https://www.agmrc.org/commodities-products/grains-oilseeds/dry-edible-bean-profile

Cel projektu:

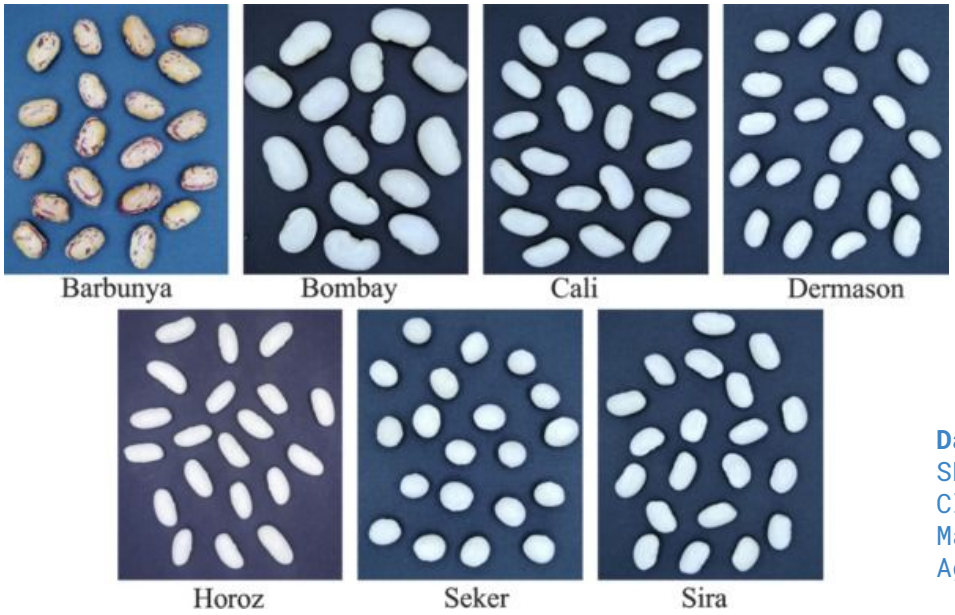
Klasyfikacja różnego gatunku fasolek na podstawie cech fizycznych (wymiarów).

Potencjalne zastosowania biznesowe:

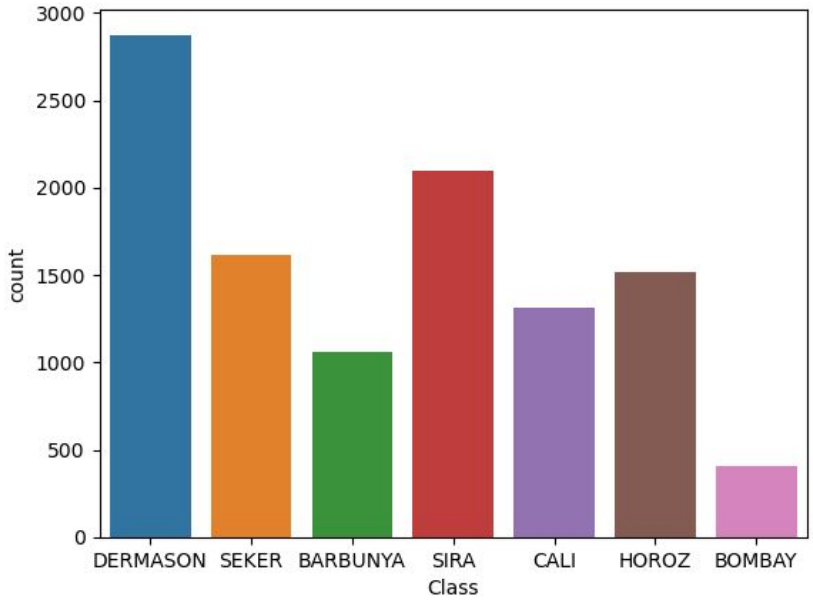
- hurtownie żywności
- aplikacja do gotowania, używająca zdjęć składników użytkownika
- kontrola jakości na plantacji fasoli

Dane: Dry Bean Dataset

- 7 rodzajów fasolek
- 16 cech fizycznych



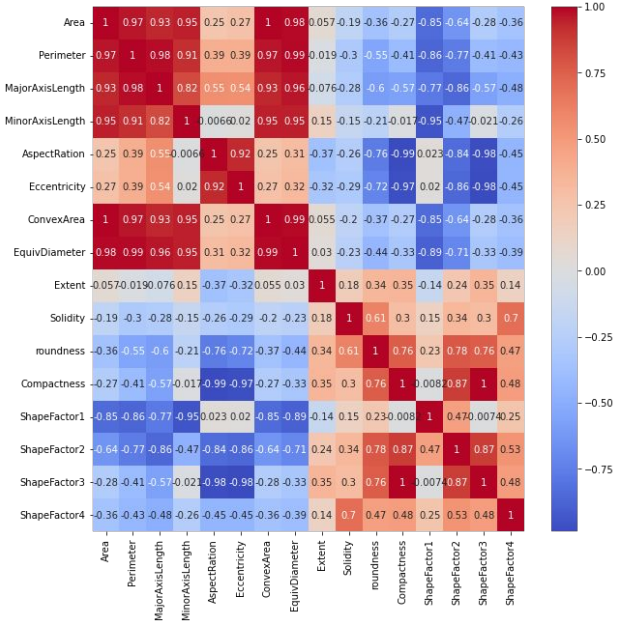
https://ars.els-cdn.com/content/image/1-s2.0-S0168169919311573-gr3.jpg



Dataset Citation (artykuł podany jako źródło na kagglu):
SKOKLU, M. and OZKAN, I.A., (2020), Multiclass Classification of Dry Beans Using Computer Vision and Machine Learning Techniques. Computers and Electronics in Agriculture, 174, 105507.

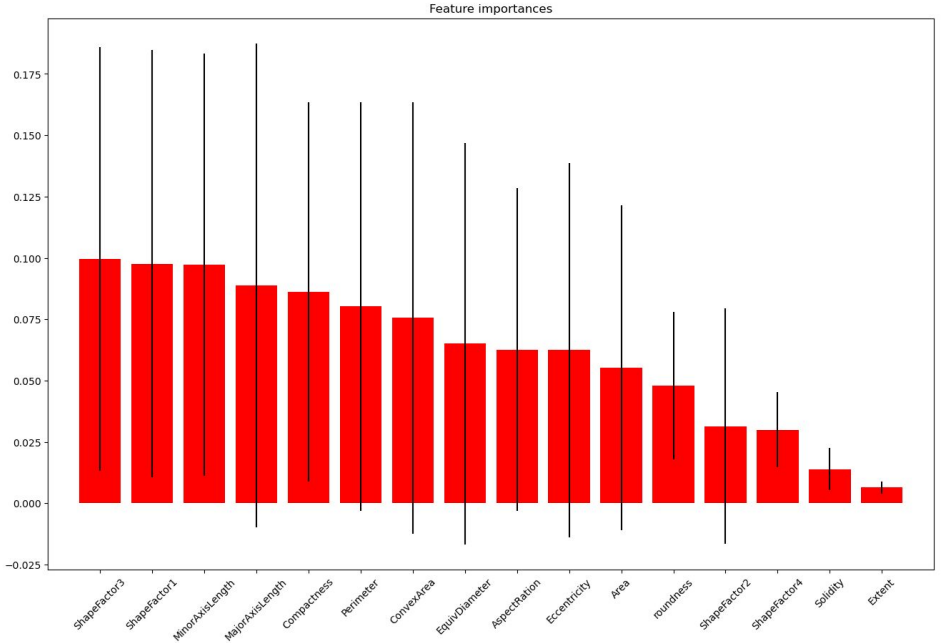
Wnioski z EDA:

- brak brakujących danych
- kilka mocnych korelacji: kandydaci do usunięcia
- Bombay odstaje od reszty



Feature selection:

- na podstawie macierzy korelacji: *usunięto kolumn z top 3 korelacjami*
- ranking na podstawie Random Forest: *usunięto cechy z 2 najmniejszymi wartościami*



Przetestowane metody standaryzacji:

- MinMaxScaler (mały negatywny wpływ)
- box-cox (regresja działa lepiej)
- **Standard Scaler** (finalnie użyty)

Encoding:

- OneHotEncoding
- LabelEncoder *(działa lepiej dla Random Forest)*

- binarne dla one vs rest

ONE VS REST (z użyciem Random Forest)

	BOMBAY	SEKER	DERMASON	HOROZ	SIRA	BARBUNYA	CALI
accuracy	1.00000	0.98863	0.95669	0.98688	0.95188	0.98731	0.98206
f1 score (macro avg)	1.00000	0.97732	0.94446	0.97275	0.92080	0.96301	0.95618
accuracy (selected features)	0.99956	0.97594	0.93526	0.97813	0.92738	0.97550	0.97332
f1 score (selected features)	0.99693	0.95232	0.91718	0.95493	0.88380	0.92987	0.93726

