

Załadowanie bibliotek

```
In [1]: import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline
from matplotlib import pyplot as plt

#Ustalenie stylu wykresów jako ggplot
# plt.style.use('ggplot')
from sklearn import preprocessing
from sklearn import tree
import sklearn.metrics as metrics
from sklearn.model_selection import train_test_split
import pydotplus
from IPython.display import Image
```

Wczytanie danych

```
In [2]: # Ustalenie ścieżki do datasetu
filename_mushrooms = './agaricus-lepiota.csv'

# Wczytanie datasetu jako dataframe
mushrooms_dataframe = pd.read_csv(filename_mushrooms, sep=";")

# Wyświetlenie dataframe
display(mushrooms_dataframe)
```

Zbiór danych ma 8124 wiersze i 23 kolumny (pierwsza kolumna to atrybut decyzyjny, a pozostałe 22 kolumny to atrybuty warunkowe). W celu dalszego zbadania datasetu i weryfikacji typów danych kategoriycznych w każdej kolumnie, wypisano unikalne wartości każdej kolumny. Sprawdzono również, czy zbiór danych zawiera brakujące wartości lub niepotrzebne kolumny.

```
In [3]: print("Liczba różnych wartości atrybutów wraz z ich wartościami dla każdej kolumny:")
for x in mushrooms_dataframe.columns:
    x_unique = mushrooms_dataframe[x].unique()
    print("{:>25}: {:>2} {}".format(x, x_unique.shape[0], x_unique))
```

Liczba różnych wartości atrybutów wraz z ich wartościami dla każdej kolumny:

```
classes: 2 ['p' 'e']
cap-shape: 6 ['x' 'b' 's' 'f' 'k' 'c']
cap-surface: 4 ['s' 'y' 'f' 'g']
cap-color: 10 ['n' 'y' 'w' 'g' 'e' 'p' 'b' 'u' 'c' 'r']
```

```

bruises: 2 ['t' 'f']
odor: 9 ['p' 'a' 'l' 'n' 'f' 'c' 'y' 's' 'm']
gill-attachment: 2 ['f' 'a']
gill-spacing: 2 ['c' 'w']
gill-size: 2 ['n' 'b']
gill-color: 12 ['k' 'n' 'g' 'p' 'w' 'h' 'u' 'e' 'b' 'r' 'y' 'o']
stalk-shape: 2 ['e' 't']
stalk-root: 5 ['e' 'c' 'b' 'r' '?']
stalk-surface-above-ring: 4 ['s' 'f' 'k' 'y']
stalk-surface-below-ring: 4 ['s' 'f' 'y' 'k']
stalk-color-above-ring: 9 ['w' 'g' 'p' 'n' 'b' 'e' 'o' 'c' 'y']
stalk-color-below-ring: 9 ['w' 'p' 'g' 'b' 'n' 'e' 'y' 'o' 'c']
veil-type: 1 ['p']
veil-color: 4 ['w' 'n' 'o' 'y']
ring-number: 3 ['o' 't' 'n']
ring-type: 5 ['p' 'e' 'l' 'f' 'n']
spore-print-color: 9 ['k' 'n' 'u' 'h' 'w' 'r' 'o' 'y' 'b']
population: 6 ['s' 'n' 'a' 'v' 'y' 'c']
habitat: 7 ['u' 'g' 'm' 'd' 'p' 'w' 'l']

```

Zauważono, że spośród 22 atrybutów warunkowych, jedynie ‘veil-type’ zawiera tylko jedną wartość “p”. Zatem atrybut ten nie zapewnia żadnej wartości dodanej do klasyfikatora. Podjęto decyzję o usunięciu tej kolumny - utworzono generyczny kod usuwający wszystkie kolumny zawierające jedną wartość.

```

In [4]: print("Rozmiar mushrooms_dataframe przed usunięciem: ",mushrooms_dataframe.shape)

# Usunięcie kolumn zawierających jedną wartość
for col in mushrooms_dataframe.columns.values:
    col_unique = mushrooms_dataframe[col].unique()
    if len(col_unique) == 1:
        print("Usunięto kolumnę '{}',która zawiera tylko jedną wartość: {}".format(col, col_unique[0]))
        mushrooms_dataframe = mushrooms_dataframe.drop(col, 1)

print("Rozmiar mushrooms_dataframe po usunięciu: ",mushrooms_dataframe.shape)

```

```

Rozmiar mushrooms_dataframe przed usunięciem: (8124, 23)
Usunięto kolumnę 'veil-type',która zawiera tylko jedną wartość: p
Rozmiar mushrooms_dataframe po usunięciu: (8124, 22)

