
-- Epoch 15
Norm: 37.37, NNZs: 4, Bias: -14.798968, T: 1680, Avg. loss: 0.148323
Total training time: 0.01 seconds.
-- Epoch 16
Norm: 36.26, NNZs: 4, Bias: -14.830780, T: 1792, Avg. loss: 0.099459
Total training time: 0.01 seconds.
Convergence after 16 epochs took 0.01 seconds
</pre></details>
 Done in 0.008081912994384766 seconds.

Testing...
 Done.

Accuracy
 0.8421052631578947

Classification report
```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        13
Iris-versicolor       0.71      0.92      0.80        13
 Iris-virginica       0.88      0.58      0.70        12

       accuracy                           0.84        38
      macro avg       0.86      0.84      0.83        38
   weighted avg       0.86      0.84      0.84        38

```

Gaussian Naive Bayes
Creating GaussianNB...
 Done.

Parameters
```
       priors = None
var_smoothing = 1e-09
```

```

#### Training...
 Done in 0.0 seconds.

#### Testing...
 Done.

#### Accuracy
 0.9210526315789473

#### Classification report
```
                 precision   recall f1-score   support

    Iris-setosa   1.00   1.00   1.00   13
Iris-versicolor   0.92   0.85   0.88   13
 Iris-virginica   0.85   0.92   0.88   12

       accuracy   0.92   38
      macro avg   0.92   0.92   0.92   38
   weighted avg   0.92   0.92   0.92   38

```
```

4. Plik wynikowy 17.md

````
# Dataset 17
# Dataset 850
## Checking if file exists
 The file "850.pkl" exists.

## Loading dataset from file...
 Done.

## Abstract
Images of the Kecimen and Besni raisin varieties were obtained with CVS. A total of 900 raisins
were used, including 450 from both varieties, and 7 morphological features were extracted.

## Summary
Images of Kecimen and Besni raisin varieties grown in Turkey were obtained with CVS. A total of
900 raisin grains were used, including 450 pieces from both varieties. These images were subjected
to various stages of pre-processing and 7 morphological features were extracted. These features
have been classified using three different artificial intelligence techniques.

## Data URL
https://archive.ics.uci.edu/static/public/850/data.csv

## Variables
```
              name     role  ... units missing_values
0             Area  Feature  ...  None             no
1  MajorAxisLength  Feature  ...  None             no
2  MinorAxisLength  Feature  ...  None             no
3     Eccentricity  Feature  ...  None             no
4       ConvexArea  Feature  ...  None             no
5           Extent  Feature  ...  None             no
6        Perimeter  Feature  ...  None             no
7            Class   Target  ...  None             no

[8 rows x 7 columns]
```

## Preprocessing
### Scaling each column to fit [-1;1]...
 Done.

### Splitting data into training and testing sets...
 Done.
````

Results
Neural network
Creating MLPClassifier...
 Done.

Parameters
```
 hidden_layer_sizes = (10, 10, 10)
         activation = relu
             solver = adam
              alpha = 0.0001
         batch_size = auto
      learning_rate = constant
 learning_rate_init = 0.001
            power_t = 0.5
           max_iter = 1000
            shuffle = True
       random_state = None
                tol = 0.0001
            verbose = True
         warm_start = False
           momentum = 0.9
 nesterovs_momentum = True
     early_stopping = False
validation_fraction = 0.1
             beta_1 = 0.9
             beta_2 = 0.999
            epsilon = 1e-08
   n_iter_no_change = 10
            max_fun = 15000
```