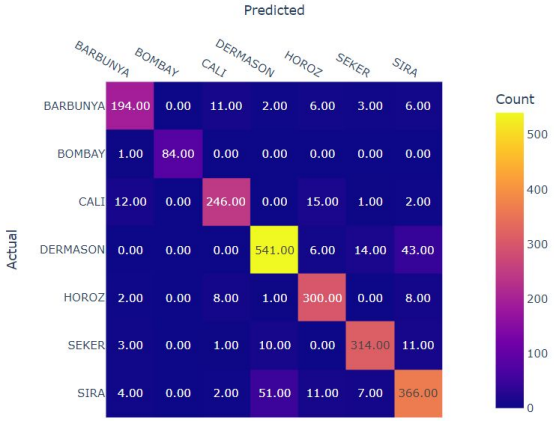
*usunięcie kolumn daje gorsze wyniki

	1	2	3	4	5
accuracy	0.92	0.92	0.89	**0.93**	0.74
f1 score (macro avg)	0.93	0.93	0.91	**0.94**	0.69
accuracy (selected features)	0.92	0.92	0.89	**0.92**	-
f1 score (selected features)	0.93	0.93	0.90	**0.93**	-

*zwykły encoding działa lepiej niż one hot encoding dla Random Forest

Testowane modele:

1) Regresja Logistyczna
2) SVM
3) Decision Tree Classifier
4) **Random Forest**
5) AdaBoost



przykładowy confusion matrix dla DecisionTreeClasiifier

Wnioski:

- SIRA i DERMASON znacznie ciężej odróżnić od siebie niż od reszty fasolek

- na ogół usunięcie kolumn nie wypływa znacząco na wyniki

KM3

Strojenie parametrów + kroswalidacja dla modeli z KM2

Strojenie parametrów:

```python
def hyperparameters_tuner(estimator, param_distributions, X, y, cv=5, n_iter=10, random_state=42):
    random_search = RandomizedSearchCV(estimator, param_distributions=param_distributions, n_iter=n_iter, cv=cv, random_state=random_state)
    random_search.fit(X, y)
    return random_search.best_params_
```

Kroswalidacja:

```python
def train_evaluate(estimator, param_distributions, X_train, y_train, X_val, y_val, cv=5, class_names=['BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON
    best_params = hyperparameters_tuner(estimator, param_distributions, X_train, y_train)
    best_model = estimator.set_params(**best_params)
    best_model.fit(X_train, y_train)

    y_pred = best_model.predict(X_val)


    accuracy = accuracy_score(y_val, y_pred)

    cv_results = cross_val_score(best_model, X_train, y_train, cv=cv)
    cv_val_results = cross_val_score(best_model, X_val, y_val, cv=cv)

    print(f"Best parameters: {best_params}")
    print('_____')
    print(f"Accuracy: {accuracy}")
    plot_confusion_matrix(y_val, y_pred, class_names)
    print(classification_report(y_val, y_pred, target_names=['BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKER', 'SIRA']))
    print("_____")
    print(f"Cross-validation results: {cv_results}")
    print(f"Mean accuracy: {cv_results.mean()}")
    print(f"Cross-validation results on validation set: {cv_val_results}")
    print(f"Mean accuracy on validation set: {cv_val_results.mean()}")
    return best_params
```

✓ 0.0s Python

Regresja Logistyczna

```
Best parameters: {'penalty': 'l1', 'C': 100}
```

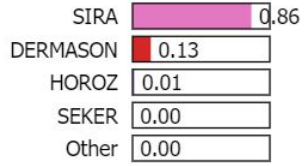
```
Accuracy: 0.9291338582677166
```

	precision	recall	f1-score	support
BARBUNYA	0.94	0.92	0.93	222
BOMBAY	1.00	1.00	1.00	85
CALI	0.96	0.92	0.94	276
DERMASON	0.93	0.93	0.93	604
HOROZ	0.93	0.95	0.94	319
SEKER	0.96	0.96	0.96	339
SIRA	0.87	0.88	0.88	441
accuracy			0.93	2286
macro avg	0.94	0.94	0.94	2286
weighted avg	0.93	0.93	0.93	2286

```
Cross-validation results: [0.92590164 0.93110236 0.93044619 0.92519685 0.92650919]
Mean accuracy: 0.927831246504023
Cross-validation results on validation set: [0.91484716 0.92778993 0.93435449 0.95185996 0.91466083]
Mean accuracy on validation set: 0.9287024738899028
```

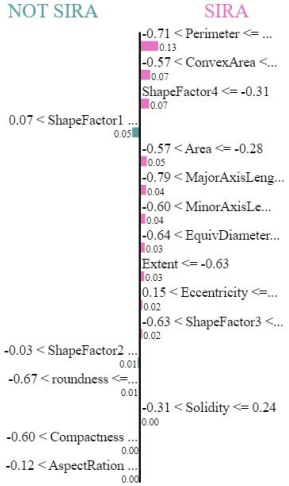
- model działa dobrze
- kroswalidacja nie wykrywa over/under-fittingu

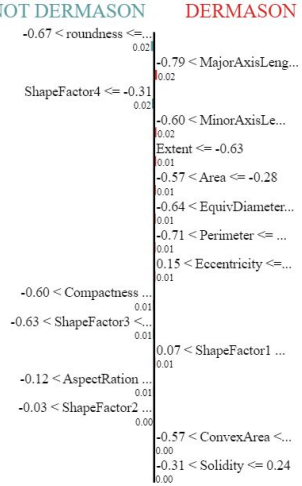
Prediction probabilities

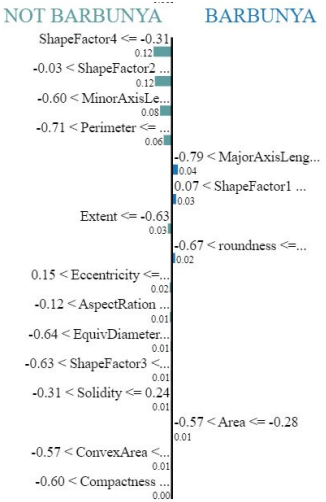


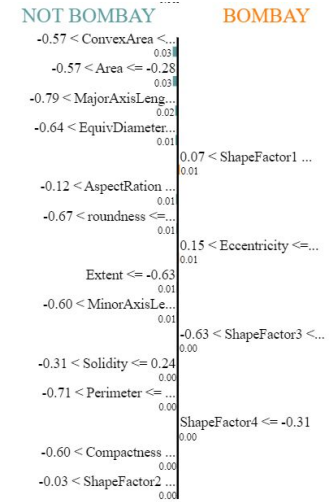
Feature	Value
Perimeter	-0.42
ConvexArea	-0.43
ShapeFactor4	-0.81
ShapeFactor1	0.44
Area	-0.43
MajorAxisLength	-0.40
MinorAxisLength	-0.46
EquivDiameter	-0.44
Extent	-1.09
Eccentricity	0.23
ShapeFactor3	-0.13
ShapeFactor2	0.03
roundness	-0.08
Solidity	-0.03
Compactness	-0.09
AspectRation	-0.04

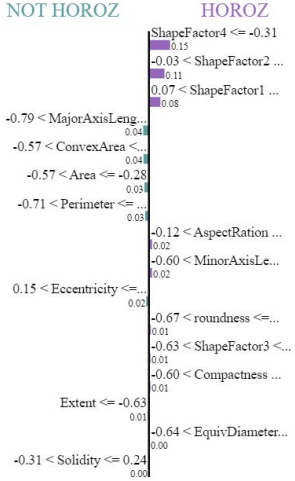
```
scaling = sklearn.preprocessing.PowerTransformer(method='box-cox')
X_train = scaling.fit_transform(X_train)
X_test = scaling.transform(X_test)
X_val = scaling.transform(X_val)
```

