Dry Bean Classification

Tomasz Żywicki
Bartłomiej Wójcik

1. Exploratary Data Analysis (EDA)

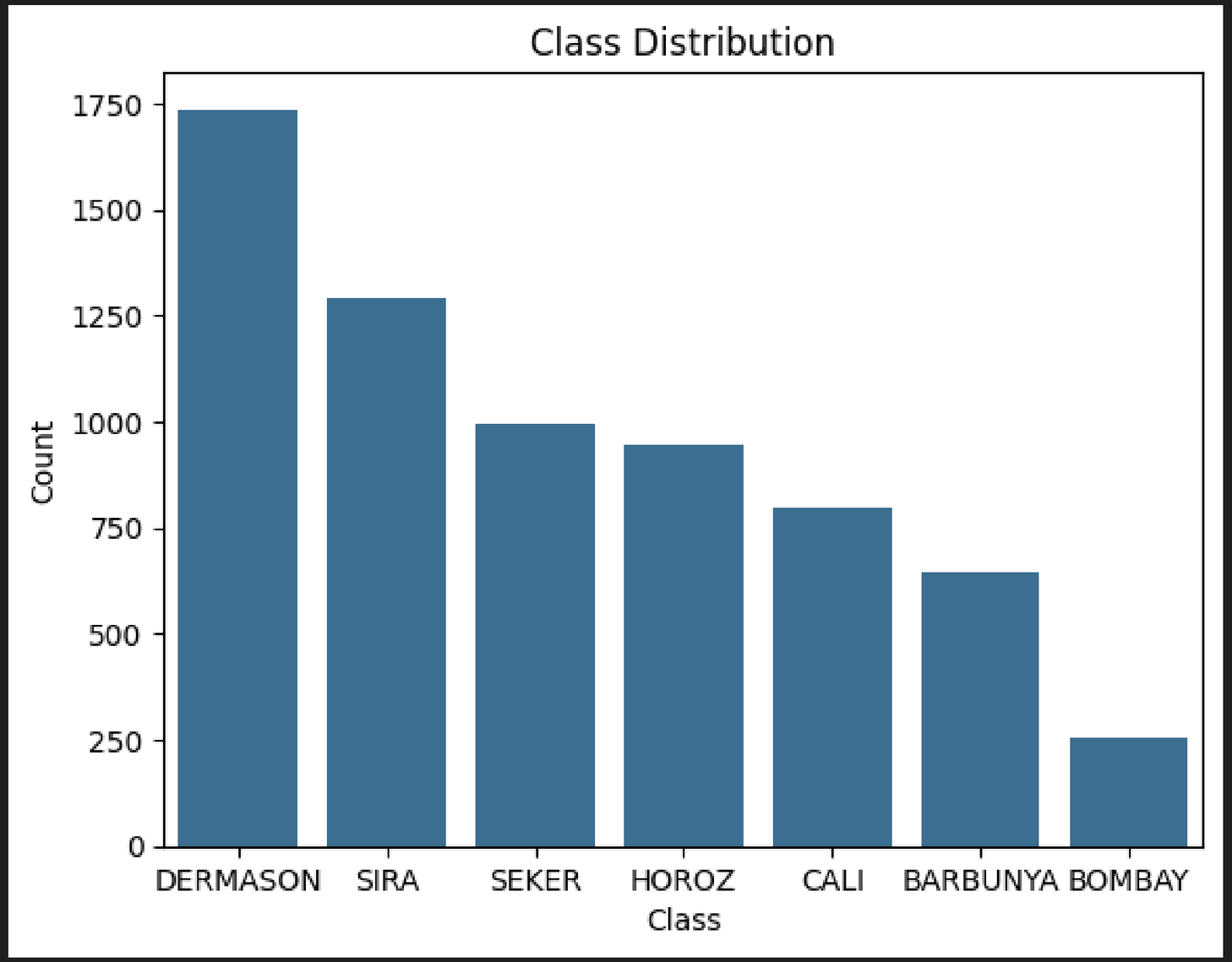
Descriptions of our variables

- 13611 rows
- 17 columns

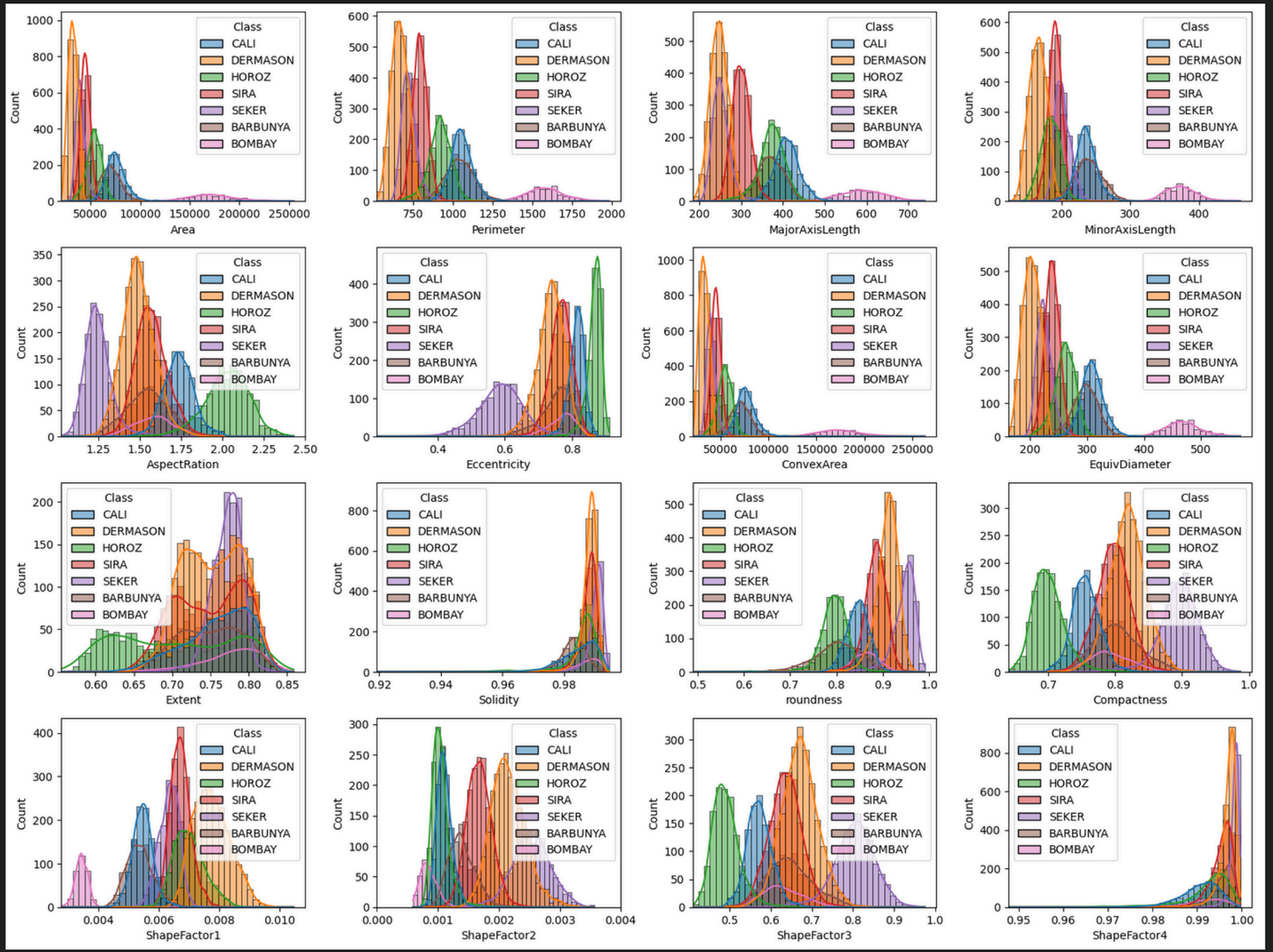
- size
- shape
- bean type

Variable Name	Type	Description
Area	Numeric	The area of a bean zone and the number of pixels within its boundaries.
Perimeter	Numeric	Bean circumference is defined as the length of its border.
MajorAxisLength	Numeric	The distance between the ends of the longest line that can be drawn from a bean.
MinorAxisLength	Numeric	The longest line that can be drawn from the bean while standing perpendicular to the main axis.
AspectRatio	Numeric	Defines the relationship between MajorAxisLength and MinorAxisLength.
Eccentricity	Numeric	Eccentricity of the ellipse having the same moments as the region.
ConvexArea	Numeric	Number of pixels in the smallest convex polygon that can contain the area of a bean seed.
EquivDiameter	Numeric	The diameter of a circle having the same area as a bean seed area.
Extent	Numeric	The ratio of the pixels in the bounding box to the bean area.
Solidity	Numeric	Also known as convexity. The ratio of the pixels in the convex shell to those found in beans.
Roundness	Numeric	Calculated with the formula: (4π * Area) / (Perimeter^2).
Compactness	Numeric	Measures the roundness of an object: Eccentricity divided by AspectRatio.
Shape Factor 1	Numeric	One of the shape factors.
Shape Factor 2	Numeric	One of the shape factors.
Shape Factor 3	Numeric	One of the shape factors.
Shape Factor 4	Numeric	One of the shape factors.
Class	Categorical	One of 7 different bean types.