Training...
<details><pre>
Iteration 1, loss = 0.90378124
Iteration 2, loss = 0.87975238
Iteration 3, loss = 0.85874835
Iteration 4, loss = 0.83943898
Iteration 5, loss = 0.82314118
Iteration 6, loss = 0.80868560
Iteration 7, loss = 0.79551960
Iteration 8, loss = 0.78460342
Iteration 9, loss = 0.77419154
Iteration 10, loss = 0.76496256

```
Iteration 11, loss = 0.75646956
Iteration 12, loss = 0.74817388
Iteration 13, loss = 0.74056782
Iteration 14, loss = 0.73356281
Iteration 15, loss = 0.72635822
Iteration 16, loss = 0.71936356
Iteration 17, loss = 0.71234826
Iteration 18, loss = 0.70565445
Iteration 19, loss = 0.69885324
Iteration 20, loss = 0.69199400
Iteration 21, loss = 0.68504016
Iteration 22, loss = 0.67831601
Iteration 23, loss = 0.67125769
Iteration 24, loss = 0.66462073
Iteration 25, loss = 0.65770595
Iteration 26, loss = 0.65074495
Iteration 27, loss = 0.64366720
Iteration 28, loss = 0.63614735
Iteration 29, loss = 0.62839502
Iteration 30, loss = 0.62007774
Iteration 31, loss = 0.61060938
Iteration 32, loss = 0.60054958
Iteration 33, loss = 0.58986581
Iteration 34, loss = 0.57887364
Iteration 35, loss = 0.56854385
Iteration 36, loss = 0.55765801
Iteration 37, loss = 0.54798881
Iteration 38, loss = 0.53793779
Iteration 39, loss = 0.52858597
Iteration 40, loss = 0.51905740
Iteration 41, loss = 0.51004244
Iteration 42, loss = 0.50100826
Iteration 43, loss = 0.49248531
Iteration 44, loss = 0.48397814
Iteration 45, loss = 0.47592875
Iteration 46, loss = 0.46785510
Iteration 47, loss = 0.46039972
Iteration 48, loss = 0.45268755
Iteration 49, loss = 0.44599374
Iteration 50, loss = 0.43937651
Iteration 51, loss = 0.43271601
Iteration 52, loss = 0.42665419
Iteration 53, loss = 0.42076422
Iteration 54, loss = 0.41560947
Iteration 55, loss = 0.41047651
```

```
Iteration 56, loss = 0.40586682
Iteration 57, loss = 0.40144918
Iteration 58, loss = 0.39791251
Iteration 59, loss = 0.39460793
Iteration 60, loss = 0.39089587
Iteration 61, loss = 0.38733424
Iteration 62, loss = 0.38428988
Iteration 63, loss = 0.38215924
Iteration 64, loss = 0.37940488
Iteration 65, loss = 0.37740890
Iteration 66, loss = 0.37539692
Iteration 67, loss = 0.37353127
Iteration 68, loss = 0.37202319
Iteration 69, loss = 0.37031933
Iteration 70, loss = 0.36922690
Iteration 71, loss = 0.36776325
Iteration 72, loss = 0.36646349
Iteration 73, loss = 0.36556335
Iteration 74, loss = 0.36468230
Iteration 75, loss = 0.36345637
Iteration 76, loss = 0.36262641
Iteration 77, loss = 0.36193595
Iteration 78, loss = 0.36128767
Iteration 79, loss = 0.36042564
Iteration 80, loss = 0.35961524
Iteration 81, loss = 0.35890548
Iteration 82, loss = 0.35836137
Iteration 83, loss = 0.35768697
Iteration 84, loss = 0.35705837
Iteration 85, loss = 0.35652766
Iteration 86, loss = 0.35598525
Iteration 87, loss = 0.35544514
Iteration 88, loss = 0.35510197
Iteration 89, loss = 0.35483006
Iteration 90, loss = 0.35415704
Iteration 91, loss = 0.35401537
Iteration 92, loss = 0.35374841
Iteration 93, loss = 0.35333632
Iteration 94, loss = 0.35256632
Iteration 95, loss = 0.35214439
Iteration 96, loss = 0.35275991
Iteration 97, loss = 0.35256575
Iteration 98, loss = 0.35130399
Iteration 99, loss = 0.35088545
Iteration 100, loss = 0.35063800
```

```
Iteration 101, loss = 0.35011403
Iteration 102, loss = 0.34983535
Iteration 103, loss = 0.34933441
Iteration 104, loss = 0.34926722
Iteration 105, loss = 0.34875159
Iteration 106, loss = 0.34853581
Iteration 107, loss = 0.34814987
Iteration 108, loss = 0.34766411
Iteration 109, loss = 0.34745802
Iteration 110, loss = 0.34700243
Iteration 111, loss = 0.34693228
Iteration 112, loss = 0.34638213
Iteration 113, loss = 0.34602688
Iteration 114, loss = 0.34568421
Iteration 115, loss = 0.34535773
Iteration 116, loss = 0.34492860
Iteration 117, loss = 0.34483684
Iteration 118, loss = 0.34468016
Iteration 119, loss = 0.34423936
Iteration 120, loss = 0.34360905
Iteration 121, loss = 0.34351779
Iteration 122, loss = 0.34302246
Iteration 123, loss = 0.34261342
Iteration 124, loss = 0.34241793
Iteration 125, loss = 0.34213402