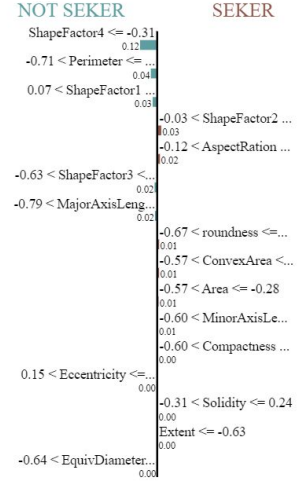


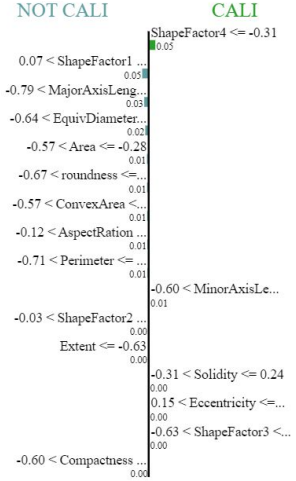


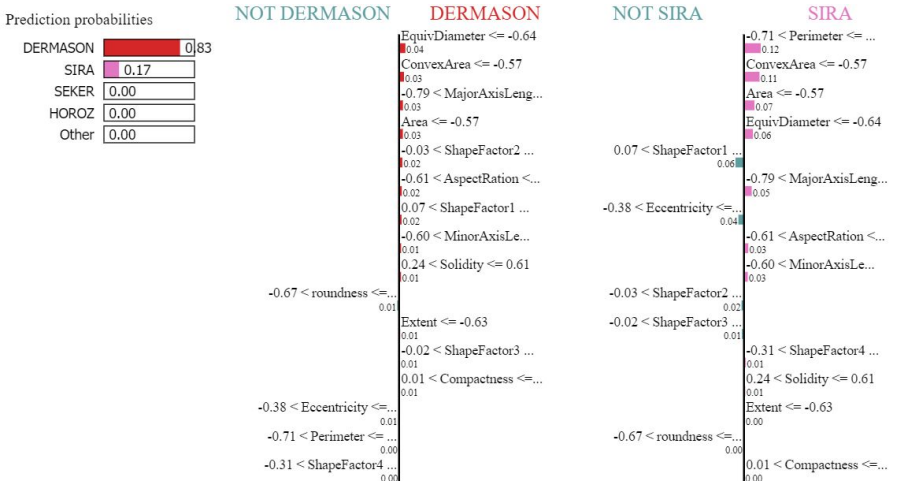


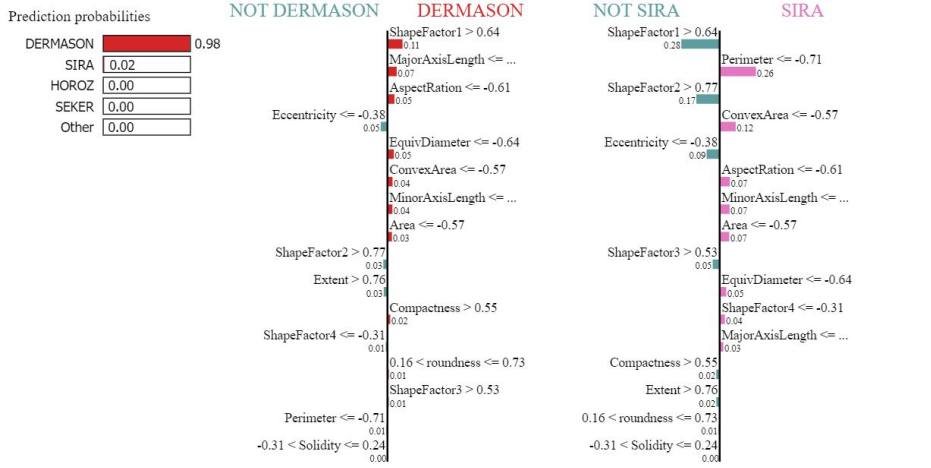


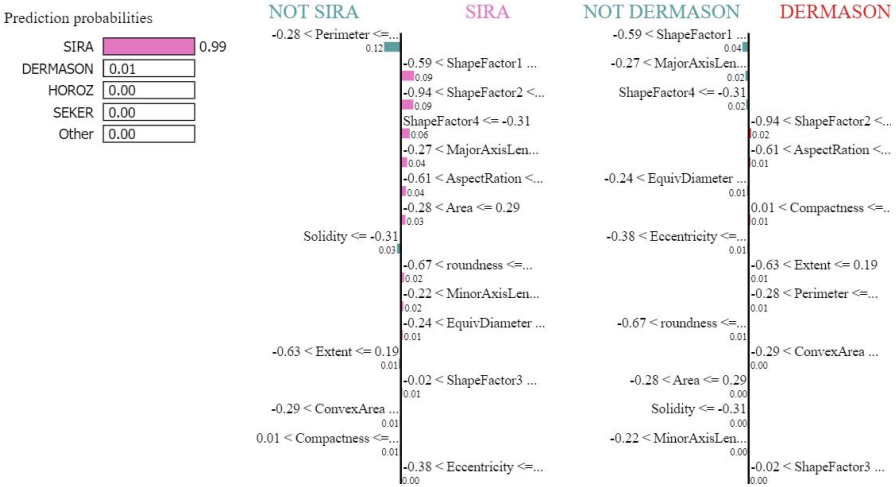


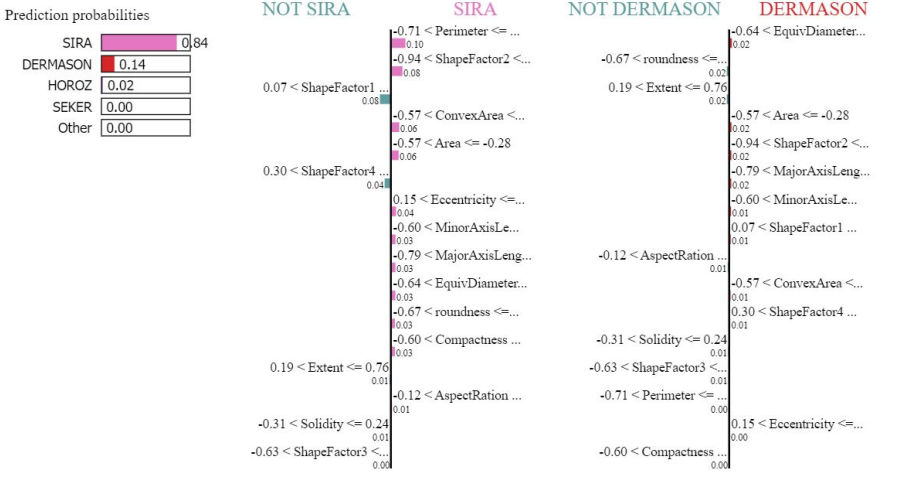





Prediction probabilities
DERMASON 0.83
SIRA 0.17
SEKER 0.00
HOROZ 0.00
Other 0.00
NOT DERMASON
DERMASON
EquivDiameter <= -0.64
ConvexArea <= -0.57
-0.79 < MajorAxisLeng...
Area <= -0.57
-0.03 < ShapeFactor2 ...
-0.61 < AspectRation <...
0.07 < ShapeFactor1 ...
-0.60 < MinorAxisLe...
0.24 < Solidity <= 0.61
-0.67 < roundness <=...
Extent <= -0.63
-0.02 < ShapeFactor3 ...
0.01 < Compactness <=...
-0.38 < Eccentricity <=...
-0.71 < Perimeter <= ...
-0.31 < ShapeFactor4 ...
NOT SIRA
SIRA
-0.71 < Perimeter <= ...
ConvexArea <= -0.57
Area <= -0.57
EquivDiameter <= -0.64
0.07 < ShapeFactor1 ...
-0.79 < MajorAxisLeng...
-0.38 < Eccentricity <=...
-0.61 < AspectRation <...
-0.03 < ShapeFactor2 ...
-0.60 < MinorAxisLe...
-0.02 < ShapeFactor3 ...
-0.31 < ShapeFactor4 ...
0.24 < Solidity <= 0.61
Extent <= -0.63
-0.67 < roundness <=...
0.01 < Compactness <=...


Prediction probabilities
DERMASON 0.98
SIRA 0.02
HOROZ 0.00
SEKER 0.00
Other 0.00
NOT DERMASON
DERMASON
ShapeFactor1 > 0.64
MajorAxisLength <= ...
AspectRation <= -0.61
Eccentricity <= -0.38
EquivDiameter <= -0.64
ConvexArea <= -0.57
MinorAxisLength <= ...
Area <= -0.57
ShapeFactor2 > 0.77
Extent > 0.76
Compactness > 0.55
ShapeFactor4 <= -0.31
0.16 < roundness <= 0.73
ShapeFactor3 > 0.53
Perimeter <= -0.71
-0.31 < Solidity <= 0.24
NOT SIRA
SIRA
ShapeFactor1 > 0.64
Perimeter <= -0.71
ShapeFactor2 > 0.77
ConvexArea <= -0.57
Eccentricity <= -0.38
AspectRation <= -0.61
MinorAxisLength <= ...
Area <= -0.57
ShapeFactor3 > 0.53
EquivDiameter <= -0.64
ShapeFactor4 <= -0.31
MajorAxisLength <= ...
Compactness > 0.55
Extent > 0.76
0.16 < roundness <= 0.73
-0.31 < Solidity <= 0.24


Prediction probabilities
SIRA 0.99
DERMASON 0.01
HOROZ 0.00
SEKER 0.00
Other 0.00
NOT SIRA
SIRA
-0.28 < Perimeter <=...
-0.59 < ShapeFactor1 ...
-0.94 < ShapeFactor2 <...
ShapeFactor4 <= -0.31
-0.27 < MajorAxisLen...
-0.61 < AspectRation <...
-0.28 < Area <= 0.29
Solidity <= -0.31
-0.67 < roundness <=...
-0.22 < MinorAxisLen...
-0.24 < EquivDiameter ...
-0.63 < Extent <= 0.19
-0.02 < ShapeFactor3 ...
-0.29 < ConvexArea ...
0.01 < Compactness <=...
-0.38 < Eccentricity <=...
NOT DERMASON
DERMASON
-0.59 < ShapeFactor1 ...
-0.27 < MajorAxisLen...
ShapeFactor4 <= -0.31
-0.94 < ShapeFactor2 <...
-0.61 < AspectRation <...
-0.24 < EquivDiameter ...
0.01 < Compactness <=...
-0.38 < Eccentricity <=...
-0.63 < Extent <= 0.19
-0.28 < Perimeter <=...
-0.67 < roundness <=...
-0.29 < ConvexArea ...
-0.28 < Area <= 0.29
Solidity <= -0.31
-0.22 < MinorAxisLen...
-0.02 < ShapeFactor3 ...