```

Stwierdzono również, że kolumna ‘stalk-root’ zawiera brakujące wartości. Zbadano udział brakujących wartości w zbiorze - utworzono generyczny kod badający udziały brakujących wartości.

```

In [5]: for x in mushrooms_dataframe.columns:
        x_unique = mushrooms_dataframe[x].unique()
        if '?' in x_unique:
            column = mushrooms_dataframe[x]
            column_count = column.count()
            column_value_count = column.value_counts()

            print("Liczba obiektów w zależności od kategorii i ich udział procentowy dla klasy '{}':\n".format(x))
            stat = column_value_count.to_frame()
            stat['percent'] = 100. * column_value_count / column_count
            print(stat)

            fig = plt.figure()
            fig.patch.set_facecolor('xkcd:white')
            ax = sns.countplot(x=x, data=mushrooms_dataframe)
            ax.set_title("Liczba obiektów w zależności od kategorii dla '{}".format(x))

            #         for p in ax.patches:
            #             height = p.get_height()
            #             ax.text(p.get_x()+0.25, height+ 3, 'n=%0f'%(height))

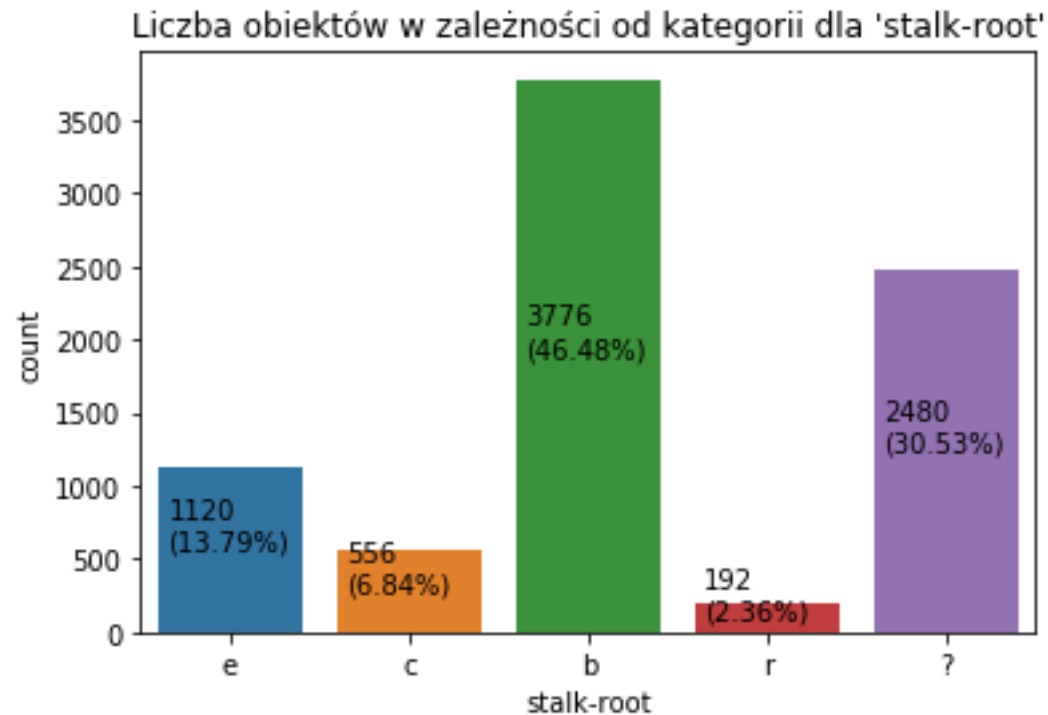
            for p in ax.patches:
                ax.annotate('{:.0f}\n({:.2f}%)'.format(p.get_height(),
                                                         100. * p.get_height() / column_count),
                            (p.get_x()+0.05, p.get_height()/2))

            plt.show()

```

Liczba obiektów w zależności od kategorii i ich udział procentowy dla klasy 'stalk-root':

	stalk-root	percent
b	3776	46.479567
?	2480	30.526834
e	1120	13.786312
c	556	6.843919
r	192	2.363368



Możliwe działania do podjęcia w przypadku występowania brakujących danych to m.in. usunięcie kolumn lub wierszy zawierających brakujące dane, wypełnienie brakujących wartości inną wartością np. z poprzedniej lub następnej komórki. Stwierdzono, że udział procentowy brakujących wartości ('?') dla atrybutu 'stalk-root' wynosi ponad 30,5%. Podjęto decyzję sporządzeniu dwóch wersji zbioru danych: z usuniętymi wierszami oraz z usuniętymi kolumnami zawierającymi brakujące wartości.

Utworzono generyczny kod oczyszczający zbiór danych z wierszy zawierających brakujące wartości:

```
In [6]: # Wykonanie kopii danych
mushrooms_dataframe_dropped_rows = mushrooms_dataframe.copy(deep=True)

# Usunięcie wierszy
for x in mushrooms_dataframe_dropped_rows.columns:
    to_delete_count = mushrooms_dataframe_dropped_rows[mushrooms_dataframe_dropped_rows[x] == '?'].shape[0]
    if to_delete_count > 0:
        mushrooms_dataframe_dropped_rows = mushrooms_dataframe_dropped_rows[mushrooms_dataframe_dropped_rows[x] != '?']
        print("W kolumnie '{}' usunięto {} wierszy zawierających brakujące wartości.".format(x, to_delete_count))
```

```

print("mushrooms_dataframe_dropped_rows: ",mushrooms_dataframe_dropped_rows.shape)

print("\n Podział atrybutu decyzyjnego:")
print(mushrooms_dataframe_dropped_rows['classes'].value_counts())

```

W kolumnie 'stalk-root' usunięto 2480 wierszy zawierających brakujące wartości.
 mushrooms_dataframe_dropped_rows: (5644, 22)

```

Podział atrybutu decyzyjnego:
e    3488
p    2156
Name: classes, dtype: int64

```

Utworzono generyczny kod oczyszczający zbiór danych z kolumn zawierających brakujące wartości powyżej zadanego progu procentowego (25%):

```

In [7]: # Próg procentowy usuwania kolumn z brakującymi wartościami
drop_percentage = 0.25

# Wykonanie kopii danych
mushrooms_dataframe_dropped_cols = mushrooms_dataframe.copy(deep=True)

# Zastąpienie znaku ? wartościami nan
for col in mushrooms_dataframe_dropped_cols:
    mushrooms_dataframe_dropped_cols.loc[mushrooms_dataframe_dropped_cols[col] == '?', col] = np.nan

# Usunięcie kolumn
for col in mushrooms_dataframe_dropped_cols.columns.values:
    no_rows = mushrooms_dataframe_dropped_cols[col].isnull().sum()
    percentage = no_rows / mushrooms_dataframe_dropped_cols.shape[0]
    if percentage >= drop_percentage:
        del mushrooms_dataframe_dropped_cols[col]
        print("Kolumna '{}' zawierająca {} brakujących wartości ({}% procent zbioru) została usunięta.".format(col, no_rows, percentage))

print("mushrooms_dataframe_dropped_cols: ",mushrooms_dataframe_dropped_cols.shape)

print("\n Podział atrybutu decyzyjnego:")
print(mushrooms_dataframe_dropped_cols['classes'].value_counts())