Iteration 126, loss = 0.34175397
Iteration 127, loss = 0.34152462
Iteration 128, loss = 0.34124129
Iteration 129, loss = 0.34099853
Iteration 130, loss = 0.34075113
Iteration 131, loss = 0.34050866
Iteration 132, loss = 0.34042796
Iteration 133, loss = 0.33999848
Iteration 134, loss = 0.33967105
Iteration 135, loss = 0.33947720
Iteration 136, loss = 0.33911707
Iteration 137, loss = 0.33879204
Iteration 138, loss = 0.33860081
Iteration 139, loss = 0.33838185
Iteration 140, loss = 0.33833878
Iteration 141, loss = 0.33790697
Iteration 142, loss = 0.33752456
Iteration 143, loss = 0.33713569
Iteration 144, loss = 0.33794044
Iteration 145, loss = 0.33746017
```

```
Iteration 146, loss = 0.33678111
Iteration 147, loss = 0.33667554
Iteration 148, loss = 0.33616499
Iteration 149, loss = 0.33584337
Iteration 150, loss = 0.33550948
Iteration 151, loss = 0.33519804
Iteration 152, loss = 0.33493934
Iteration 153, loss = 0.33491773
Iteration 154, loss = 0.33488842
Iteration 155, loss = 0.33429853
Iteration 156, loss = 0.33411930
Iteration 157, loss = 0.33415921
Iteration 158, loss = 0.33416987
Iteration 159, loss = 0.33396019
Iteration 160, loss = 0.33370968
Iteration 161, loss = 0.33344659
Iteration 162, loss = 0.33315305
Iteration 163, loss = 0.33302895
Iteration 164, loss = 0.33364461
Iteration 165, loss = 0.33361802
Iteration 166, loss = 0.33283628
Iteration 167, loss = 0.33279973
Iteration 168, loss = 0.33275195
Iteration 169, loss = 0.33234659
Iteration 170, loss = 0.33222837
Iteration 171, loss = 0.33197208
Iteration 172, loss = 0.33175992
Iteration 173, loss = 0.33157356
Iteration 174, loss = 0.33211190
Iteration 175, loss = 0.33215886
Iteration 176, loss = 0.33163063
Iteration 177, loss = 0.33143726
Iteration 178, loss = 0.33169801
Iteration 179, loss = 0.33118997
Iteration 180, loss = 0.33090217
Iteration 181, loss = 0.33038710
Iteration 182, loss = 0.33006661
Iteration 183, loss = 0.32993785
Iteration 184, loss = 0.32969147
Iteration 185, loss = 0.32940932
Iteration 186, loss = 0.32949296
Iteration 187, loss = 0.32971378
Iteration 188, loss = 0.32898041
Iteration 189, loss = 0.32868901
Iteration 190, loss = 0.32841611
```

```
Iteration 191, loss = 0.32836759
Iteration 192, loss = 0.32814091
Iteration 193, loss = 0.32799997
Iteration 194, loss = 0.32791998
Iteration 195, loss = 0.32765270
Iteration 196, loss = 0.32772870
Iteration 197, loss = 0.32739274
Iteration 198, loss = 0.32703472
Iteration 199, loss = 0.32693167
Iteration 200, loss = 0.32696557
Iteration 201, loss = 0.32698681
Iteration 202, loss = 0.32683737
Iteration 203, loss = 0.32662895
Iteration 204, loss = 0.32638814
Iteration 205, loss = 0.32636113
Iteration 206, loss = 0.32624776
Iteration 207, loss = 0.32612028
Iteration 208, loss = 0.32581824
Iteration 209, loss = 0.32566913
Iteration 210, loss = 0.32613084
Iteration 211, loss = 0.32578786
Iteration 212, loss = 0.32545779
Iteration 213, loss = 0.32560994
Iteration 214, loss = 0.32521250
Iteration 215, loss = 0.32562080
Iteration 216, loss = 0.32596877
Iteration 217, loss = 0.32581896
Iteration 218, loss = 0.32485717
Iteration 219, loss = 0.32468226
Iteration 220, loss = 0.32474016
Iteration 221, loss = 0.32471816
Iteration 222, loss = 0.32449438
Iteration 223, loss = 0.32450649
Iteration 224, loss = 0.32435060
Iteration 225, loss = 0.32426126
Iteration 226, loss = 0.32416263
Iteration 227, loss = 0.32411273
Iteration 228, loss = 0.32421668
Iteration 229, loss = 0.32399771
Iteration 230, loss = 0.32371364
Iteration 231, loss = 0.32375955
Iteration 232, loss = 0.32364013
Iteration 233, loss = 0.32352189
Iteration 234, loss = 0.32361200
Iteration 235, loss = 0.32369991
```

```
Iteration 236, loss = 0.32326761
Iteration 237, loss = 0.32323114
Iteration 238, loss = 0.32334455
Iteration 239, loss = 0.32324432
Iteration 240, loss = 0.32331143
Iteration 241, loss = 0.32288033
Iteration 242, loss = 0.32289257
Iteration 243, loss = 0.32274166
Iteration 244, loss = 0.32264072
Iteration 245, loss = 0.32263632
Iteration 246, loss = 0.32248160
Iteration 247, loss = 0.32239843