Prediction probabilities
SIRA 0.84
DERMASON 0.14
HOROZ 0.02
SEKER 0.00
Other 0.00
NOT SIRA
SIRA
-0.71 < Perimeter <= ...
-0.94 < ShapeFactor2 <...
0.07 < ShapeFactor1 ...
-0.57 < ConvexArea <...
-0.57 < Area <= -0.28
0.30 < ShapeFactor4 ...
0.15 < Eccentricity <=...
-0.60 < MinorAxisLe...
-0.79 < MajorAxisLeng...
-0.64 < EquivDiameter...
-0.67 < roundness <=...
-0.60 < Compactness ...
0.19 < Extent <= 0.76
-0.12 < AspectRation ...
-0.31 < Solidity <= 0.24
-0.63 < ShapeFactor3 <...
NOT DERMASON
DERMASON
-0.64 < EquivDiameter...
-0.67 < roundness <=...
0.19 < Extent <= 0.76
-0.57 < Area <= -0.28
-0.94 < ShapeFactor2 <...
-0.79 < MajorAxisLeng...
-0.60 < MinorAxisLe...
0.07 < ShapeFactor1 ...
-0.12 < AspectRation ...
-0.57 < ConvexArea <...
0.30 < ShapeFactor4 ...
-0.31 < Solidity <= 0.24
-0.63 < ShapeFactor3 <...
-0.71 < Perimeter <= ...
0.15 < Eccentricity <=...
-0.60 < Compactness ...

SVM

```
Best parameters: {'kernel': 'rbf', 'gamma': 0.01, 'C': 1000}
```

```
Accuracy: 0.9330708661417323
```

	precision	recall	f1-score	support
BARBUNYA	0.95	0.93	0.94	222
BOMBAY	1.00	1.00	1.00	85
CALI	0.95	0.92	0.94	276
DERMASON	0.92	0.95	0.93	604
HOROZ	0.94	0.96	0.95	319
SEKER	0.97	0.96	0.97	339
SIRA	0.89	0.87	0.88	441
accuracy			0.93	2286
macro avg	0.95	0.94	0.94	2286
weighted avg	0.93	0.93	0.93	2286

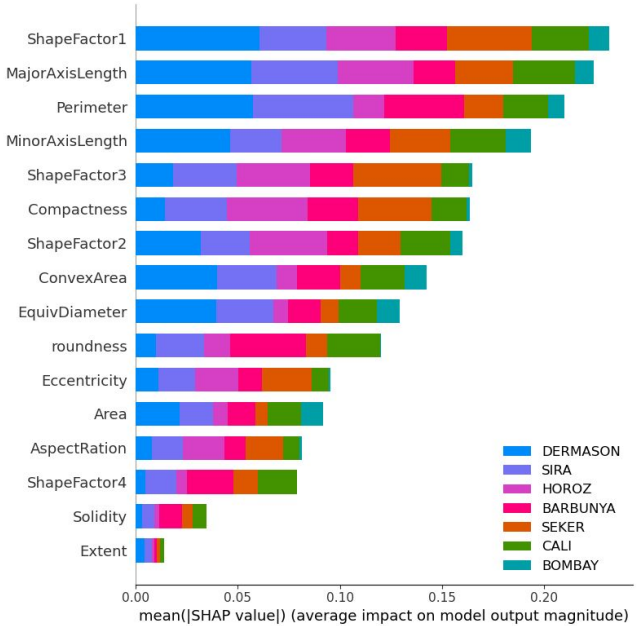
```
Cross-validation results: [0.92327869 0.93307087 0.93175853 0.93110236 0.92650919]
Mean accuracy: 0.929143926681296
Cross-validation results on validation set: [0.91266376 0.91247265 0.93435449 0.92997812 0.91028446]
Mean accuracy on validation set: 0.9199506941989242
```

Random Forest

Best parameters: {'n_estimators': 200, 'min_samples_split': 5, 'max_depth': 25, 'criterion': 'log_loss'}

Accuracy: 0.9243219597550306

	precision	recall	f1-score	support
BARBUNYA	0.93	0.93	0.93	222
BOMBAY	1.00	0.99	0.99	85
CALI	0.95	0.92	0.94	276
DERMASON	0.91	0.93	0.92	604
HOROZ	0.94	0.95	0.94	319
SEKER	0.96	0.96	0.96	339
SIRA	0.87	0.86	0.87	441
accuracy			0.92	2286
macro avg	0.94	0.93	0.94	2286
weighted avg	0.92	0.92	0.92	2286



Cross-validation results: [0.9147541 0.92782152 0.92454068 0.92388451 0.92716535]

Mean accuracy: 0.9236332343702939

Cross-validation results on validation set: [0.91266376 0.9059081 0.92997812 0.92997812 0.89496718]

Mean accuracy on validation set: 0.9146990530610685

KM3 - dodatkowe modele

Naive Bayes

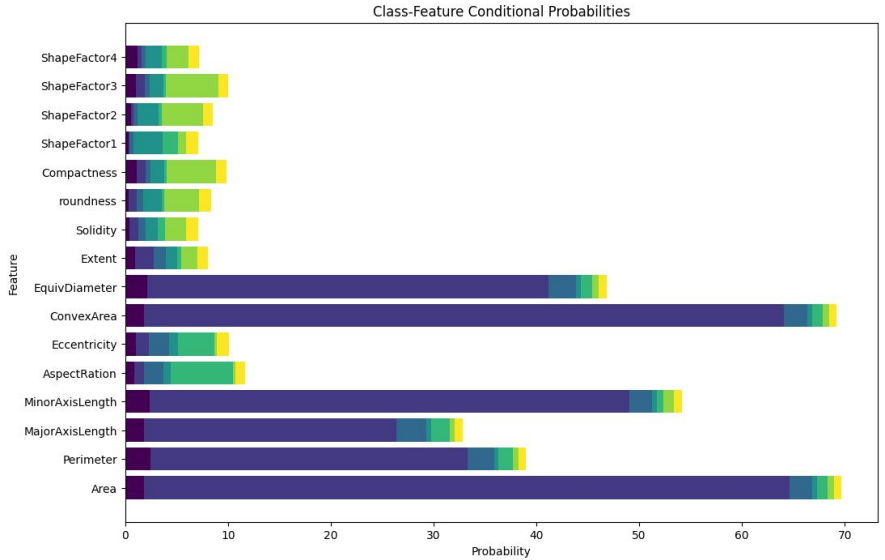
```
Best parameters: {'var_smoothing': 2.848035868435799e-08}
```

```
Accuracy: 0.9002624671916011
```

	precision	recall	f1-score	support
BARBUNYA	0.89	0.85	0.87	222
BOMBAY	1.00	1.00	1.00	85
CALI	0.91	0.89	0.90	276
DERMASON	0.93	0.88	0.90	604
HOROZ	0.91	0.96	0.93	319
SEKER	0.93	0.96	0.94	339
SIRA	0.82	0.85	0.84	441
accuracy			0.90	2286
macro avg	0.91	0.91	0.91	2286
weighted avg	0.90	0.90	0.90	2286

```
Cross-validation results: [0.89180328 0.89895013 0.9015748  0.89238845 0.89501312]
Mean accuracy: 0.8959459575749753
Cross-validation results on validation set: [0.8930131  0.9059081  0.90809628 0.9059081  0.88402626]
Mean accuracy on validation set: 0.8993903662580145
```

```
Global Explanation for Naive Bayes Classifier:
----------------------------------------------
Class Priors (Prior Probabilities):
BARBUNYA: 0.0975
BOMBAY: 0.0371
CALI: 0.1206
DERMASON: 0.2640
HOROZ: 0.1396
SEKER: 0.1483
SIRA: 0.1929
```