```

Kolumna 'stalk-root' zawierająca 2480 brakujących wartości (0.3052683407188577% procent zbioru) została usunięta.
 mushrooms_dataframe_dropped_cols: (5644, 22)

```
Podział atrybutu decyzyjnego:
e    4208
p    3916
Name: classes, dtype: int64
```

Przed przystąpieniem do budowy drzewa decyzyjnego należy zakodować wartości atrybutów (kolumn). Do zakodowania wartości kategoriycznych użyta zostanie technika kodowania etykiet, która konwertuje każdą wartość w kolumnie na liczbę.

```
In [8]: print('Kodowanie danych z usuniętymi wierszami:')
le_rows = preprocessing.LabelEncoder()
for column in mushrooms_dataframe_dropped_rows.columns:
    mushrooms_dataframe_dropped_rows[column] = le_rows.fit_transform(mushrooms_dataframe_dropped_rows[column])

print("Liczba różnych wartości atrybutów wraz z ich wartościami dla każdej kolumny po zakodowaniu:")
for x in mushrooms_dataframe_dropped_rows.columns:
    x_unique = mushrooms_dataframe_dropped_rows[x].unique()
    print("{:>25}: {:>2} {}".format(x, x_unique.shape[0], x_unique))

print('\n\nKodowanie danych z usuniętymi kolumnami:')
le_cols = preprocessing.LabelEncoder()
for column in mushrooms_dataframe_dropped_cols.columns:
    mushrooms_dataframe_dropped_cols[column] = le_cols.fit_transform(mushrooms_dataframe_dropped_cols[column])

print("Liczba różnych wartości atrybutów wraz z ich wartościami dla każdej kolumny po zakodowaniu:")
for x in mushrooms_dataframe_dropped_cols.columns:
    x_unique = mushrooms_dataframe_dropped_cols[x].unique()
    print("{:>25}: {:>2} {}".format(x, x_unique.shape[0], x_unique))
```

Kodowanie danych z usuniętymi wierszami:

Liczba różnych wartości atrybutów wraz z ich wartościami dla każdej kolumny po zakodowaniu:

```
classes: 2 [1 0]
cap-shape: 6 [5 0 4 2 3 1]
cap-surface: 4 [2 3 0 1]
cap-color: 8 [4 7 6 3 2 5 0 1]
bruises: 2 [1 0]
odor: 7 [6 0 3 5 2 1 4]
gill-attachment: 2 [1 0]
gill-spacing: 2 [0 1]
gill-size: 2 [1 0]
gill-color: 9 [2 3 0 4 7 1 6 5 8]
```

```

stalk-shape: 2 [0 1]
stalk-root: 4 [2 1 0 3]
stalk-surface-above-ring: 4 [2 0 1 3]
stalk-surface-below-ring: 4 [2 0 3 1]
stalk-color-above-ring: 7 [5 2 4 3 0 1 6]
stalk-color-below-ring: 7 [5 4 2 0 3 1 6]
veil-color: 2 [0 1]
ring-number: 3 [1 2 0]
ring-type: 4 [3 0 1 2]
spore-print-color: 6 [1 2 4 0 3 5]
population: 6 [3 2 0 4 5 1]
habitat: 6 [5 1 3 0 4 2]

```

Kodowanie danych z usuniętymi kolumnami:

Liczba różnych wartości atrybutów wraz z ich wartościami dla każdej kolumny po zakodowaniu:

```

classes: 2 [1 0]
cap-shape: 6 [5 0 4 2 3 1]
cap-surface: 4 [2 3 0 1]
cap-color: 10 [4 9 8 3 2 5 0 7 1 6]
bruises: 2 [1 0]
odor: 9 [6 0 3 5 2 1 8 7 4]
gill-attachment: 2 [1 0]
gill-spacing: 2 [0 1]
gill-size: 2 [1 0]
gill-color: 12 [4 5 2 7 10 3 9 1 0 8 11 6]
stalk-shape: 2 [0 1]
stalk-surface-above-ring: 4 [2 0 1 3]
stalk-surface-below-ring: 4 [2 0 3 1]
stalk-color-above-ring: 9 [7 3 6 4 0 2 5 1 8]
stalk-color-below-ring: 9 [7 6 3 0 4 2 8 5 1]
veil-color: 4 [2 0 1 3]
ring-number: 3 [1 2 0]
ring-type: 5 [4 0 2 1 3]
spore-print-color: 9 [2 3 6 1 7 5 4 8 0]
population: 6 [3 2 0 4 5 1]
habitat: 7 [5 1 3 0 4 6 2]

```

Mając zakodowane dane, należy dokonać ich podziału na atrybuty warunkowe (zmienna X_*) i decyzyjne (zmienna Y_*).

In [9]: `X_dropped_rows = mushrooms_dataframe_dropped_rows.drop(['classes'], axis=1)`

```

Y_dropped_rows = mushrooms_dataframe_dropped_rows['classes']

X_dropped_cols = mushrooms_dataframe_dropped_cols.drop(['classes'], axis=1)
Y_dropped_cols = mushrooms_dataframe_dropped_cols['classes']

```

Kolejnym podziałem, który należy wykonać, jest podział danych na część treningową i testową. Założono, że rozmiar części testowej będzie wynosił 33% wszystkich danych. W celu zachowania powtarzalności wyników parametr `random_state` ustawiono na wartość 30 (ustawienie innej wartości będzie powodowało wygenerowanie innego podziału danych i innego drzewa decyzyjnego).

```
In [10]: random_state = 30
```

```

X_train_dr, X_test_dr, Y_train_dr, Y_test_dr = train_test_split(X_dropped_rows, Y_dropped_rows, test_size = 0.33, random_state=random_state)

X_train_dc, X_test_dc, Y_train_dc, Y_test_dc = train_test_split(X_dropped_cols, Y_dropped_cols, test_size = 0.33, random_state=random_state)