Iteration 248, loss = 0.32238269
Iteration 249, loss = 0.32232221
Iteration 250, loss = 0.32231103
Iteration 251, loss = 0.32205091
Iteration 252, loss = 0.32199177
Iteration 253, loss = 0.32206492
Iteration 254, loss = 0.32196158
Iteration 255, loss = 0.32175275
Iteration 256, loss = 0.32175671
Iteration 257, loss = 0.32180410
Iteration 258, loss = 0.32185182
Iteration 259, loss = 0.32171391
Iteration 260, loss = 0.32149974
Iteration 261, loss = 0.32144834
Iteration 262, loss = 0.32158665
Iteration 263, loss = 0.32155661
Iteration 264, loss = 0.32126743
Iteration 265, loss = 0.32133630
Iteration 266, loss = 0.32098807
Iteration 267, loss = 0.32133440
Iteration 268, loss = 0.32141702
Iteration 269, loss = 0.32132810
Iteration 270, loss = 0.32100690
Iteration 271, loss = 0.32082877
Iteration 272, loss = 0.32083740
Iteration 273, loss = 0.32128710
Iteration 274, loss = 0.32137459
Iteration 275, loss = 0.32071934
Iteration 276, loss = 0.32063376
Iteration 277, loss = 0.32056027
Iteration 278, loss = 0.32059748
Iteration 279, loss = 0.32070825
Iteration 280, loss = 0.32018987
```

```
Iteration 281, loss = 0.32014224
Iteration 282, loss = 0.32006562
Iteration 283, loss = 0.32007106
Iteration 284, loss = 0.32023829
Iteration 285, loss = 0.32007685
Iteration 286, loss = 0.31966142
Iteration 287, loss = 0.32003398
Iteration 288, loss = 0.32012640
Iteration 289, loss = 0.31979748
Iteration 290, loss = 0.31948173
Iteration 291, loss = 0.31962156
Iteration 292, loss = 0.31970627
Iteration 293, loss = 0.31952013
Iteration 294, loss = 0.31928906
Iteration 295, loss = 0.31929546
Iteration 296, loss = 0.31927686
Iteration 297, loss = 0.31913370
Iteration 298, loss = 0.31921414
Iteration 299, loss = 0.31906333
Iteration 300, loss = 0.31909651
Iteration 301, loss = 0.31892700
Iteration 302, loss = 0.31892752
Iteration 303, loss = 0.31885870
Iteration 304, loss = 0.31893146
Iteration 305, loss = 0.31859979
Iteration 306, loss = 0.31888660
Iteration 307, loss = 0.31891634
Iteration 308, loss = 0.31921734
Iteration 309, loss = 0.31865618
Iteration 310, loss = 0.31860283
Iteration 311, loss = 0.31862567
Iteration 312, loss = 0.31869619
Iteration 313, loss = 0.31862234
Iteration 314, loss = 0.31829348
Iteration 315, loss = 0.31835039
Iteration 316, loss = 0.31824862
Iteration 317, loss = 0.31809318
Iteration 318, loss = 0.31830307
Iteration 319, loss = 0.31827789
Iteration 320, loss = 0.31815150
Iteration 321, loss = 0.31778962
Iteration 322, loss = 0.31828785
Iteration 323, loss = 0.31783946
Iteration 324, loss = 0.31805330
Iteration 325, loss = 0.31772796
```

```
Iteration 326, loss = 0.31777980
Iteration 327, loss = 0.31754664
Iteration 328, loss = 0.31758137
Iteration 329, loss = 0.31755604
Iteration 330, loss = 0.31740923
Iteration 331, loss = 0.31733635
Iteration 332, loss = 0.31738138
Iteration 333, loss = 0.31721017
Iteration 334, loss = 0.31714768
Iteration 335, loss = 0.31712043
Iteration 336, loss = 0.31709368
Iteration 337, loss = 0.31709232
Iteration 338, loss = 0.31697627
Iteration 339, loss = 0.31710064
Iteration 340, loss = 0.31692219
Iteration 341, loss = 0.31687599
Iteration 342, loss = 0.31677097
Iteration 343, loss = 0.31657363
Iteration 344, loss = 0.31683803
Iteration 345, loss = 0.31664533
Iteration 346, loss = 0.31696207
Iteration 347, loss = 0.31661456
Iteration 348, loss = 0.31628974
Iteration 349, loss = 0.31659406
Iteration 350, loss = 0.31671840
Iteration 351, loss = 0.31668941
Iteration 352, loss = 0.31628882
Iteration 353, loss = 0.31632274
Iteration 354, loss = 0.31607383
Iteration 355, loss = 0.31613459
Iteration 356, loss = 0.31609102
Iteration 357, loss = 0.31600799
Iteration 358, loss = 0.31603175
Iteration 359, loss = 0.31588946
Iteration 360, loss = 0.31584015
Iteration 361, loss = 0.31611729
Iteration 362, loss = 0.31583862
Iteration 363, loss = 0.31590811
Iteration 364, loss = 0.31576076
Iteration 365, loss = 0.31557206
Iteration 366, loss = 0.31585786
Iteration 367, loss = 0.31512827
Iteration 368, loss = 0.31588837
Iteration 369, loss = 0.31594590
Iteration 370, loss = 0.31561937
```