generalnie słabo wypada ten model

KM3 - dodatkowe modele

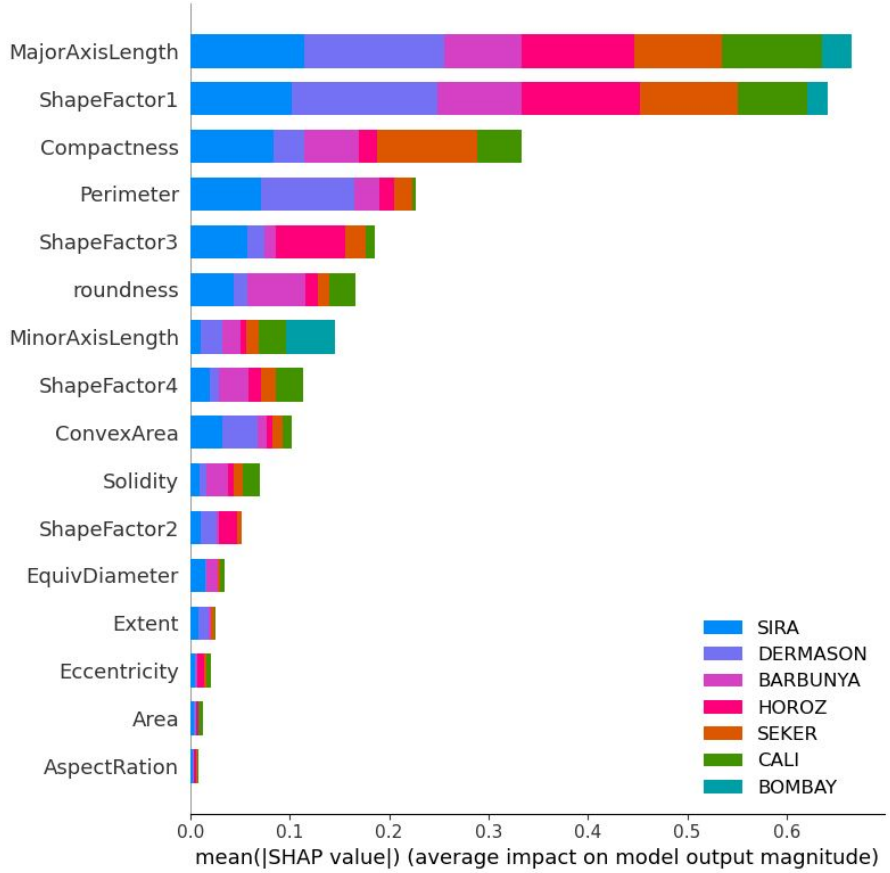
DecisionTree

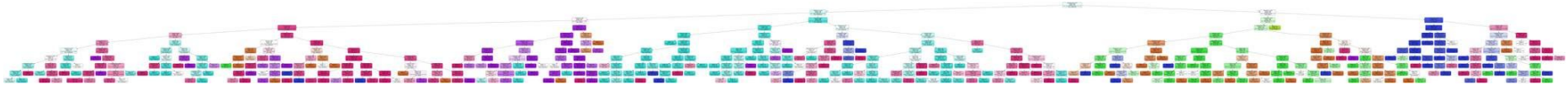
```
Best parameters: {'min_samples_split': 10, 'max_depth': 20, 'criterion': 'entropy'}
```

```
Accuracy: 0.8985126859142607
```

	precision	recall	f1-score	support
BARBUNYA	0.86	0.89	0.88	222
BOMBAY	1.00	0.99	0.99	85
CALI	0.92	0.89	0.90	276
DERMASON	0.89	0.92	0.90	604
HOROZ	0.92	0.94	0.93	319
SEKER	0.96	0.91	0.93	339
SIRA	0.84	0.83	0.83	441
accuracy			0.90	2286
macro avg	0.91	0.91	0.91	2286
weighted avg	0.90	0.90	0.90	2286

```
Cross-validation results: [0.88590164 0.89895013 0.90354331 0.88845144 0.89501312]
Mean accuracy: 0.8943719289187213
Cross-validation results on validation set: [0.88209607 0.88621444 0.89277899 0.8643326  0.86652079]
Mean accuracy on validation set: 0.8783885794004951
```





KM3 - dodatkowe modele

K Neighbors, SVC

Best parameters: {'n_neighbors': 9}

Accuracy: 0.9243219597550306

	precision	recall	f1-score	support
BARBUNYA	0.97	0.91	0.94	222
BOMBAY	1.00	1.00	1.00	85
CALI	0.95	0.93	0.94	276
DERMASON	0.91	0.93	0.92	604
HOROZ	0.93	0.96	0.94	319
SEKER	0.96	0.96	0.96	339
SIRA	0.86	0.85	0.86	441
accuracy			0.92	2286
macro avg	0.94	0.93	0.94	2286
weighted avg	0.92	0.92	0.92	2286

Cross-validation results: [0.91672131 0.93307087 0.9324147 0.92191601 0.92519685]
Mean accuracy: 0.9258639473344521
Cross-validation results on validation set: [0.90611354 0.9059081 0.92341357 0.93435449 0.90809628]
Mean accuracy on validation set: 0.9155771932003862

Best parameters: {'kernel': 'rbf', 'gamma': 0.01, 'C': 10000}

Accuracy: 0.9278215223097113

	precision	recall	f1-score	support
BARBUNYA	0.92	0.92	0.92	222
BOMBAY	1.00	1.00	1.00	85
CALI	0.95	0.91	0.93	276
DERMASON	0.91	0.95	0.93	604
HOROZ	0.94	0.95	0.94	319
SEKER	0.97	0.95	0.96	339
SIRA	0.89	0.86	0.88	441
accuracy			0.93	2286
macro avg	0.94	0.94	0.94	2286
weighted avg	0.93	0.93	0.93	2286

Cross-validation results: [0.92459016 0.93569554 0.92913386 0.93175853 0.92782152]
Mean accuracy: 0.9298999225506647
Cross-validation results on validation set: [0.89519651 0.90809628 0.93654267 0.92560175 0.9059081]
Mean accuracy on validation set: 0.9142690606098247

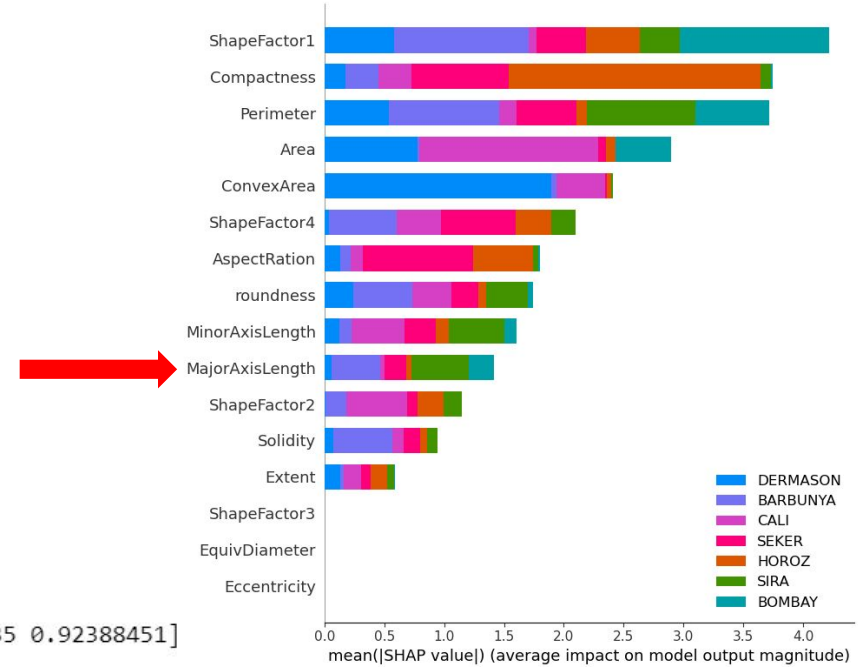
KM3 - dodatkowe modele

XGBOOST

```
Best parameters: {'subsample': 1.0, 'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.1, 'gamma': 0, 'colsample_bytree': 1.0}
```

```
Accuracy: 0.9282589676290464
```

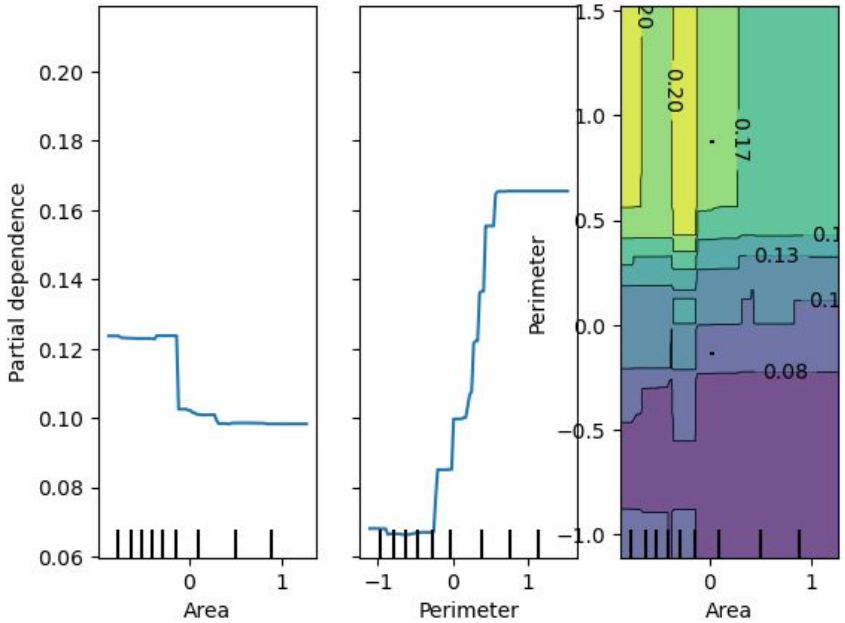
	precision	recall	f1-score	support
BARBUNYA	0.94	0.92	0.93	222
BOMBAY	1.00	1.00	1.00	85
CALI	0.97	0.92	0.94	276
DERMASON	0.91	0.94	0.93	604
HOROZ	0.93	0.96	0.95	319
SEKER	0.96	0.96	0.96	339
SIRA	0.88	0.86	0.87	441
accuracy			0.93	2286
macro avg	0.94	0.94	0.94	2286
weighted avg	0.93	0.93	0.93	2286



```
Cross-validation results: [0.92721311 0.93897638 0.92847769 0.92716535 0.92388451]
Mean accuracy: 0.9291434103523946
Cross-validation results on validation set: [0.90829694 0.89715536 0.93435449 0.93654267 0.89277899]
Mean accuracy on validation set: 0.9138256906156537
```