```

Utworzono funkcję sprawdzającą jakość klasyfikacji zbudowanego drzewa, funkcję budującą drzewo oraz funkcję sprawdzającą istotność atrybutu. Funkcji sprawdzająca jakość klasyfikacji zbudowanego drzewa, wypisuje wartości `Accuracy`, czyli współczynnik dokładności modelu do danych testowych, `Precision` - procent elementów będących istotnymi oraz `Recall` - procent istotnych elementów, które zostały wybrane.

```

In [11]: def test_tree(clf, X_train, X_test, Y_train, Y_test, print_res=True):
    clf      = clf.fit(X_train, Y_train)
    score    = clf.score(X_test, Y_test)
    precision = metrics.precision_score(Y_test, clf.predict(X_test))
    recall    = metrics.recall_score(Y_test, clf.predict(X_test))
    res = (score, precision, recall)
    if print_res:
        print("Accuracy = %f / Precision = %f / Recall = %f" % res)
    return res

def build_tree(X, X_train, X_test, Y_train, Y_test, random_state, **kwargs):
    clf = tree.DecisionTreeClassifier(random_state=random_state, **kwargs)
    clf = clf.fit(X_train, Y_train)

    dot_data = tree.export_graphviz(clf, out_file=None,
                                    feature_names=X.columns,
                                    class_names=['p', 'e'],
                                    filled=True, rounded=True,
                                    special_characters=True)

    graph = pydotplus.graph_from_dot_data(dot_data)
    display(Image(graph.create_png()))

    test_tree(clf, X_train, X_test, Y_train, Y_test);

```



```

return clf

def attribute_importance(clf, X):
    attrs = X.columns.values
    attr_importance = clf.feature_importances_
    sorted_attr_importance = np.argsort(attr_importance)
    range_sorted_attr_importance = range(len(sorted_attr_importance))

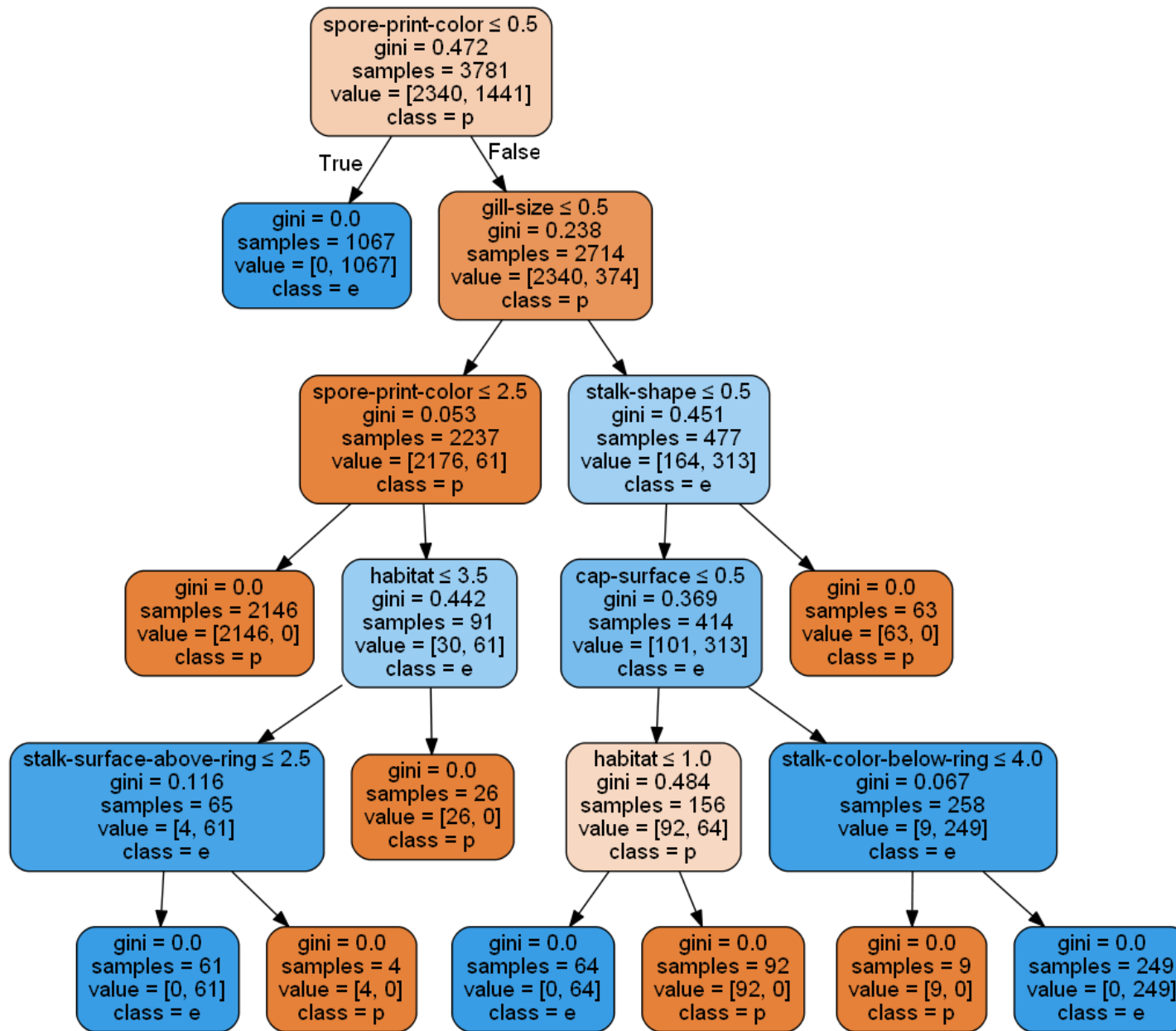
    plt.figure(figsize=(8, 7))
    plt.barh(range_sorted_attr_importance, attr_importance[sorted_attr_importance])
    plt.yticks(range_sorted_attr_importance, attrs[sorted_attr_importance])
    plt.xlabel('Importance')
    plt.title('Attribute importances')
    plt.draw()
    plt.show()

```

W oparciu o przygotowane dane zbudowano drzewa decyzyjne.