Iteration 371, loss = 0.31525742
Iteration 372, loss = 0.31524591
Iteration 373, loss = 0.31517454
Iteration 374, loss = 0.31517642
Iteration 375, loss = 0.31504830
Iteration 376, loss = 0.31538320
Iteration 377, loss = 0.31501182
Iteration 378, loss = 0.31505703
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
</pre></details>
Done in 0.8822076320648193 seconds.

Testing...
Done.

Accuracy
0.8666666666666667

Classification report
```
              precision    recall  f1-score   support

       Besni       0.91      0.81      0.86       112
     Kecimen       0.83      0.92      0.87       113

    accuracy                           0.87       225
   macro avg       0.87      0.87      0.87       225
weighted avg       0.87      0.87      0.87       225

```

Logistic regression
Creating LogisticRegression...
Done.

Parameters
```
          penalty = l2
             dual = False
              tol = 0.0001
                C = 1.0
    fit_intercept = True
intercept_scaling = 1
     class_weight = None
     random_state = None
```

          solver = newton-cholesky
        max_iter = 100
     multi_class = auto
         verbose = True
      warm_start = False
          n_jobs = None
        l1_ratio = None
```

#### Training...
&lt;details&gt;&lt;pre&gt;
Newton iter=1
  Check Convergence
    1. max |gradient| 0.024846805341717612 &lt;= 0.0001
Newton iter=2
  Check Convergence
    1. max |gradient| 0.0069768354413986215 &lt;= 0.0001
Newton iter=3
  Check Convergence
    1. max |gradient| 0.0013256195619903589 &lt;= 0.0001
Newton iter=4
  Check Convergence
    1. max |gradient| 5.120379778975927e-05 &lt;= 0.0001
    2. Newton decrement 0.00023198762610573961 &lt;= 0.0001
Newton iter=5
  Check Convergence
    1. max |gradient| 7.368604711122106e-08 &lt;= 0.0001
    2. Newton decrement 3.3564041648913514e-07 &lt;= 0.0001
  Solver did converge at loss = 0.3844188787088002.
&lt;/pre&gt;&lt;/details&gt;
 Done in 0.0 seconds.

#### Testing...
 Done.

#### Accuracy
 0.8666666666666667

#### Classification report
```
              precision   recall f1-score   support

       Besni   0.89   0.84   0.86   112
     Kecimen   0.85   0.89   0.87   113

```
    accuracy                           0.87       225
   macro avg       0.87      0.87      0.87       225
weighted avg       0.87      0.87      0.87       225

```

Decision tree classifier
Creating DecisionTreeClassifier...
 Done.

