
	Politechnika Bydgoska im. J. J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki <b>Zakład Systemów Teleinformatycznych</b>		
<b>Przedmiot</b>	Wstęp do uczenia maszynowego		
<b>Prowadzący</b>	mgr inż. Martyna Tarczewska		
<b>Temat</b>	Rozpoznawanie owoców i warzyw		
<b>Studenci (+ nr indeksów)</b>	Łukasz Rydz 118849 Bartosz Sowiński 118862		
<b>Ocena</b>		<b>Data oddania spr.</b>	17.01.2024

## Contents

1. Zbiór danych: .....	2
1. O zbiorze danych: .....	2
2. Zawartość: .....	2
3. Właściwości: .....	2
2. Struktura plików: .....	2
3. Kod: .....	3
1. Ładowanie danych z plików: .....	3
2. Trenowanie modelu knn: .....	4
3. Trenowanie modelu drzewa decyzyjnego: .....	5
4. Wykresy: .....	6
a. Drzewo decyzyjne: .....	6
b. KNN: .....	7
5. Wyniki: .....	8
a. Drzewo decyzyjne: .....	8
b. KNN: .....	8
6. Co można ulepszyć: .....	9
1. Augmentacja danych: .....	9
2. Ekstrakcja cech: .....	9
3. Walidacja krzyżowa: .....	9
7. Wnioski: .....	9

## 1. Zbiór danych:

### 1. O zbiorze danych:

Zbiór danych Fruits-360 to kompleksowa kolekcja obrazów przedstawiających różnorodne owoce i warzywa. Wersja z 2020.05.18.0 obejmuje obszerny zestaw 131 różnych klas, każda reprezentująca unikalny owoc lub warzywo. Zbiór jest przeznaczony do różnych zastosowań w dziedzinie uczenia maszynowego i widzenia komputerowego, zwłaszcza w klasyfikacji obrazów.

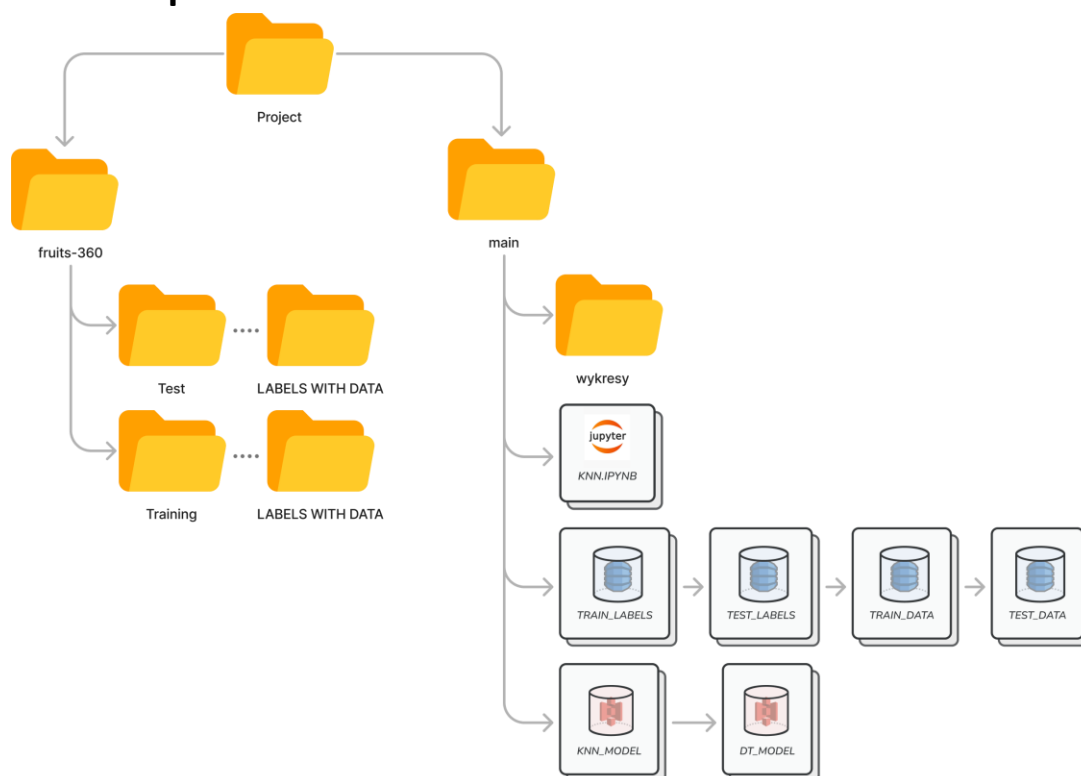
### 2. Zawartość:

Zestaw zawiera obrazy owoców i warzyw, takich jak jabłka, banany, awokado, jagody, owoce cytrusowe i wiele innych. Różne odmiany tego samego owocu są traktowane jako odrębne klasy, co zwiększa szczegółowość zbioru.

### 3. Właściwości:

- łączna liczba obrazów: 90,483 Rozmiar zbioru treningowego: 67,692 obrazów (każdy obraz zawiera pojedynczy owoc lub warzywo)
- Rozmiar zbioru testowego: 22,688 obrazów (każdy obraz zawiera pojedynczy owoc lub warzywo)
- Liczba klas: 131 (reprezentujących owoce i warzywa)
- Rozmiar obrazu: 100x100 pikseli
- Format nazwy pliku: obraz\_index\_100.jpg (lub warianty jak r\_obraz\_index\_100.jpg, r2\_obraz\_index\_100.jpg, r3\_obraz\_index\_100.jpg) "r" oznacza obrócony owoc; "r2" oznacza obrót wokół trzeciej osi; "100" odnosi się do rozmiaru obrazu.

## 2. Struktura plików:



Rysunek 1 - Struktura plików.

### 3. Kod:

#### 1. Ładowanie danych z plików:

```
1 # Funkcja usuwająca numerację z nazwy klasy
2 def remove_number(label):
3     return ''.join([i for i in label if not i.isdigit()])
4
5 def load_data_from_path(path, save_data_path, save_labels_path):
6     """
7     =====
8     Algorytm wczytujący dane z folderu i zapisujący je do plików .npz
9     Funkcja przyjmuje 3 argumenty:
10    path - ścieżka do folderu z danymi
11    save_data_path - ścieżka do pliku .npz z danymi
12    save_labels_path - ścieżka do pliku .npz z etykietami
13    =====
14    """
15    if os.path.exists(save_data_path) and os.path.exists(save_labels_path):
16        # Odczytywanie danych z plików .npz
17        data = np.load(save_data_path)
18        labels = np.load(save_labels_path)
19    else:
20        data = []
21        labels = []
22        total_files = sum([len(files) for _, _, files in os.walk(path)])
23        progress_bar = tqdm(total=total_files, unit="img", desc="Global Progress")
24
25        # Iteracja po folderach klas i załadunek obrazów
26        for class_name in os.listdir(path):
27            class_path = os.path.join(path, class_name)
28            progress_bar.set_postfix_str(f"Current Class: {class_name}")
29            for img_name in os.listdir(class_path):
30                img_path = os.path.join(class_path, img_name)
31
32                img = cv2.imread(img_path, cv2.IMREAD_COLOR)
33                img = cv2.resize(img, (100, 100))
34                img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
35                img_array = img.flatten()
36
37                # Usuwanie numeracji z nazw klas
38                base_label = remove_number(class_name)
39
40                data.append(img_array)
41                labels.append(base_label)
42
43                progress_bar.update(1)
44
45        progress_bar.close()
46
47        # Zapisywanie plików
48        np.save(save_data_path, data)
49        np.save(save_labels_path, labels)
50
51    return data, labels
52
53
54 # Dane treningowe
55 train_data, train_labels = load_data_from_path(train_path, saved_train_data_path, saved_train_labels_path)
56
57 # Dane testowe
58 test_data, test_labels = load_data_from_path(test_path, saved_test_data_path, saved_test_labels_path)
```

Rysunek 2 - Kod wczytujący dane.

Funkcja `remove_number(label)` odpowiada za usuwanie numeracji etykiet (nazw folderów) które się powtarzają. Na przykład zbiór danych zawiera 3 klasy Apple Golden które przedstawiają 3 inne jabłka tego samego gatunku. Na potrzeby projektu takie klasy łączone są w jedną klasę.