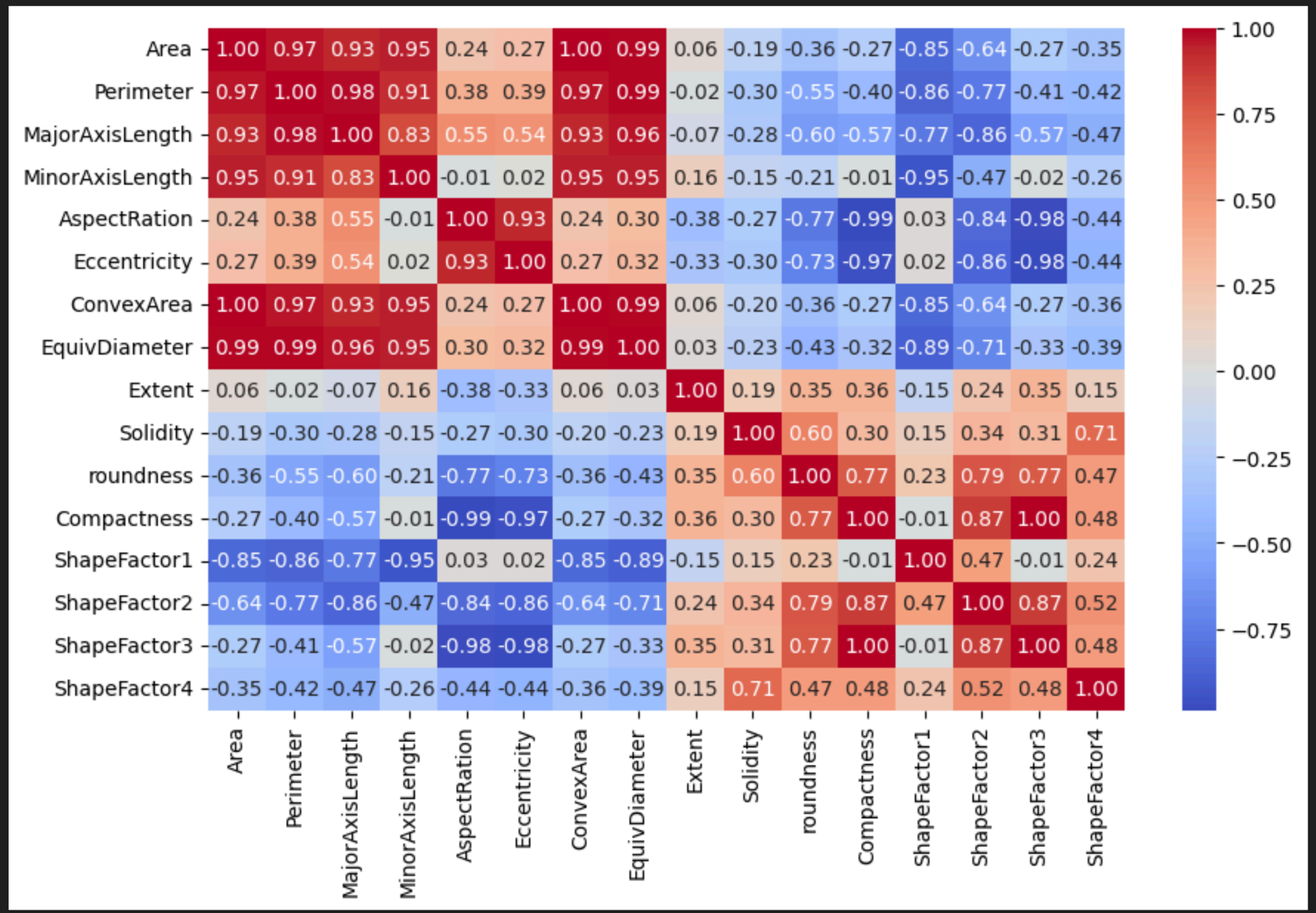
Target variable distribution



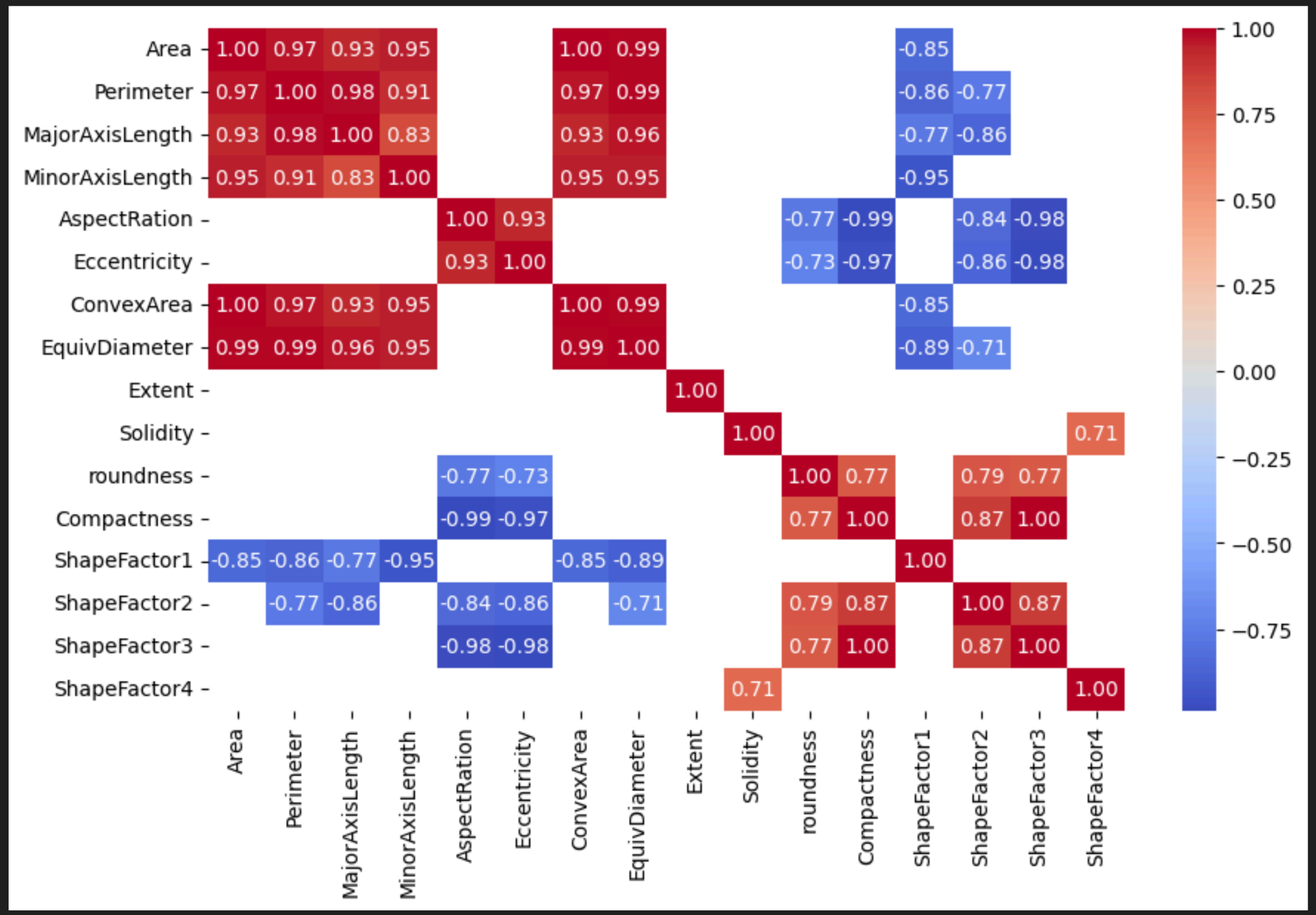
Variables distributions



Heatmap


Area 1.00 0.97 0.93 0.95 0.24 0.27 1.00 0.99 0.06 -0.19 -0.36 -0.27 -0.85 -0.64 -0.27 -0.35
Perimeter 0.97 1.00 0.98 0.91 0.38 0.39 0.97 0.99 -0.02 -0.30 -0.55 -0.40 -0.86 -0.77 -0.41 -0.42
MajorAxisLength 0.93 0.98 1.00 0.83 0.55 0.54 0.93 0.96 -0.07 -0.28 -0.60 -0.57 -0.77 -0.86 -0.57 -0.47
MinorAxisLength 0.95 0.91 0.83 1.00 -0.01 0.02 0.95 0.95 0.16 -0.15 -0.21 -0.01 -0.95 -0.47 -0.02 -0.26
AspectRation 0.24 0.38 0.55 -0.01 1.00 0.93 0.24 0.30 -0.38 -0.27 -0.77 -0.99 0.03 -0.84 -0.98 -0.44
Eccentricity 0.27 0.39 0.54 0.02 0.93 1.00 0.27 0.32 -0.33 -0.30 -0.73 -0.97 0.02 -0.86 -0.98 -0.44
ConvexArea 1.00 0.97 0.93 0.95 0.24 0.27 1.00 0.99 0.06 -0.20 -0.36 -0.27 -0.85 -0.64 -0.27 -0.36
EquivDiameter 0.99 0.99 0.96 0.95 0.30 0.32 0.99 1.00 0.03 -0.23 -0.43 -0.32 -0.89 -0.71 -0.33 -0.39
Extent 0.06 -0.02 -0.07 0.16 -0.38 -0.33 0.06 0.03 1.00 0.19 0.35 0.36 -0.15 0.24 0.35 0.15
Solidity -0.19 -0.30 -0.28 -0.15 -0.27 -0.30 -0.20 -0.23 0.19 1.00 0.60 0.30 0.15 0.34 0.31 0.71
roundness -0.36 -0.55 -0.60 -0.21 -0.77 -0.73 -0.36 -0.43 0.35 0.60 1.00 0.77 0.23 0.79 0.77 0.47
Compactness -0.27 -0.40 -0.57 -0.01 -0.99 -0.97 -0.27 -0.32 0.36 0.30 0.77 1.00 -0.01 0.87 1.00 0.48
ShapeFactor1 -0.85 -0.86 -0.77 -0.95 0.03 0.02 -0.85 -0.89 -0.15 0.15 0.23 -0.01 1.00 0.47 -0.01 0.24