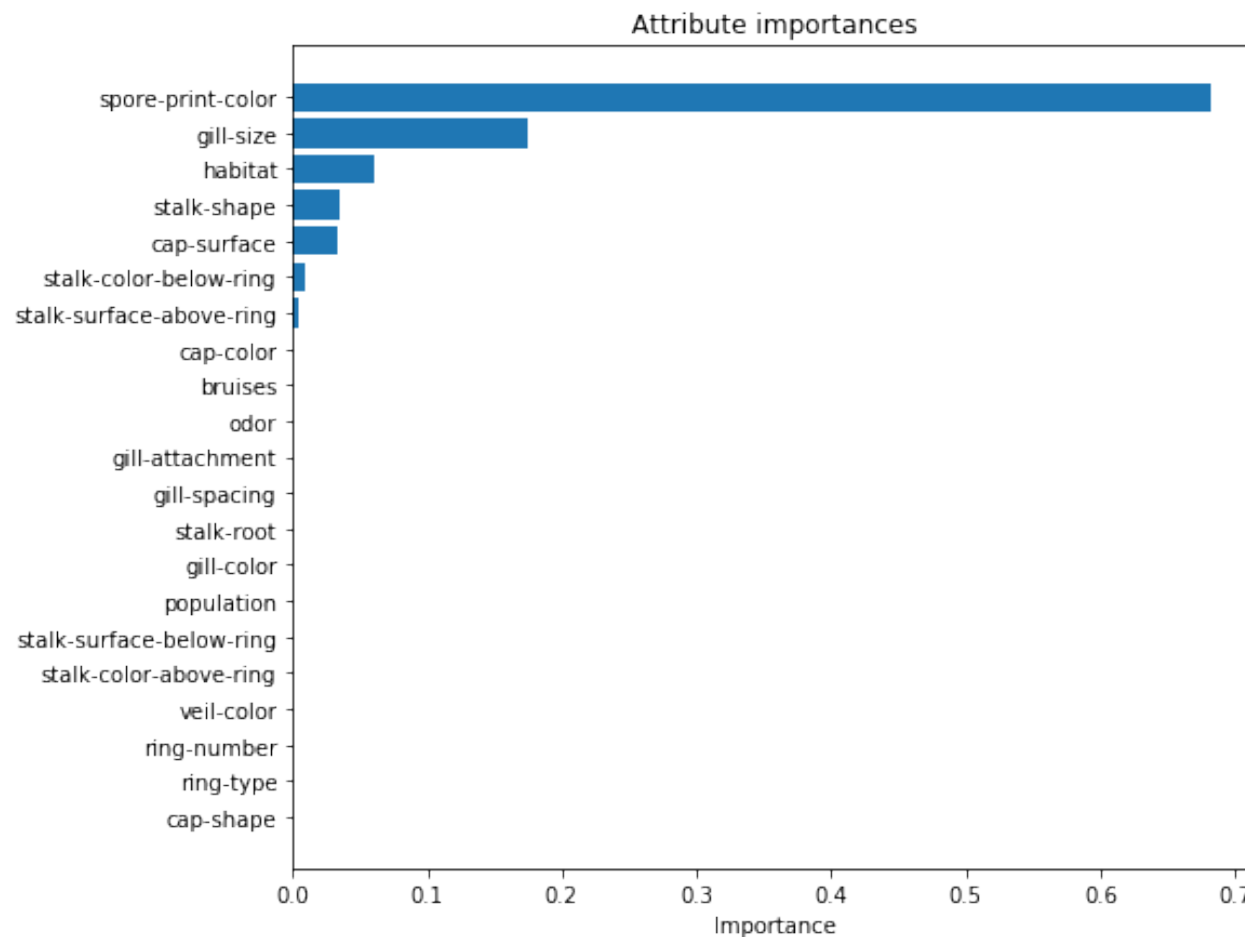
1. Dokonano klasyfikacji zbioru danych z usuniętymi wierszami, które zawierały braki danych:

```
In [12]: clf = build_tree(X_dropped_rows, X_train_dr, X_test_dr, Y_train_dr, Y_test_dr, random_state)
```