Funkcja `load_data_from_path` odpowiada za wczytywanie danych, konwertowanie ich w odpowiedni sposób oraz zapisuje je do podanych lokalizacji. Funkcja przyjmuje 3 parametry:

- `path` – ścieżka do plików
- `save_data_path` – ścieżka do zapisu danych
- `save_labels_path` – ścieżka do zapisu etykiet

Funkcja zwraca:

- `data` – listę przetworzonych danych
- `labels` – listę pobranych etykiet.

## 2. Trenowanie modelu knn:

```
# Sprawdzenie czy plik modelu istnieje
if not os.path.exists(knn_model_path):
    print("Trwa podział danych na zbiór treningowy i testowy...")
    X_train, X_test, y_train, y_test = train_test_split(train_data,
                                                         train_labels,
                                                         test_size=0.2,
                                                         random_state=32)

    # Inicjalizacja i trenowanie modelu k-NN
    knn = KNeighborsClassifier(n_neighbors=3)

    print("Trwa trenowanie modelu k-NN...")
    knn.fit(X_train, y_train)

    print("Trwa predykcja na zbiorze testowym...")
    y_pred = knn.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    print("Dokładność modelu k-NN: {:.2%}".format(accuracy))

    print("Testowanie na danych testowych...")
    test_accuracy = accuracy_score(test_labels, knn.predict(test_data))

    print("Dokładność modelu k-NN na danych testowych: {:.2%}".format(test_accuracy))

    # Zapisanie modelu do pliku
    joblib.dump(knn, knn_model_path)
else:
    # Wczytanie istniejącego modelu z pliku
    knn = joblib.load(knn_model_path)
    print("Wczytano istniejący model k-NN z pliku.")
```

```
Trwa podział danych na zbiór treningowy i testowy...
Trwa trenowanie modelu k-NN...
Trwa predykcja na zbiorze testowym...
Dokładność modelu k-NN: 99.96%
Testowanie na danych testowych...
Dokładność modelu k-NN na danych testowych: 91.25%
```

Rysunek 3 - Trenowanie modelu knn.