Parameters
```
               criterion = gini
                splitter = best
               max_depth = None
       min_samples_split = 2
        min_samples_leaf = 1
min_weight_fraction_leaf = 0.0
            max_features = None
            random_state = None
          max_leaf_nodes = None
   min_impurity_decrease = 0.0
            class_weight = None
               ccp_alpha = 0.0
```

Training...
 Done in 0.0 seconds.

Testing...
 Done.

Accuracy
 0.7777777777777778

Classification report
```
              precision    recall  f1-score   support

       Besni       0.79      0.76      0.77       112
     Kecimen       0.77      0.80      0.78       113

    accuracy                           0.78       225
   macro avg       0.78      0.78      0.78       225
weighted avg       0.78      0.78      0.78       225
```

```

### Linear Support Vector Classification
#### Creating LinearSVC...
 Done.

##### Parameters
```
          penalty = l2
             loss = squared_hinge
             dual = auto
              tol = 0.0001
                C = 1.0
      multi_class = ovr
    fit_intercept = True
intercept_scaling = 1
     class_weight = None
          verbose = 0
     random_state = None
         max_iter = 1000
```

#### Training...
 Done in 0.0 seconds.

#### Testing...
 Done.

#### Accuracy
 0.8622222222222222

#### Classification report
```
              precision   recall f1-score   support

       Besni   0.89   0.83   0.86   112
     Kecimen   0.84   0.89   0.87   113

    accuracy   0.86   225
   macro avg   0.86   0.86   0.86   225
weighted avg   0.86   0.86   0.86   225

```
```

Stochastic Gradient Descent Classification
Creating SGDClassifier...
 Done.

Parameters
```
               loss = hinge
            penalty = l2
              alpha = 0.0001
           l1_ratio = 0.15
      fit_intercept = True
           max_iter = 1000
                tol = 0.001
            shuffle = True
            verbose = True
            epsilon = 0.1
             n_jobs = None
       random_state = None
      learning_rate = optimal
               eta0 = 0.0
            power_t = 0.5
     early_stopping = False
validation_fraction = 0.1
   n_iter_no_change = 5
       class_weight = None
         warm_start = False
            average = False
```