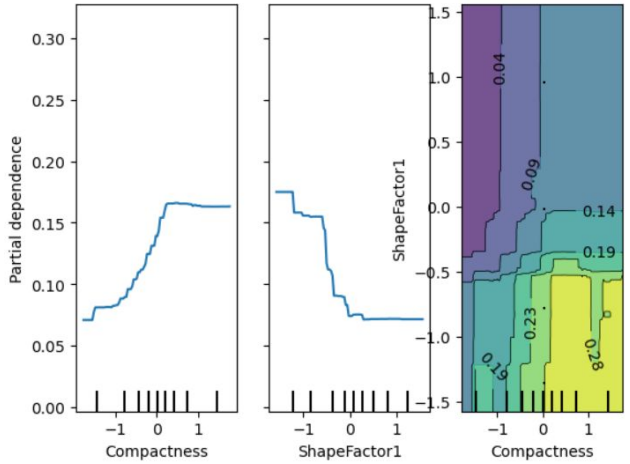
Partial Dependence Plots

BARBUNYA

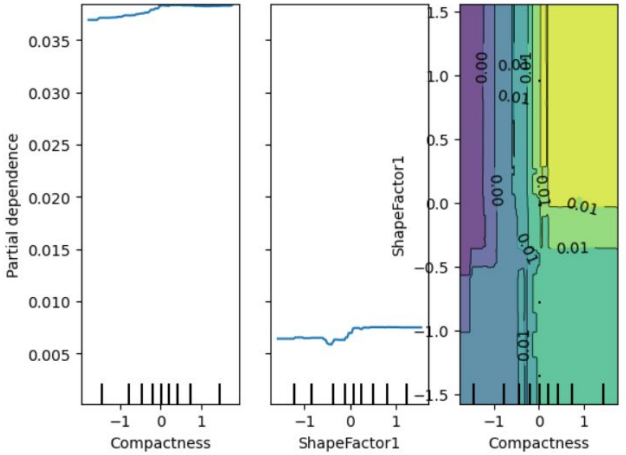



```
scaling = sklearn.preprocessing.PowerTransformer(method='box-cox')
X_train = scaling.fit_transform(X_train)
X_test = scaling.transform(X_test)
X_val = scaling.transform(X_val)
```

BARBUNYA

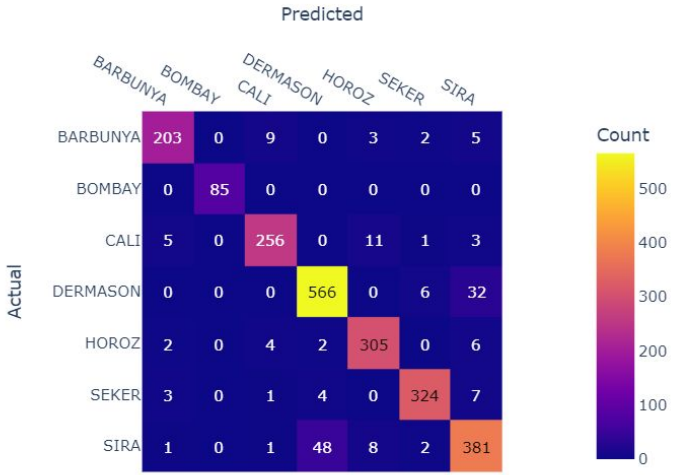


BOMBAY



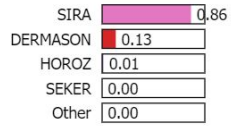
KM3 - dodatkowe modele
Stacking Classifier

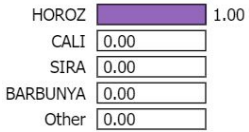
```
from sklearn.ensemble import StackingClassifier


lr = LogisticRegression(C=100, penalty='l1', max_iter=1000,solver='saga', multi_class='multinomial')
svc = SVC(kernel= 'rbf', gamma= 0.01, C= 10000)
dt = DecisionTreeClassifier(min_samples_split= 10, max_depth= 15, criterion ='entropy')
nb = GaussianNB(var_smoothing= 2.848035868435799e-08)
kn = KNeighborsClassifier(n_neighbors= 9)
rf = RandomForestClassifier(n_estimators= 200, min_samples_split= 5, max_depth = 25, criterion= "log_loss")

models = [
    ('lr', lr),
    ('svc', svc), ('nb', nb), ('rf', rf), ('dt', dt), ('kn', kn)]
stack = StackingClassifier(estimators=models, final_estimator=LogisticRegression( max_iter=1000,solver='saga', multi_class='multinomial'))
stack.fit(X_train, y_encoded2)
y_pred = stack.predict(X_val)
y_val = labelencoder.inverse_transform(y_val_encoded2)
y_pred = labelencoder.inverse_transform(y_pred)
print(classification_report(y_val, y_pred, target_names=['BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKER', 'SIRA']))
plot_confusion_matrix(y_val, y_pred, class_names)
```

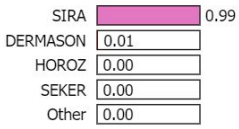


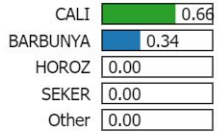

```
Cross-validation results: [0.92393443 0.93700787 0.93044619 0.93044619 0.92913386]
Mean accuracy: 0.9301937093928834
Cross-validation results on validation set: [0.91048035 0.91247265 0.94310722 0.94748359 0.91247265]
Mean accuracy on validation set: 0.9252032908755602
```