ShapeFactor2 -0.64 -0.77 -0.86 -0.47 -0.84 -0.86 -0.64 -0.71 0.24 0.34 0.79 0.87 0.47 1.00 0.87 0.52
ShapeFactor3 -0.27 -0.41 -0.57 -0.02 -0.98 -0.98 -0.27 -0.33 0.35 0.31 0.77 1.00 -0.01 0.87 1.00 0.48
ShapeFactor4 -0.35 -0.42 -0.47 -0.26 -0.44 -0.44 -0.36 -0.39 0.15 0.71 0.47 0.48 0.24 0.52 0.48 1.00
Area
Perimeter
MajorAxisLength
MinorAxisLength
AspectRation
Eccentricity
ConvexArea
EquivDiameter
Extent
Solidity
roundness
Compactness
ShapeFactor1
ShapeFactor2
ShapeFactor3
ShapeFactor4
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Heatmap


Area 1.00 0.97 0.93 0.95 1.00 0.99 -0.85
Perimeter 0.97 1.00 0.98 0.91 0.97 0.99 -0.86 -0.77
MajorAxisLength 0.93 0.98 1.00 0.83 0.93 0.96 -0.77 -0.86
MinorAxisLength 0.95 0.91 0.83 1.00 0.95 0.95 -0.95
AspectRation 1.00 0.93 -0.77 -0.99 -0.84 -0.98
Eccentricity 0.93 1.00 -0.73 -0.97 -0.86 -0.98
ConvexArea 1.00 0.97 0.93 0.95 1.00 0.99 -0.85
EquivDiameter 0.99 0.99 0.96 0.95 0.99 1.00 -0.89 -0.71
Extent 1.00
Solidity 1.00 0.71
roundness -0.77 -0.73 1.00 0.77 0.79 0.77
Compactness -0.99 -0.97 0.77 1.00 0.87 1.00
ShapeFactor1 -0.85 -0.86 -0.77 -0.95 -0.85 -0.89 1.00
ShapeFactor2 -0.77 -0.86 -0.84 -0.86 -0.71 0.79 0.87 1.00 0.87
ShapeFactor3 -0.98 -0.98 0.77 1.00 0.87 1.00
ShapeFactor4 0.71 1.00
Area
Perimeter
MajorAxisLength
MinorAxisLength
AspectRation
Eccentricity
ConvexArea
EquivDiameter
Extent
Solidity
roundness
Compactness
ShapeFactor1
ShapeFactor2
ShapeFactor3
ShapeFactor4
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