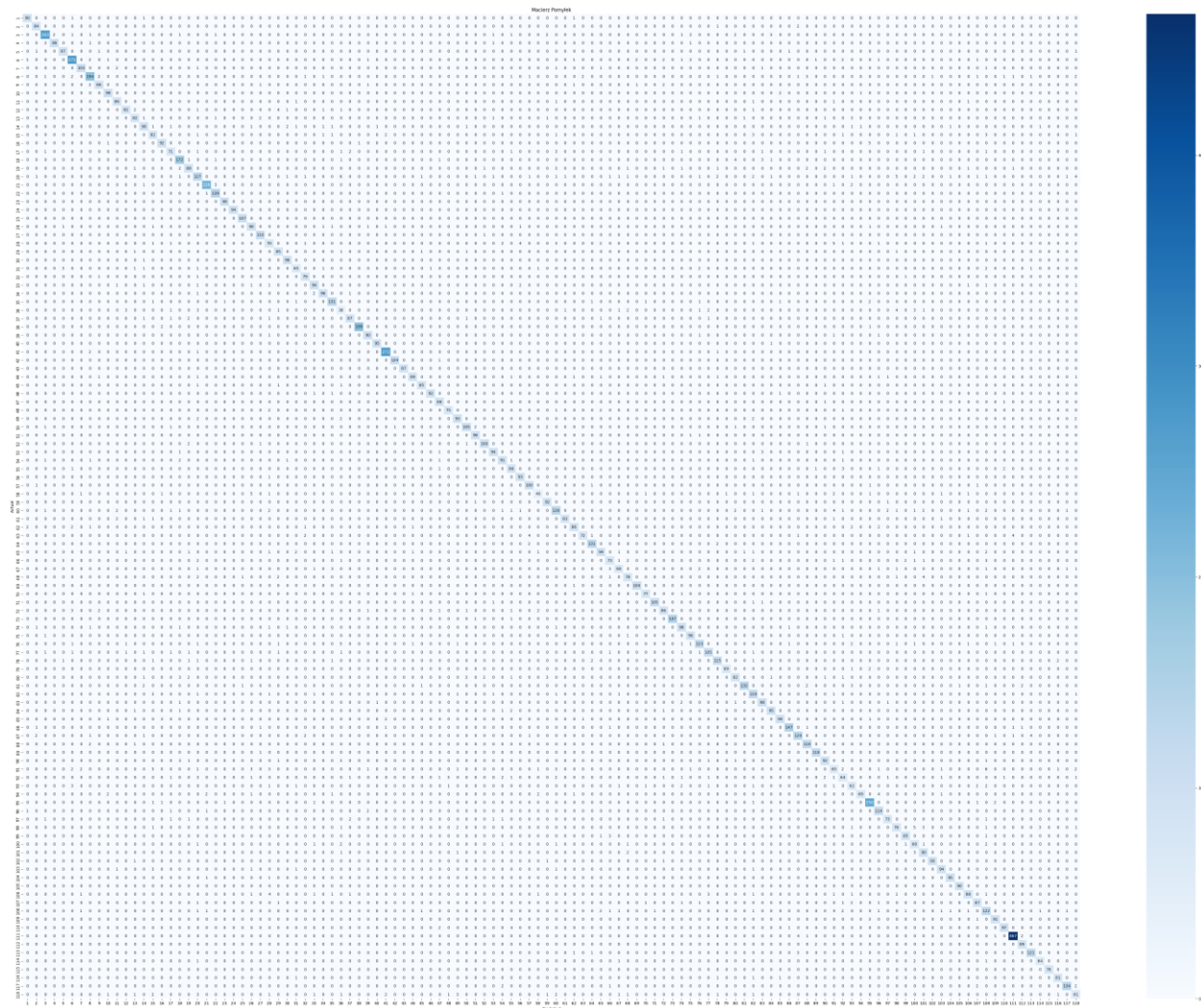
### 3. Trenowanie modelu drzewa decyzyjnego:

```
1 from sklearn.tree import DecisionTreeClassifier
2 from sklearn.metrics import accuracy_score
3 from sklearn.model_selection import train_test_split
4 import joblib
5 import os
6 import numpy as np
7
8 # Konwersja na numpy array
9 train_data = np.array(train_data)
10 train_labels = np.array(train_labels)
11 test_data = np.array(test_data)
12 test_labels = np.array(test_labels)
13
14 # Podział danych na zbiór treningowy i testowy
15 print("Podział danych na zbiór treningowy i testowy")
16 X_train, X_test, y_train, y_test = train_test_split(train_data,
17                                                    train_labels,
18                                                    test_size=0.2,
19                                                    random_state=64)
20
21 model_path = "decision_tree_model.joblib"
22
23 # Sprawdź, czy plik z modelem istnieje
24 if os.path.exists(model_path):
25     # Wczytaj istniejący model
26     tree_clf = joblib.load(model_path)
27     print("Wczytano istniejący model drzewa decyzyjnego.")
28 else:
29     # Inicjalizacja i trenowanie modelu drzewa decyzyjnego
30     tree_clf = DecisionTreeClassifier(random_state=42)
31
32     print("Trenowanie modelu drzewa decyzyjnego...")
33     tree_clf.fit(X_train, y_train)
34
35     # Zapisz model do pliku
36     joblib.dump(tree_clf, model_path)
37     print("Model drzewa decyzyjnego został zapisany do pliku.")
38
39 # Predykcja na zbiorze testowym
40 print("Predykcja na zbiorze testowym...")
41 y_pred_tree = tree_clf.predict(X_test)
42
43 # Dokładność modelu
44 accuracy_tree = accuracy_score(y_test, y_pred_tree)
45 print("Dokładność modelu drzewa decyzyjnego: {:.2%}".format(accuracy_tree))
46
47 # Testowanie na danych testowych
48 print("Testowanie na danych testowych...")
49 test_accuracy_tree = accuracy_score(test_labels, tree_clf.predict(test_data))
50 print("Dokładność modelu drzewa decyzyjnego na danych testowych: {:.2%}".format(test_accuracy_tree))
51
```