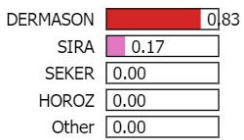
Regresja Logistyczna

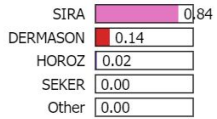




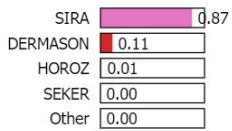


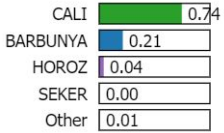


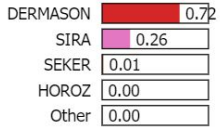


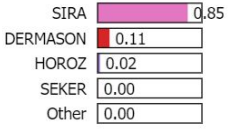


XGBoost

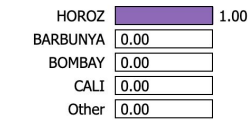


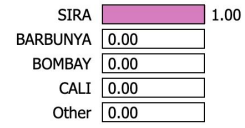


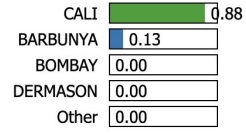


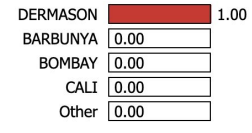


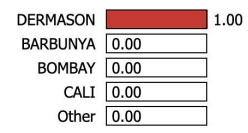
KNeighbors

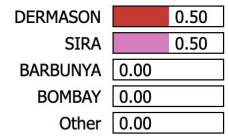




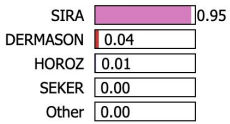


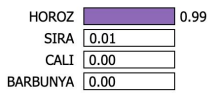


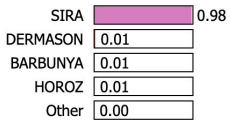


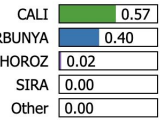


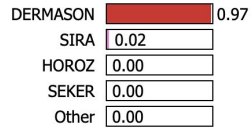
Stacking

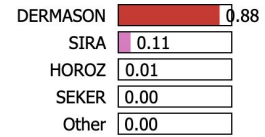


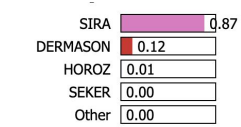




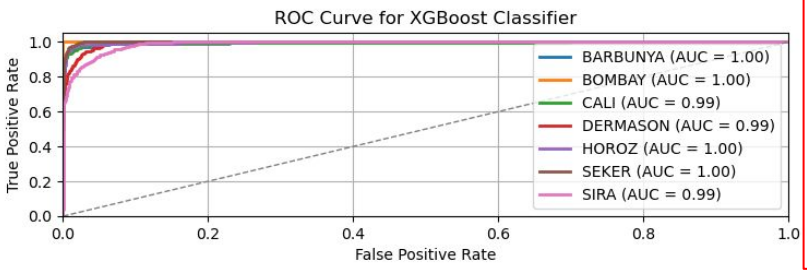


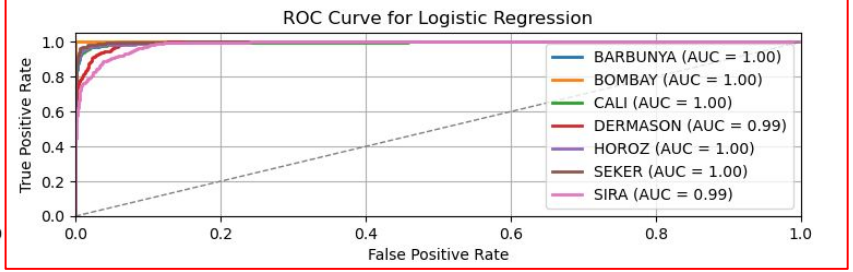


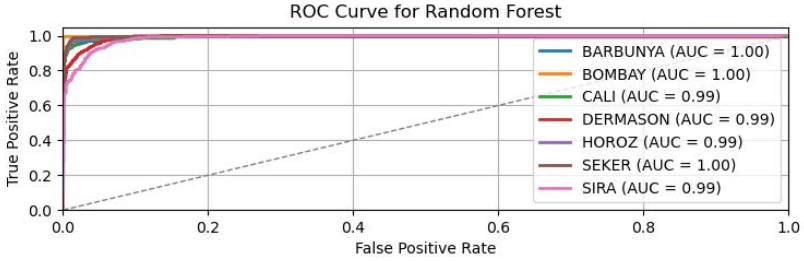


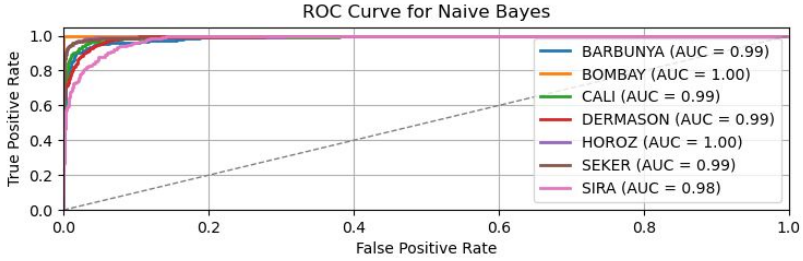


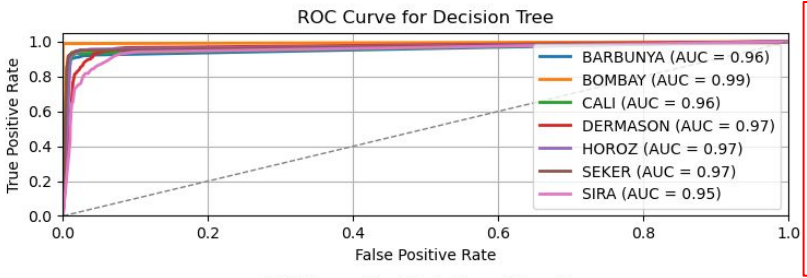
CROSS-VALIDATION RESULTS	RL	RF	SVC	NB	DT	KN	XGB	Stacking
before tuning accuracy	0.92	0.93	0.92	-	0.89	-	-	-
accuracy	**0.923**	0.924	0.928	0.900	0.899	0.924	0.928	**0.930**
cv mean accuracy	**0.925**	0.924	0.930	0.896	0.898	0.926	0.929	**0.930**
cv mean accuracy val	**0.923**	0.914	0.914	0.899	0.881	0.916	0.913	**0.925**

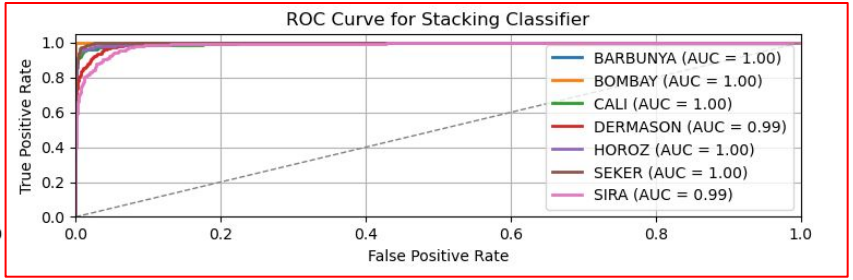

ROC Curve for XGBoost Classifier
True Positive Rate
False Positive Rate
BARBUNYA (AUC = 1.00)
BOMBAY (AUC = 1.00)
CALI (AUC = 0.99)
DERMASON (AUC = 0.99)
HOROZ (AUC = 1.00)
SEKER (AUC = 1.00)
SIRA (AUC = 0.99)

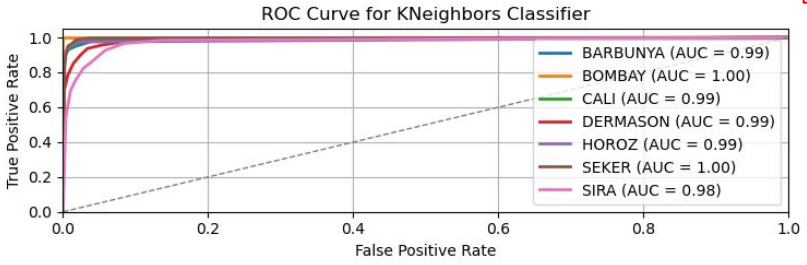

ROC Curve for Logistic Regression
True Positive Rate
False Positive Rate
BARBUNYA (AUC = 1.00)
BOMBAY (AUC = 1.00)
CALI (AUC = 1.00)
DERMASON (AUC = 0.99)
HOROZ (AUC = 1.00)
SEKER (AUC = 1.00)
SIRA (AUC = 0.99)

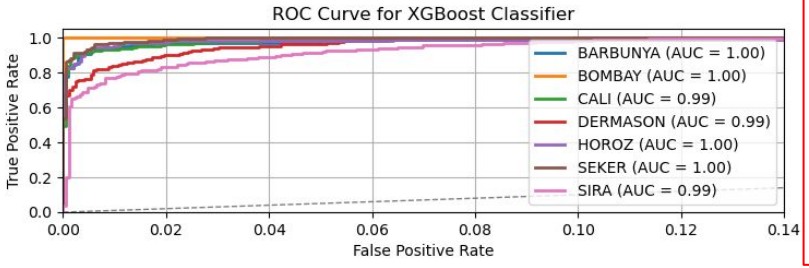

ROC Curve for Random Forest
True Positive Rate
False Positive Rate
BARBUNYA (AUC = 1.00)
BOMBAY (AUC = 1.00)
CALI (AUC = 0.99)
DERMASON (AUC = 0.99)
HOROZ (AUC = 0.99)
SEKER (AUC = 1.00)
SIRA (AUC = 0.99)

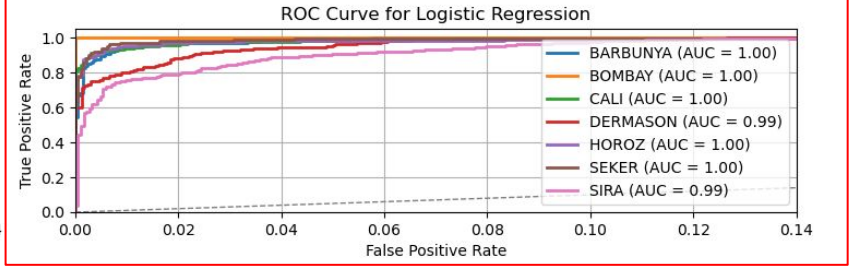

ROC Curve for Naive Bayes
True Positive Rate
False Positive Rate
BARBUNYA (AUC = 0.99)
BOMBAY (AUC = 1.00)
CALI (AUC = 0.99)
DERMASON (AUC = 0.99)
HOROZ (AUC = 1.00)
SEKER (AUC = 0.99)
SIRA (AUC = 0.98)

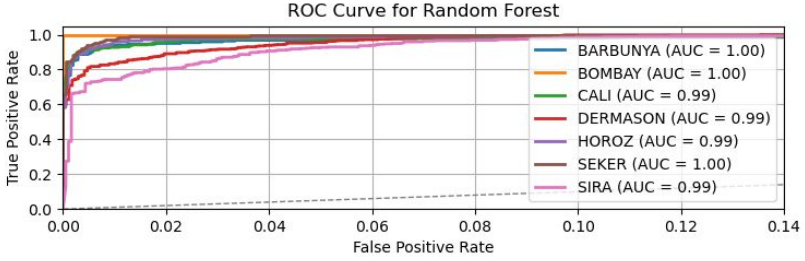

ROC Curve for Decision Tree
True Positive Rate
False Positive Rate
BARBUNYA (AUC = 0.96)
BOMBAY (AUC = 0.99)
CALI (AUC = 0.96)
DERMASON (AUC = 0.97)
HOROZ (AUC = 0.97)
SEKER (AUC = 0.97)
SIRA (AUC = 0.95)