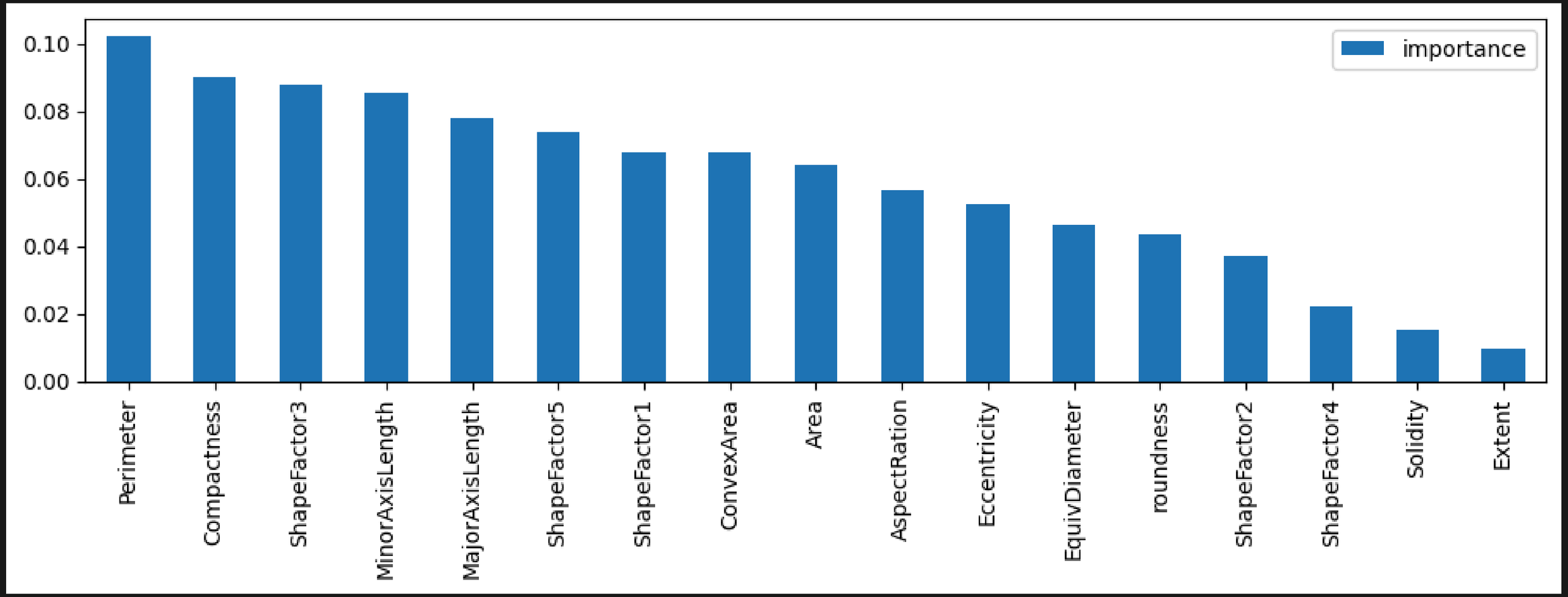
2. Basic model evaluations

```python
def evaluate_model_basic(models, X, y, X_test, y_test):
    results = []
    for model_name, model in models:
        if model_name == 'XGBoost':
            y = LabelEncoder().fit_transform(y)
            y_test = LabelEncoder().fit_transform(y_test)
        model.fit(X, y)
        y_pred = model.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred, average='weighted')
        recall = recall_score(y_test, y_pred, average='weighted')
        results.append((model_name, accuracy, precision, recall))
    results = pd.DataFrame(results, columns=['Model Name', 'Accuracy', 'Precision', 'Recall'])
    results.sort_values('Accuracy', ascending=False, inplace=True)
    results.reset_index(drop=True, inplace=True)
    return results
```