Accuracy = 1.000000 / Precision = 1.000000 / Recall = 1.000000

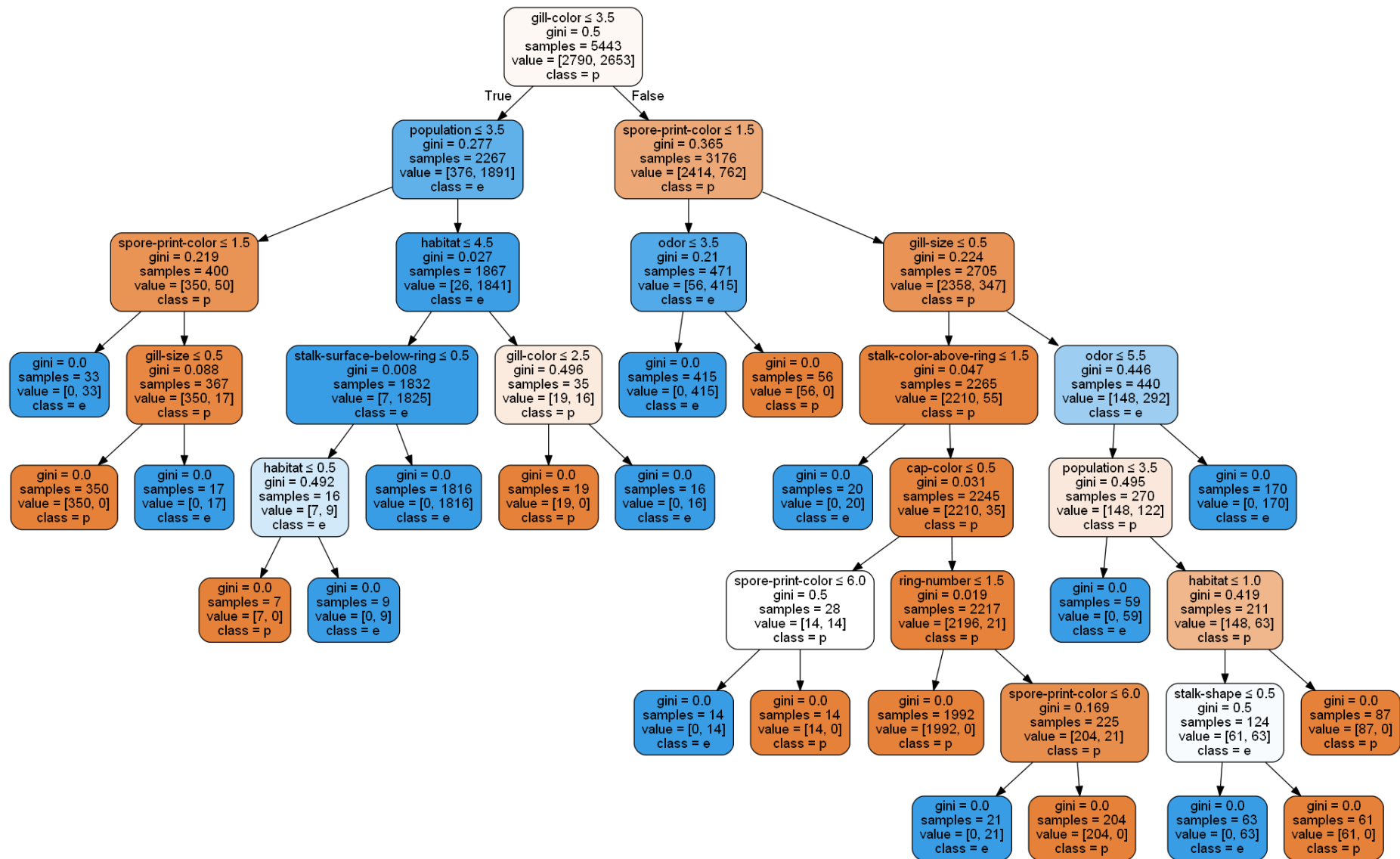
```
In [13]: attribute_importance(clf, X_dropped_rows)
```



Wygenerowane drzewo decyzyjne posiada głębokość wynoszącą 5. Ponadto zauważono, że najważniejszymi atrybutami są: spore-print-color oraz gill-size. Mniejsze znaczenie mają atrybuty: habitat, stalk-shape oraz cap-surface. Pozostałe atrybuty uznano za nieznaczące.

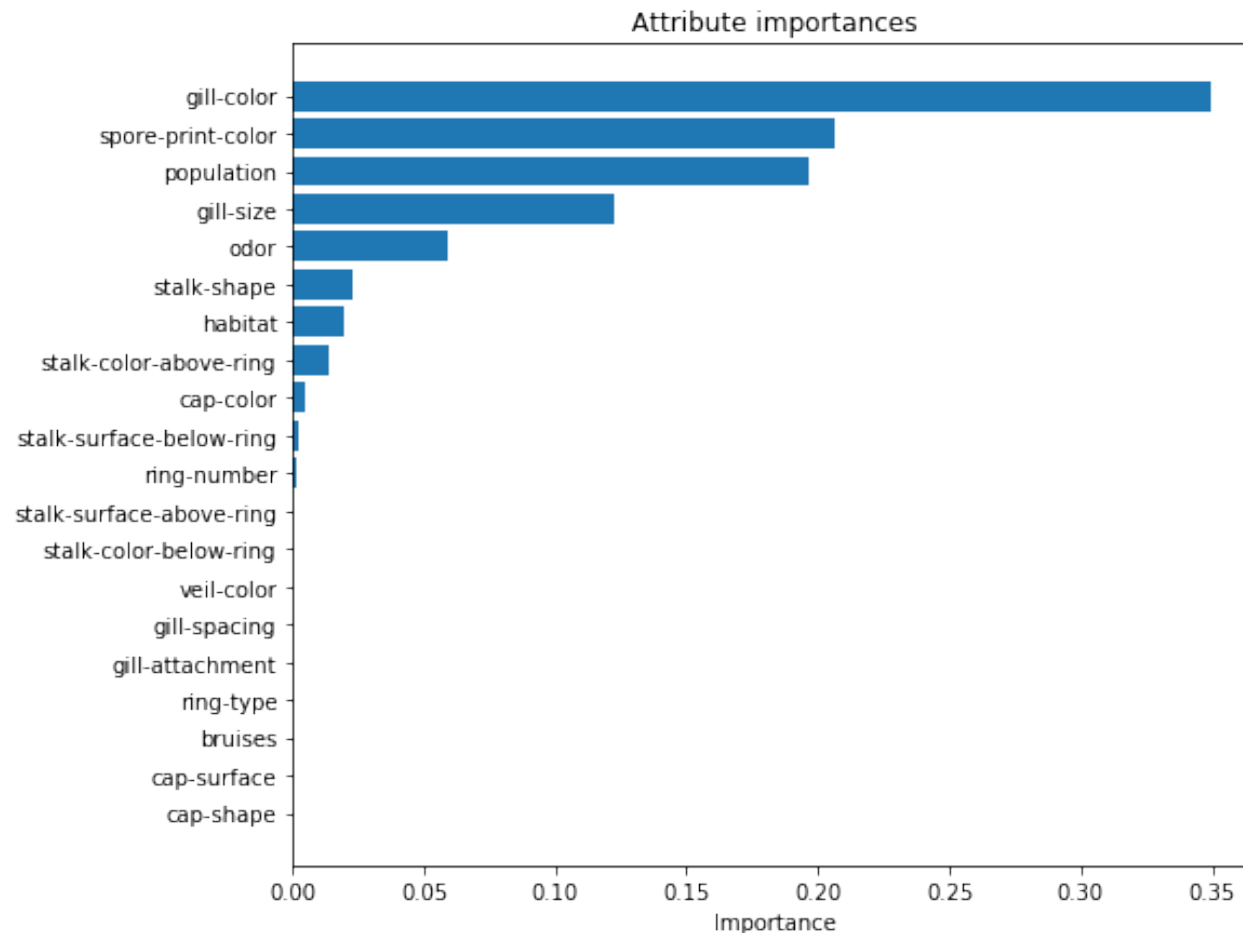
2. Dokonano klasyfikacji zbioru danych z usuniętymi kolumnami, które zawierały braki danych:

```
In [14]: clf = build_tree(X_dropped_cols, X_train_dc, X_test_dc ,Y_train_dc, Y_test_dc, random_state)
```