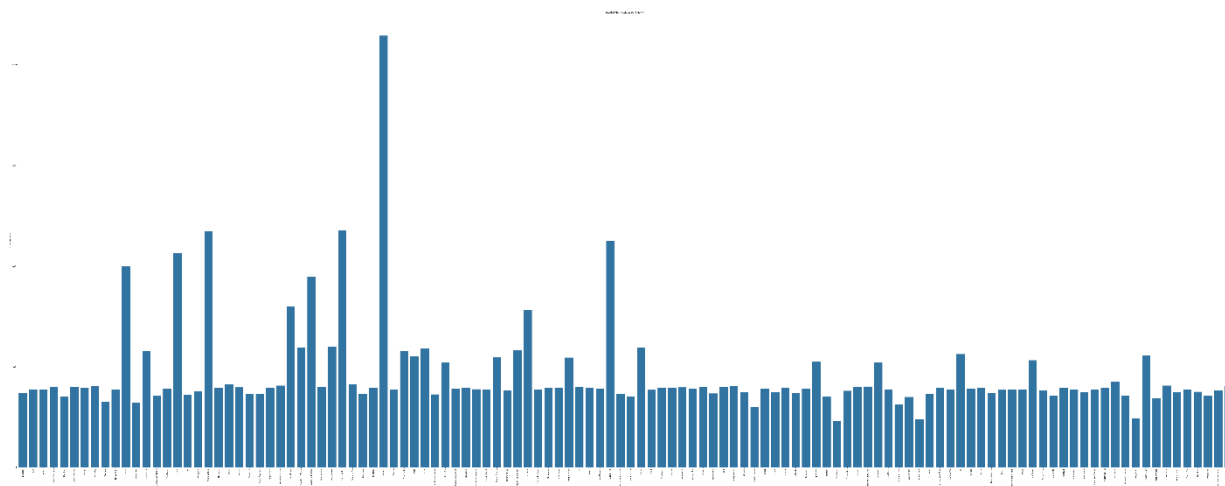
Rysunek 4 - Trenowanie drzewa decyzyjnego.

## 4. Wykresy:

### a. Drzewo decyzyjne:

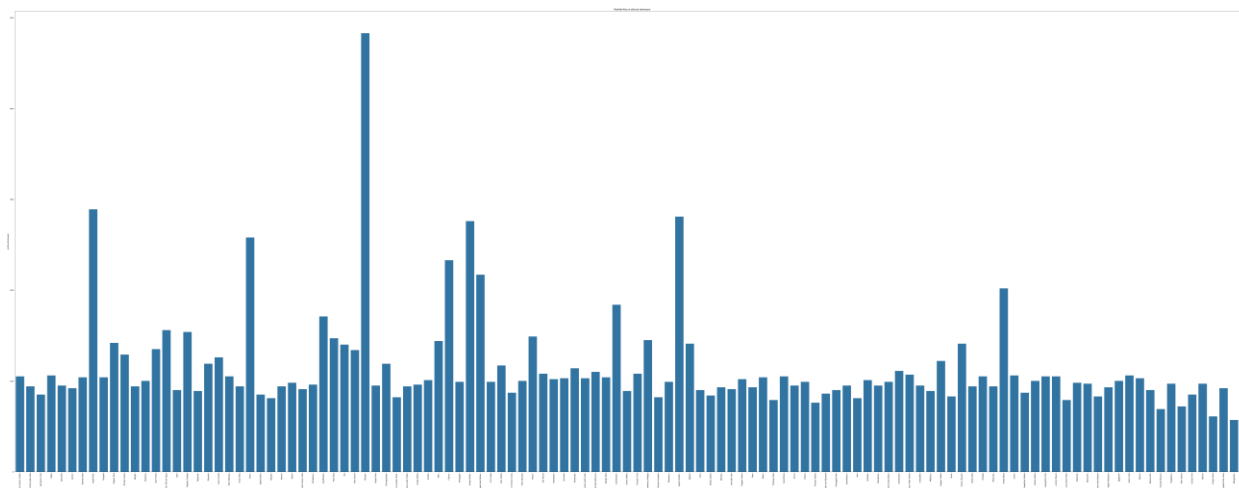


Rysunek 5 - Macierz pomyłek dla drzewa decyzyjnego.