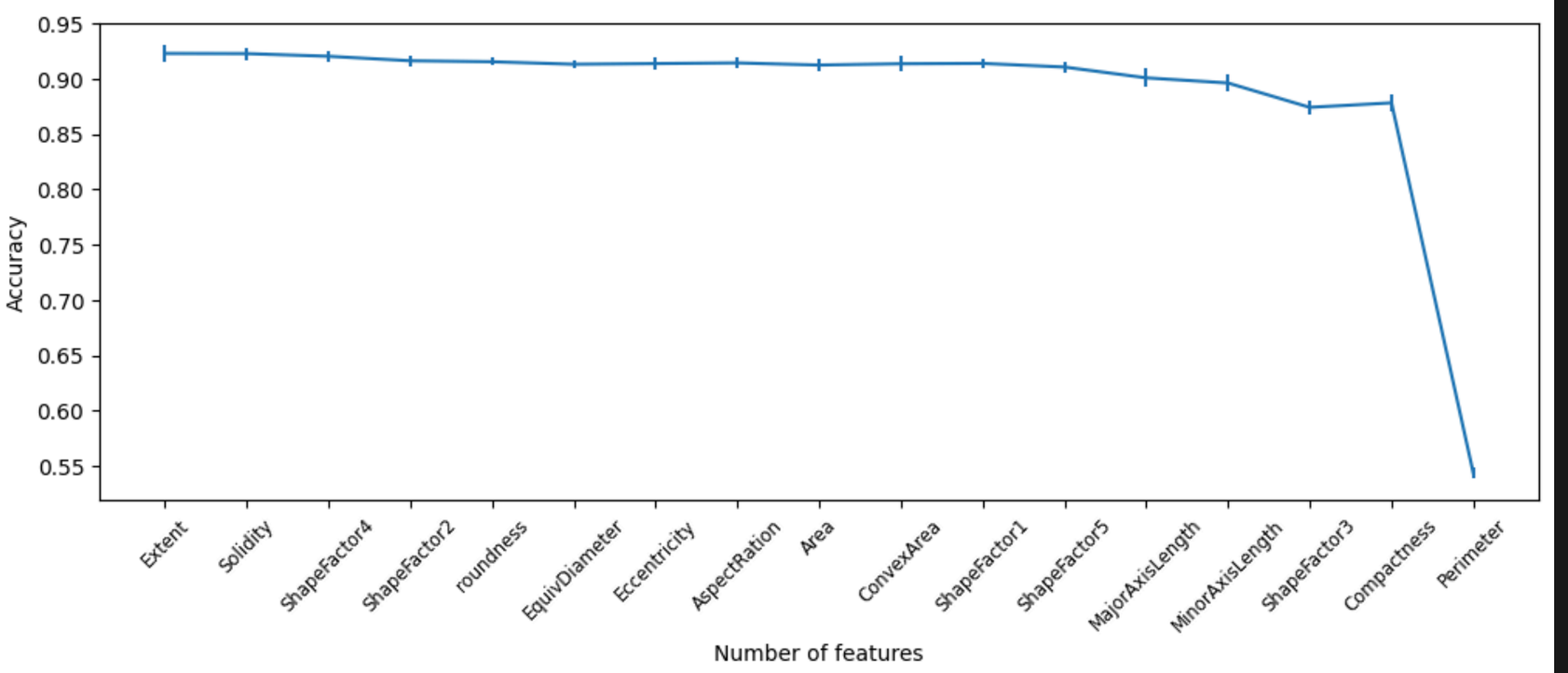
	Model Name	Accuracy	Precision	Recall
0	LGBM	0.9286	0.9289	0.9286
1	XGBoost	0.9286	0.9289	0.9286
2	Random Forest	0.9220	0.9222	0.9220
3	SVC	0.9108	0.9114	0.9108
4	Decision Tree	0.9080	0.9091	0.9080
5	K Nearest Neighbors	0.7118	0.7086	0.7118
6	Linear SVC	0.4781	0.4452	0.4781
7	Dummy Classifier	0.2606	0.0679	0.2606