Accuracy = 1.000000 / Precision = 1.000000 / Recall = 1.000000

```
In [15]: attribute_importance(clf, X_dropped_cols)
```



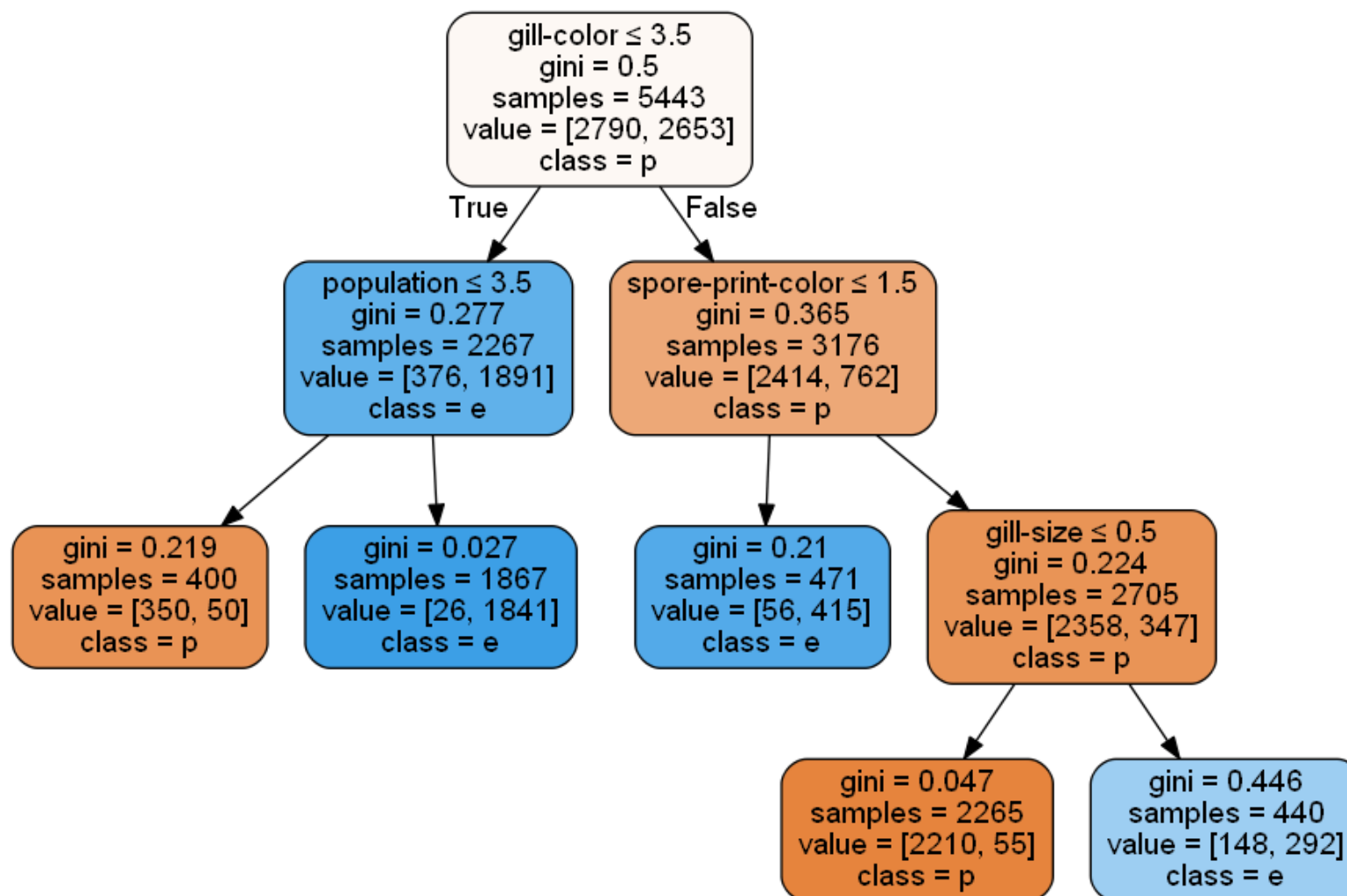
Zauważono, że drzewo wygenerowane przy użyciu danych z usuniętymi kolumnami jest bardziej rozbudowane niż drzewo wygenerowane przy użyciu danych z usuniętymi wierszami, ponieważ zostało ono skonstruowane przy użyciu zbioru o większej liczbie próbek. Wygenerowane drzewo decyzyjne posiada głębokość wynoszącą 7. Zauważono również, że najważniejszymi atrybutami są: gill-color, spore-print-color, population oraz gill-size. Mniejsze znaczenie mają atrybuty: odor, stalk-shape, habitat oraz stalk-color-above-ring. Pozostałe atrybuty uznano za nieznaczące.