Training...
<details><pre>
-- Epoch 1
Norm: 35.88, NNZs: 7, Bias: -21.376863, T: 675, Avg. loss: 2.956266
Total training time: 0.00 seconds.
-- Epoch 2
Norm: 24.55, NNZs: 7, Bias: -8.647663, T: 1350, Avg. loss: 1.966449
Total training time: 0.00 seconds.
-- Epoch 3
Norm: 17.15, NNZs: 7, Bias: -13.723817, T: 2025, Avg. loss: 1.657224
Total training time: 0.00 seconds.
-- Epoch 4
Norm: 19.84, NNZs: 7, Bias: -13.626143, T: 2700, Avg. loss: 1.286919
Total training time: 0.00 seconds.
-- Epoch 5
Norm: 17.91, NNZs: 7, Bias: -11.284641, T: 3375, Avg. loss: 1.193591

```
Total training time: 0.00 seconds.
-- Epoch 6
Norm: 16.60, NNZs: 7, Bias: -13.353912, T: 4050, Avg. loss: 1.052571
Total training time: 0.00 seconds.
-- Epoch 7
Norm: 14.32, NNZs: 7, Bias: -11.514329, T: 4725, Avg. loss: 0.911396
Total training time: 0.00 seconds.
-- Epoch 8
Norm: 15.64, NNZs: 7, Bias: -9.825672, T: 5400, Avg. loss: 0.904837
Total training time: 0.00 seconds.
-- Epoch 9
Norm: 15.21, NNZs: 7, Bias: -9.785874, T: 6075, Avg. loss: 0.809030
Total training time: 0.00 seconds.
-- Epoch 10
Norm: 14.41, NNZs: 7, Bias: -11.065931, T: 6750, Avg. loss: 0.800717
Total training time: 0.00 seconds.
-- Epoch 11
Norm: 14.45, NNZs: 7, Bias: -8.572333, T: 7425, Avg. loss: 0.755370
Total training time: 0.00 seconds.
-- Epoch 12
Norm: 13.05, NNZs: 7, Bias: -8.655329, T: 8100, Avg. loss: 0.734244
Total training time: 0.00 seconds.
-- Epoch 13
Norm: 13.73, NNZs: 7, Bias: -9.667042, T: 8775, Avg. loss: 0.711635
Total training time: 0.00 seconds.
-- Epoch 14
Norm: 13.17, NNZs: 7, Bias: -9.611083, T: 9450, Avg. loss: 0.667406
Total training time: 0.00 seconds.
-- Epoch 15
Norm: 13.77, NNZs: 7, Bias: -9.547893, T: 10125, Avg. loss: 0.685189
Total training time: 0.00 seconds.
-- Epoch 16
Norm: 12.53, NNZs: 7, Bias: -7.851012, T: 10800, Avg. loss: 0.669850
Total training time: 0.00 seconds.
-- Epoch 17
Norm: 12.61, NNZs: 7, Bias: -7.892760, T: 11475, Avg. loss: 0.592490
Total training time: 0.00 seconds.
-- Epoch 18
Norm: 12.50, NNZs: 7, Bias: -7.883053, T: 12150, Avg. loss: 0.607285
Total training time: 0.00 seconds.
-- Epoch 19
Norm: 12.04, NNZs: 7, Bias: -7.167631, T: 12825, Avg. loss: 0.590708
Total training time: 0.00 seconds.
-- Epoch 20
Norm: 12.32, NNZs: 7, Bias: -7.875460, T: 13500, Avg. loss: 0.608760
```

```
Total training time: 0.00 seconds.
-- Epoch 21
Norm: 12.11, NNZs: 7, Bias: -7.200789, T: 14175, Avg. loss: 0.558418
Total training time: 0.00 seconds.
-- Epoch 22
Norm: 11.68, NNZs: 7, Bias: -7.851582, T: 14850, Avg. loss: 0.578400
Total training time: 0.00 seconds.
-- Epoch 23
Norm: 11.94, NNZs: 7, Bias: -7.228002, T: 15525, Avg. loss: 0.577888
Total training time: 0.00 seconds.
-- Epoch 24
Norm: 12.10, NNZs: 7, Bias: -7.238481, T: 16200, Avg. loss: 0.546707
Total training time: 0.00 seconds.
-- Epoch 25
Norm: 12.16, NNZs: 7, Bias: -7.232757, T: 16875, Avg. loss: 0.537708
Total training time: 0.00 seconds.
-- Epoch 26
Norm: 10.92, NNZs: 7, Bias: -8.315031, T: 17550, Avg. loss: 0.538295
Total training time: 0.00 seconds.
-- Epoch 27
Norm: 11.61, NNZs: 7, Bias: -6.718387, T: 18225, Avg. loss: 0.559316
Total training time: 0.00 seconds.
-- Epoch 28
Norm: 11.62, NNZs: 7, Bias: -6.225227, T: 18900, Avg. loss: 0.536795
Total training time: 0.00 seconds.
-- Epoch 29
Norm: 11.10, NNZs: 7, Bias: -7.226786, T: 19575, Avg. loss: 0.501897
Total training time: 0.00 seconds.
-- Epoch 30
Norm: 11.52, NNZs: 7, Bias: -7.684815, T: 20250, Avg. loss: 0.492772
Total training time: 0.00 seconds.
-- Epoch 31
Norm: 11.23, NNZs: 7, Bias: -7.215878, T: 20925, Avg. loss: 0.509534
Total training time: 0.00 seconds.
-- Epoch 32
Norm: 11.32, NNZs: 7, Bias: -6.763224, T: 21600, Avg. loss: 0.492959
Total training time: 0.00 seconds.
-- Epoch 33
Norm: 11.40, NNZs: 7, Bias: -7.198870, T: 22275, Avg. loss: 0.483004
Total training time: 0.00 seconds.
-- Epoch 34
Norm: 11.49, NNZs: 7, Bias: -6.348013, T: 22950, Avg. loss: 0.498832
Total training time: 0.00 seconds.
-- Epoch 35
Norm: 11.37, NNZs: 7, Bias: -7.575541, T: 23625, Avg. loss: 0.485383
```

```
Total training time: 0.00 seconds.
-- Epoch 36
Norm: 11.82, NNZs: 7, Bias: -6.373269, T: 24300, Avg. loss: 0.488727
Total training time: 0.00 seconds.
-- Epoch 37
Norm: 11.17, NNZs: 7, Bias: -6.383553, T: 24975, Avg. loss: 0.476671
Total training time: 0.00 seconds.
-- Epoch 38
Norm: 11.05, NNZs: 7, Bias: -7.132599, T: 25650, Avg. loss: 0.472298
Total training time: 0.00 seconds.
-- Epoch 39
Norm: 11.34, NNZs: 7, Bias: -6.384528, T: 26325, Avg. loss: 0.495888
Total training time: 0.00 seconds.
-- Epoch 40
Norm: 11.40, NNZs: 7, Bias: -6.386781, T: 27000, Avg. loss: 0.479995
Total training time: 0.00 seconds.
-- Epoch 41
Norm: 11.29, NNZs: 7, Bias: -7.088584, T: 27675, Avg. loss: 0.463971
Total training time: 0.00 seconds.
-- Epoch 42
Norm: 11.47, NNZs: 7, Bias: -6.395493, T: 28350, Avg. loss: 0.468511
Total training time: 0.00 seconds.
-- Epoch 43
Norm: 11.39, NNZs: 7, Bias: -6.394603, T: 29025, Avg. loss: 0.449873
Total training time: 0.00 seconds.
-- Epoch 44
Norm: 11.89, NNZs: 7, Bias: -5.750365, T: 29700, Avg. loss: 0.475303
Total training time: 0.00 seconds.
-- Epoch 45
Norm: 11.25, NNZs: 7, Bias: -6.715470, T: 30375, Avg. loss: 0.482660
Total training time: 0.00 seconds.
-- Epoch 46
Norm: 11.53, NNZs: 7, Bias: -6.088180, T: 31050, Avg. loss: 0.447278
Total training time: 0.00 seconds.
-- Epoch 47
Norm: 11.83, NNZs: 7, Bias: -5.484599, T: 31725, Avg. loss: 0.443972
Total training time: 0.00 seconds.
-- Epoch 48
Norm: 11.03, NNZs: 7, Bias: -6.096730, T: 32400, Avg. loss: 0.463308
Total training time: 0.00 seconds.
-- Epoch 49
Norm: 11.00, NNZs: 7, Bias: -6.390348, T: 33075, Avg. loss: 0.455241
Total training time: 0.00 seconds.
-- Epoch 50
Norm: 11.10, NNZs: 7, Bias: -6.386219, T: 33750, Avg. loss: 0.447896
```