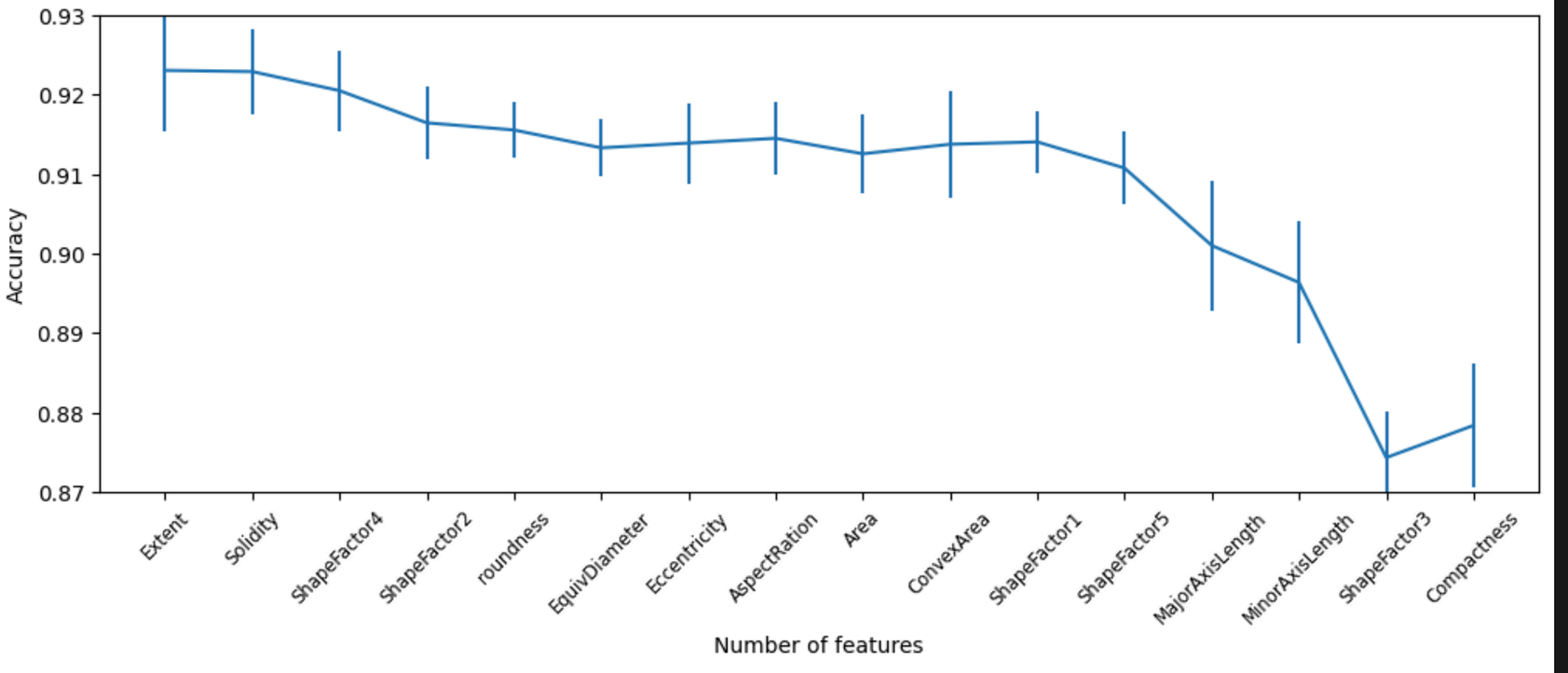
3. Feature selection

Numeric importances of features



```python
def evaluate_feature_importance(X, y, importance_threshold):
    forest = RandomForestClassifier(random_state=311)
    forest.fit(X, y)

    feature_importances = pd.DataFrame({'feature': X.columns, 'importance': forest.feature_importances_}).sort_values('importance', ascending=True)

    clf = RandomForestClassifier(random_state=0)
    clf.fit(X, y)

    results = []

    for i in range(len(feature_importances)):
        selected_features = feature_importances['feature'].iloc[i:]

        if feature_importances['importance'].iloc[i] > importance_threshold:
            break

        scores = cross_val_score(clf, X[selected_features], y, cv=5, scoring='accuracy')

        results.append({
            'num_features': len(selected_features),
            'accuracy': scores.mean(),
            'std': scores.std(),
            'dropped_feature': feature_importances['feature'].iloc[i]
        })

    return pd.DataFrame(results)
```

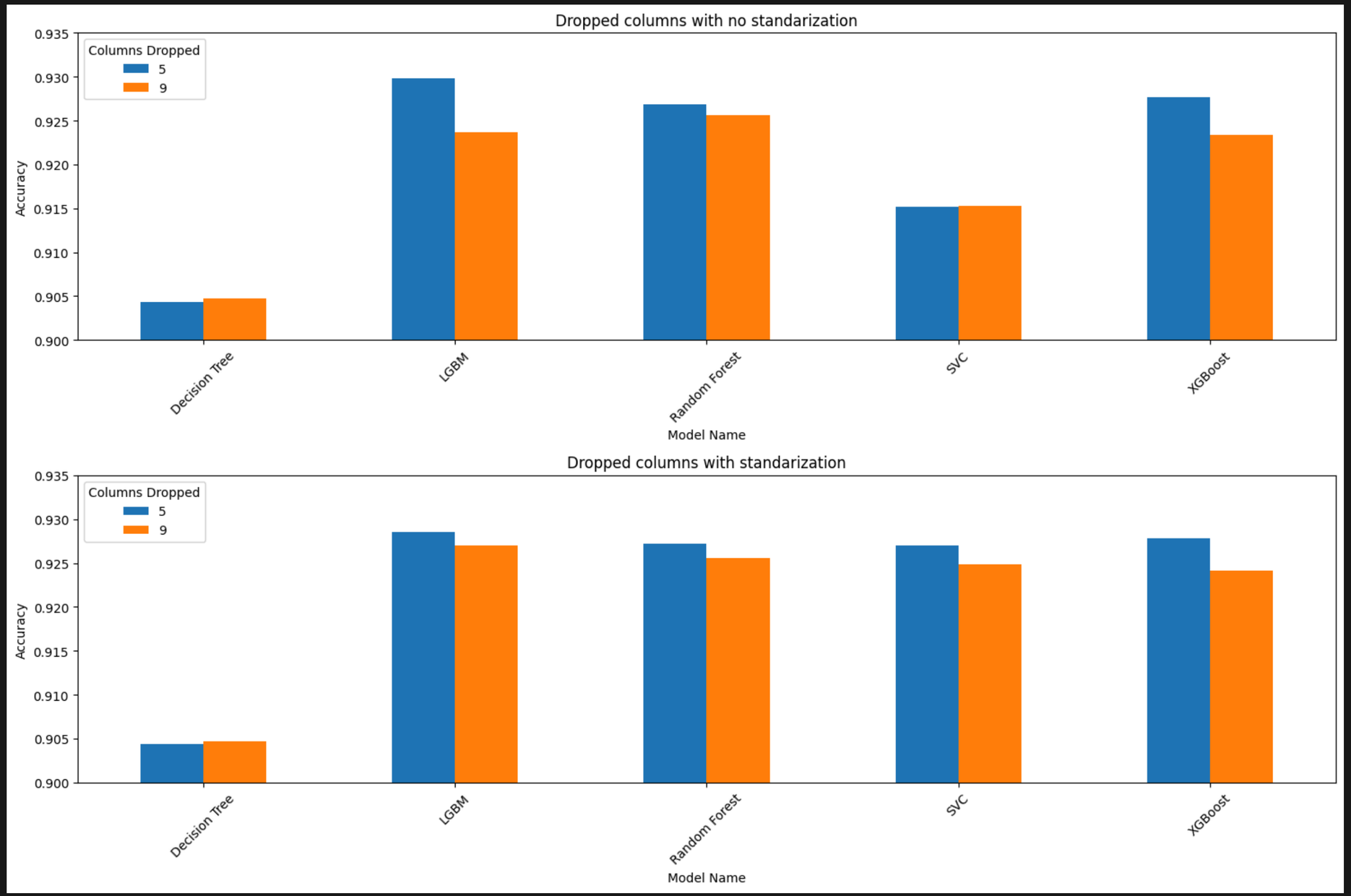

0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
Accuracy
Extent
Solidity
ShapeFactor4
ShapeFactor2
roundness
EquivDiameter
Eccentricity
AspectRation
Area
ConvexArea
ShapeFactor1
ShapeFactor5
MajorAxisLength
MinorAxisLength
ShapeFactor3
Compactness
Perimeter
Number of features


0.93
0.92
0.91
0.90
0.89
0.88
0.87
Accuracy
Extent
Solidity
ShapeFactor4
ShapeFactor2
roundness
EquivDiameter
Eccentricity
AspectRation
Area
ConvexArea
ShapeFactor1
ShapeFactor5
MajorAxisLength
MinorAxisLength
ShapeFactor3
Compactness
Number of features