Do kolejnych eksperymentów użyto z usuniętymi kolumnami, ponieważ założono że większy zbiór danych może lepiej odzwierciedlać problem klasyfikacji grzybów.

3. Dokonano klasyfikacji zbioru danych z usuniętymi kolumnami ze zmienionymi parametrami. Parametry dobierano tak, aby zmniejszyć wielkość drzewa decyzyjnego,

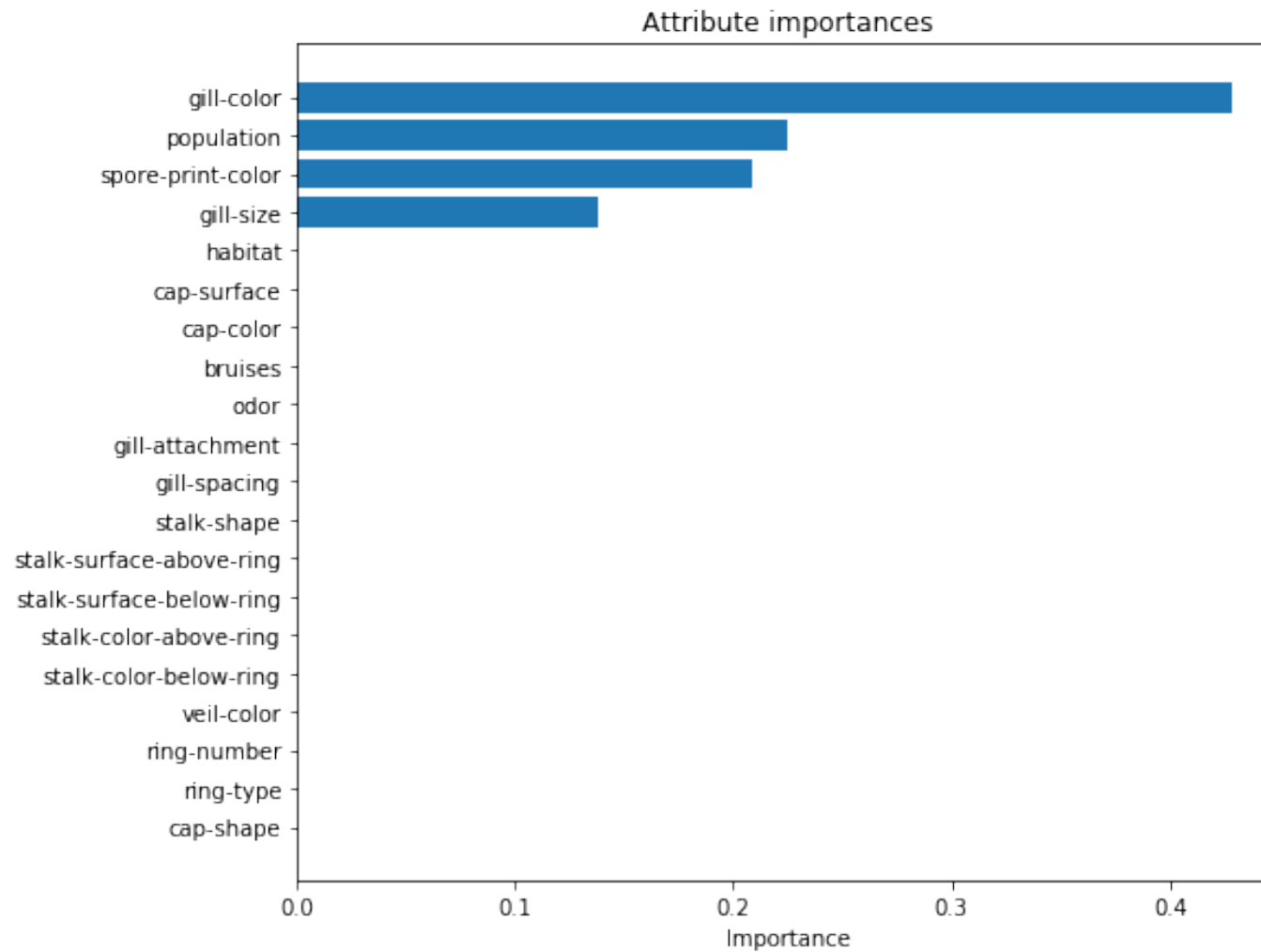
jednocześnie zachowując wysoką jakość klasyfikacji. Podczas eksperymentów zauważono, że w przypadku analizowanego zbioru danych kryterium podziału nie ma istotnego wpływu na wygląd i jakość drzewa.

```
In [16]: clf = build_tree(X_dropped_cols, X_train_dc, X_test_dc, Y_train_dc, Y_test_dc, random_state,  
                        criterion="gini", max_depth=2, min_samples_split=200, min_samples_leaf=60, max_leaf_nodes=5)
```



Accuracy = 0.941440 / Precision = 0.919575 / Recall = 0.959620

```
In [17]: attribute_importance(clf, X_dropped_cols)
```



Ustalenie własnych parametrów drzewa decyzyjnego pozwoliło zmniejszyć jego głębokość do 3, przy jednoczesnym zachowaniu dobrej jakości klasyfikacji. Współczynnik dokładności zmniejszył się jedynie do około 94%, wartość Precission do około 91%, a Recall do około 95%. Zauważono również, że najważniejsze atrybuty, tj. gill-color, spore-print-color, population oraz gill-size zostały zachowane.