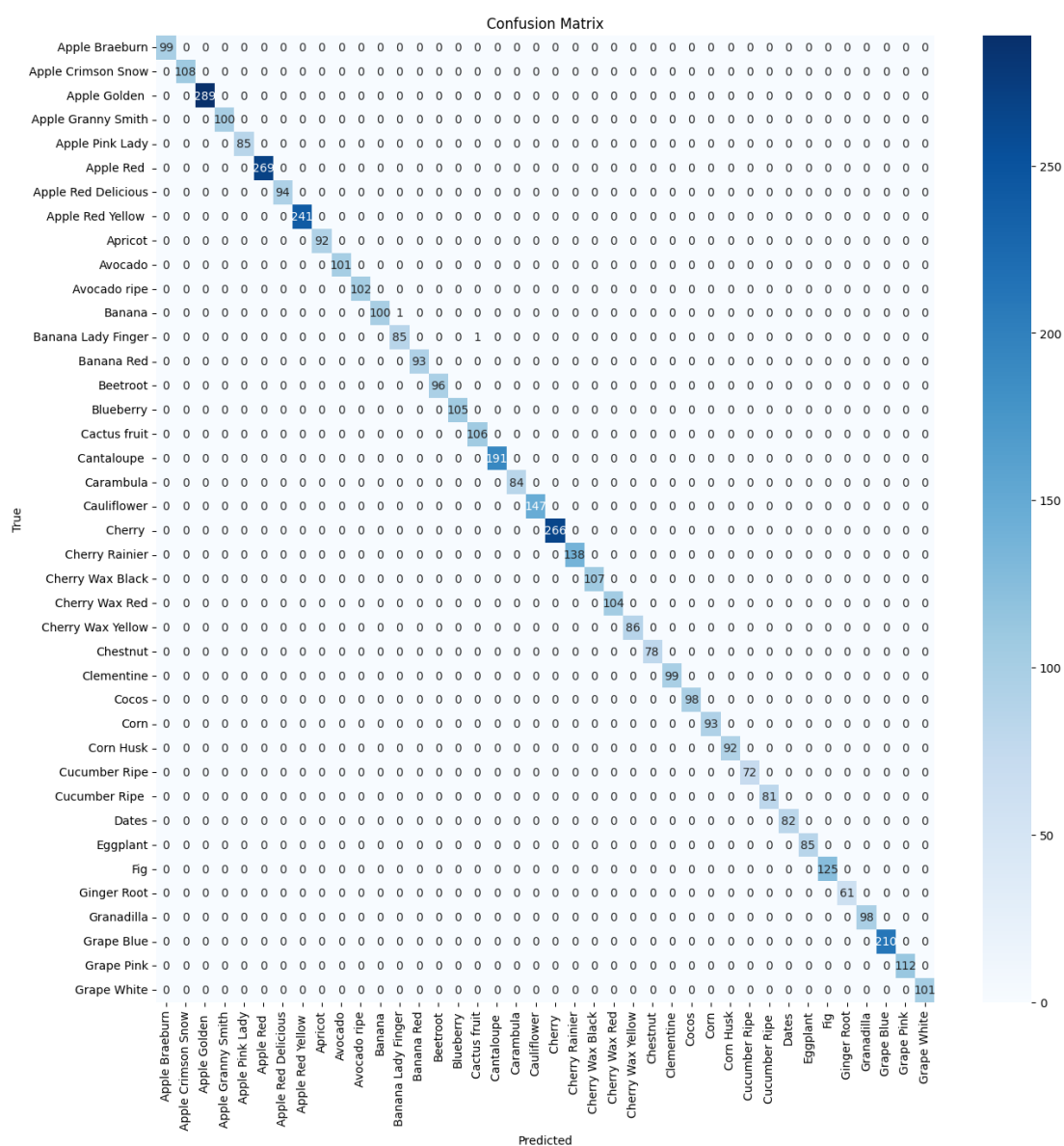


Rysunek 6 - Wykres rozkładu klas w zbiorze treningowym.