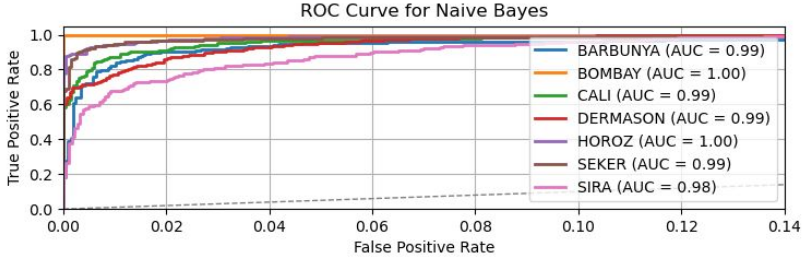

ROC Curve for Stacking Classifier
True Positive Rate
False Positive Rate
BARBUNYA (AUC = 1.00)
BOMBAY (AUC = 1.00)
CALI (AUC = 1.00)
DERMASON (AUC = 0.99)
HOROZ (AUC = 1.00)
SEKER (AUC = 1.00)
SIRA (AUC = 0.99)

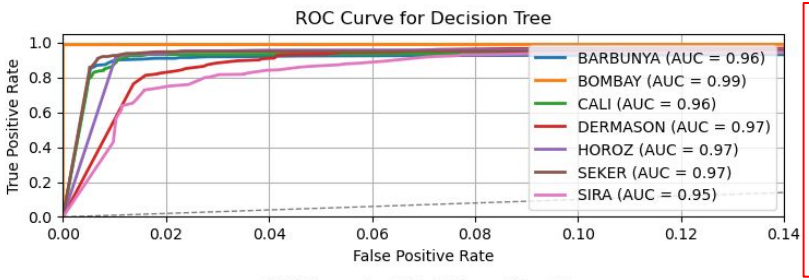

ROC Curve for KNeighbors Classifier
True Positive Rate
False Positive Rate
BARBUNYA (AUC = 0.99)
BOMBAY (AUC = 1.00)
CALI (AUC = 0.99)
DERMASON (AUC = 0.99)
HOROZ (AUC = 0.99)
SEKER (AUC = 1.00)
SIRA (AUC = 0.98)

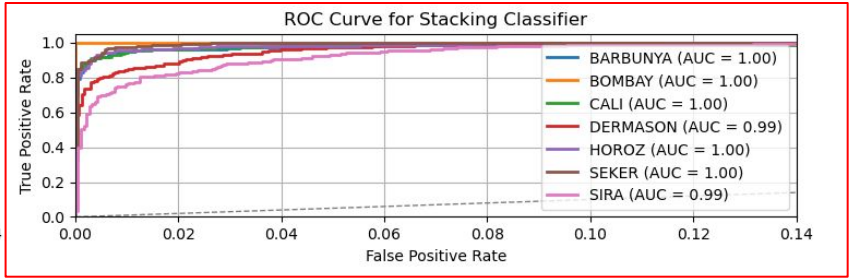

ROC Curve for XGBoost Classifier
True Positive Rate
False Positive Rate
BARBUNYA (AUC = 1.00)
BOMBAY (AUC = 1.00)
CALI (AUC = 0.99)
DERMASON (AUC = 0.99)
HOROZ (AUC = 1.00)
SEKER (AUC = 1.00)
SIRA (AUC = 0.99)

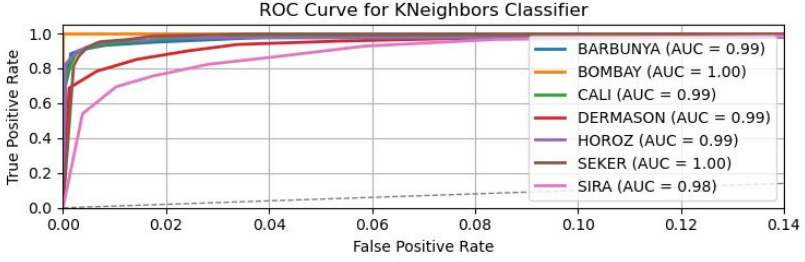

ROC Curve for Logistic Regression
True Positive Rate
False Positive Rate
BARBUNYA (AUC = 1.00)
BOMBAY (AUC = 1.00)
CALI (AUC = 1.00)
DERMASON (AUC = 0.99)
HOROZ (AUC = 1.00)
SEKER (AUC = 1.00)
SIRA (AUC = 0.99)


ROC Curve for Random Forest
True Positive Rate
False Positive Rate
BARBUNYA (AUC = 1.00)
BOMBAY (AUC = 1.00)
CALI (AUC = 0.99)
DERMASON (AUC = 0.99)
HOROZ (AUC = 0.99)
SEKER (AUC = 1.00)
SIRA (AUC = 0.99)


ROC Curve for Naive Bayes
True Positive Rate
False Positive Rate
BARBUNYA (AUC = 0.99)
BOMBAY (AUC = 1.00)
CALI (AUC = 0.99)
DERMASON (AUC = 0.99)
HOROZ (AUC = 1.00)
SEKER (AUC = 0.99)
SIRA (AUC = 0.98)


ROC Curve for Decision Tree
True Positive Rate
False Positive Rate
BARBUNYA (AUC = 0.96)
BOMBAY (AUC = 0.99)
CALI (AUC = 0.96)
DERMASON (AUC = 0.97)
HOROZ (AUC = 0.97)
SEKER (AUC = 0.97)
SIRA (AUC = 0.95)


ROC Curve for Stacking Classifier
True Positive Rate
False Positive Rate
BARBUNYA (AUC = 1.00)
BOMBAY (AUC = 1.00)
CALI (AUC = 1.00)
DERMASON (AUC = 0.99)
HOROZ (AUC = 1.00)
SEKER (AUC = 1.00)
SIRA (AUC = 0.99)


ROC Curve for KNeighbors Classifier
True Positive Rate
False Positive Rate
BARBUNYA (AUC = 0.99)
BOMBAY (AUC = 1.00)
CALI (AUC = 0.99)
DERMASON (AUC = 0.99)
HOROZ (AUC = 0.99)
SEKER (AUC = 1.00)
SIRA (AUC = 0.98)

AutoML - tpot

```
from tpot import TPOTClassifier

tpot = TPOTClassifier(generations=3,verbosity=2)

tpot.fit(X_train, y_encoded2)
```

✓ 81m 51.8s

generalnie wyniki nie są dużo lepsze od np. regresji logistycznej, więc uznajemy poprzednie modele za wystarczające

```
Generation 1 - Current best internal CV score: 0.9307187298309023

Generation 2 - Current best internal CV score: 0.9307187298309023

Generation 3 - Current best internal CV score: 0.9307187298309023

Best pipeline: MLPClassifier(input_matrix, alpha=0.0001, learning_rate_init=0.001)
```

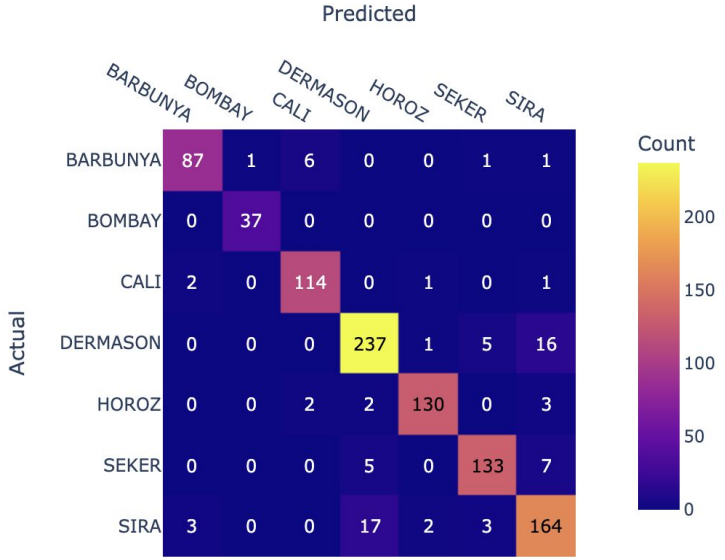
to, że nie udaje się wznieść accuracy powyżej 93% może wynikać z charakterystyki datasetu (fasolka jednego gatunku może mieć identyczne wymiary jak inny gatunek, co widać na zdjęciach)

TPOTClassifier

```
TPOTClassifier(generations=3, verbosity=2)
```

FINALNY MODEL: Stacking - test

	precision	recall	f1-score	support
BARBUNYA	0.95	0.91	0.93	96
BOMBAY	0.97	1.00	0.99	37
CALI	0.93	0.97	0.95	118
DERMASON	0.91	0.92	0.91	259
HOROZ	0.97	0.95	0.96	137
SEKER	0.94	0.92	0.93	145
SIRA	0.85	0.87	0.86	189
accuracy			0.92	981
macro avg	0.93	0.93	0.93	981
weighted avg	0.92	0.92	0.92	981



model accuracy on test data: 91,945%

https://www.agmrc.org/commodities-products/grains-oilseeds/dry-edible-bean-profile