Total training time: 0.00 seconds.
-- Epoch 51
Norm: 11.03, NNZs: 7, Bias: -6.383430, T: 34425, Avg. loss: 0.462202
Total training time: 0.00 seconds.
-- Epoch 52
Norm: 11.03, NNZs: 7, Bias: -6.102037, T: 35100, Avg. loss: 0.454543
Total training time: 0.00 seconds.
Convergence after 52 epochs took 0.00 seconds
</pre></details>
 Done in 0.0 seconds.

Testing...
 Done.

Accuracy
 0.8622222222222222

Classification report
```
              precision    recall  f1-score   support

       Besni       0.91      0.80      0.85       112
     Kecimen       0.83      0.92      0.87       113

    accuracy                           0.86       225
   macro avg       0.87      0.86      0.86       225
weighted avg       0.87      0.86      0.86       225

```

Gaussian Naive Bayes
Creating GaussianNB...
 Done.

Parameters
```
       priors = None
var_smoothing = 1e-09
```

Training...
 Done in 0.0 seconds.

Testing...
 Done.

Accuracy
 0.8711111111111111

Classification report
```
              precision    recall  f1-score   support

       Besni       0.95      0.79      0.86       112
     Kecimen       0.82      0.96      0.88       113

    accuracy                           0.87       225
   macro avg       0.88      0.87      0.87       225
weighted avg       0.88      0.87      0.87       225

```