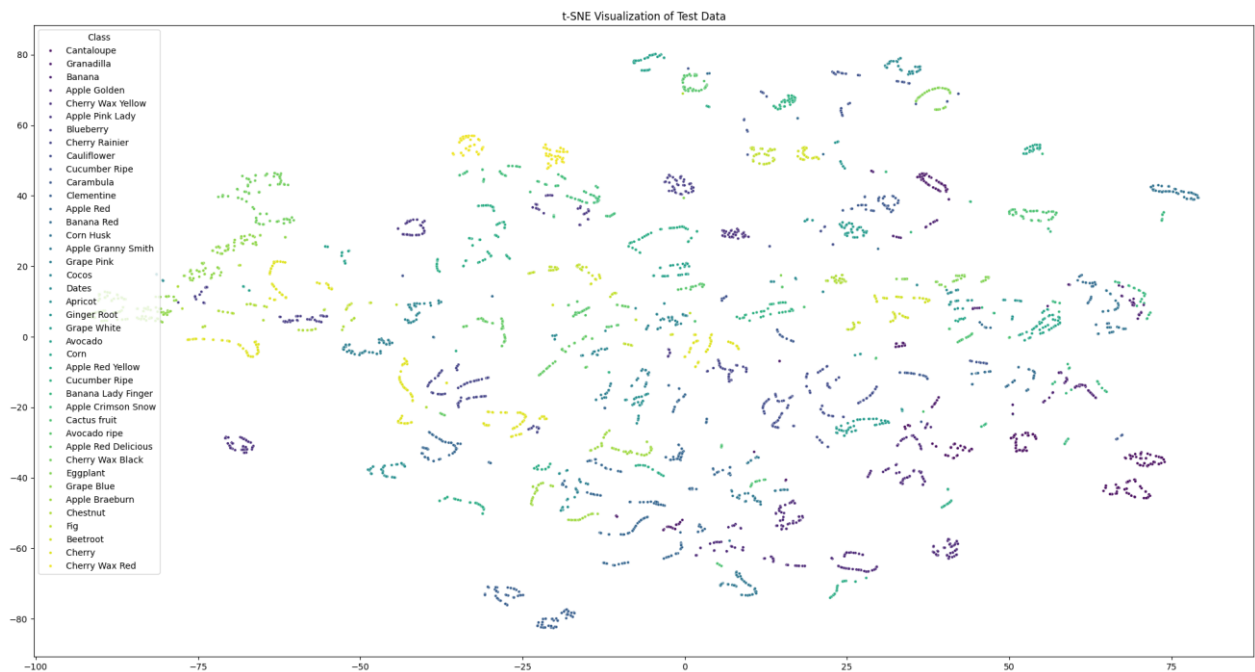




**b. KNN:**



Rysunek 8 - Macierz pomyłek.



Rysunek 9 - Wizualizacja danych w przestrzeni 2D.

## 5. Wyniki:

### a. Drzewo decyzyjne:

```
Podział danych na zbiór treningowy i testowy
Wczytano istniejący model drzewa decyzyjnego.
Predykcja na zbiorze testowym...
Dokładność modelu drzewa decyzyjnego: 91.86%
Testowanie na danych testowych...
Dokładność modelu drzewa decyzyjnego na danych testowych: 71.79%
```

Rysunek 10 - Wynik modelu drzewa decyzyjnego.

### b. KNN:

```
Podział danych na zbiór treningowy i testowy
Trenowanie modelu k-NN...
Predykcja na zbiorze testowym...
Dokładność modelu k-NN: 99.69%
Testowanie na danych testowych...
Dokładność modelu k-NN na danych testowych: 87.27%
```

Rysunek 11 - wyniki modelu KNN



## 6. Co można ulepszyć:

### 1. Augmentacja danych:

Zastosowanie technik augmentacji danych, takich jak obracanie, przesuwanie, czy zmiana skali, mogłaby zwiększyć różnorodność danych treningowych, co może poprawić generalizację modelu.

### 2. Ekstrakcja cech:

Zamiast używać surowych pikseli obrazu, można spróbować ekstrakcji cech, takich jak SIFT, HOG lub nawet głębokie cechy z wytrenowanych modeli CNN.

### 3. Walidacja krzyżowa:

Zamiast dzielić dane tylko raz, można użyć walidacji krzyżowej, aby lepiej ocenić wydajność modelu i uniknąć przeuczenia.

## 7. Wnioski:

- Język Python wraz z dostępnymi bibliotekami w bardzo dużym stopniu ułatwiają procesy modelowania i nie tylko.
- Dokładność modelu k-NN na danych testowych wynosi około 91.25% co wskazuje na ogólnie dobre dopasowanie modelu do zbioru testowego.
- Trochę gorzej poradził sobie model drzewa decyzyjnego ponieważ osiągnął wynik 71.79%.