4. Data preprocessing

- Removing certain columns
- Standarizing the data

```python
def test_models_with_preprocessing(df, models, columns_to_drop, to_standarize=False, cv=5):
    results = []
    X = df.drop(columns=['Class'])
    y = df['Class']
    for model_name, model in models:
        if model_name == 'XGBoost':
            y = LabelEncoder().fit_transform(y)

        for columns in columns_to_drop:
            X_transformed = drop_columns(X, columns)
            if to_standarize:
                for col in X_transformed.columns:
                    X_transformed[col] = standarize(X_transformed, col)

            scores = cross_validate(model, X_transformed, y, cv=cv,
                                    scoring=['accuracy', 'precision_weighted', 'recall_weighted'],
                                    return_train_score=False)
            accuracy = scores['test_accuracy'].mean()
            precision = scores['test_precision_weighted'].mean()
            recall = scores['test_recall_weighted'].mean()
            results.append((model_name, columns, accuracy, precision, recall))

    results = pd.DataFrame(results, columns=['Model Name', 'Columns Dropped', 'Accuracy', 'Precision', 'Recall'])
    results.sort_values('Accuracy', ascending=False, inplace=True)
    results.reset_index(drop=True, inplace=True)
    return results
```


Dropped columns with no standarization
Columns Dropped
5
9
Accuracy
0.935
0.930
0.925
0.920
0.915
0.910
0.905
0.900
Decision Tree
LGBM
Random Forest
SVC
XGBoost
Model Name
Dropped columns with standarization
Columns Dropped
5
9
Accuracy
0.935
0.930
0.925
0.920
0.915
0.910
0.905
0.900
Decision Tree
LGBM
Random Forest
SVC
XGBoost
Model Name

5. Hyperparameter tuning

Looking for best parameters

```python
param_grid_svc = {
    "C": [i for i in range(1, 50)],
    "kernel": ['rbf', 'linear'],
    "gamma": ['scale', 'auto']
}

param_grid_rf = {
    "n_estimators": [i for i in range(10, 150, 10)],
    "max_depth": [i for i in range(1, 8)],
    "min_samples_split": [i for i in range(2, 10)],
    "min_samples_leaf": [i for i in range(1, 10)],
    "criteria": ['gini', 'entropy', 'log_loss'],
    "bootstrap": [True, False]
}

param_grid_lgbm = {
    "boosting_type": ['gbdt', 'dart', 'goss', 'rf'],
    "max_depth": [i for i in range(1, 8)],
    "n_estimators": [i for i in range(10, 150, 10)],
}

param_grid_xgb = {
    "booster" : ['gbtree', 'gblinear'],
    "eta" : [0.05, 0.1, 0.2],
    "max_depth" : [2, 3, 4, 5],
    "min_child_weight" : [0.05, 0.1, 0.5, 1, 3]
}
```

```python
def get_best_params(X, y, model, param_grid, columns_to_drop):
    y = LabelEncoder().fit_transform(y)
    X_transformed = drop_columns(X, columns_to_drop)
    for col in X_transformed.columns:
        X_transformed[col] = standarize(X_transformed, col)
    grid = GridSearchCV(model, param_grid, cv=5, scoring='accuracy')
    grid.fit(X_transformed, y)
    return grid.best_params_
```

6. Model evaluation

Models scores

Model: Random Forest

	precision	recall	f1-score	support
BARBUNYA	0.93	0.87	0.90	181
BOMBAY	1.00	1.00	1.00	77
CALI	0.94	0.94	0.94	249
DERMASON	0.91	0.96	0.93	491
HOROZ	0.96	0.95	0.96	268
SEKER	0.95	0.96	0.96	295
SIRA	0.87	0.83	0.85	345
accuracy			0.92	1906
macro avg	0.94	0.93	0.93	1906
weighted avg	0.92	0.92	0.92	1906

Accuracy: 0.925

Model: LGBM

	precision	recall	f1-score	support
BARBUNYA	0.91	0.89	0.90	181
BOMBAY	1.00	1.00	1.00	77
CALI	0.96	0.94	0.95	249
DERMASON	0.91	0.95	0.93	491
HOROZ	0.95	0.96	0.95	268
SEKER	0.95	0.96	0.95	295
SIRA	0.89	0.84	0.86	345
accuracy			0.93	1906
macro avg	0.94	0.93	0.94	1906
weighted avg	0.93	0.93	0.93	1906

Accuracy: 0.9271

Model: XGBoost

	precision	recall	f1-score	support
0	0.92	0.90	0.91	181
1	0.99	1.00	0.99	77
2	0.96	0.93	0.95	249
3	0.91	0.95	0.93	491
4	0.96	0.96	0.96	268
5	0.95	0.97	0.96	295
6	0.87	0.83	0.85	345
accuracy			0.93	1906
macro avg	0.94	0.93	0.93	1906
weighted avg	0.93	0.93	0.93	1906

Accuracy: 0.926

Model: SVM

	precision	recall	f1-score	support
BARBUNYA	0.95	0.91	0.93	181
BOMBAY	1.00	1.00	1.00	77
CALI	0.95	0.95	0.95	249
DERMASON	0.90	0.95	0.93	491
HOROZ	0.96	0.95	0.95	268
SEKER	0.96	0.97	0.96	295
SIRA	0.89	0.83	0.86	345
accuracy			0.93	1906
macro avg	0.94	0.94	0.94	1906
weighted avg	0.93	0.93	0.93	1906

Accuracy: 0.9302

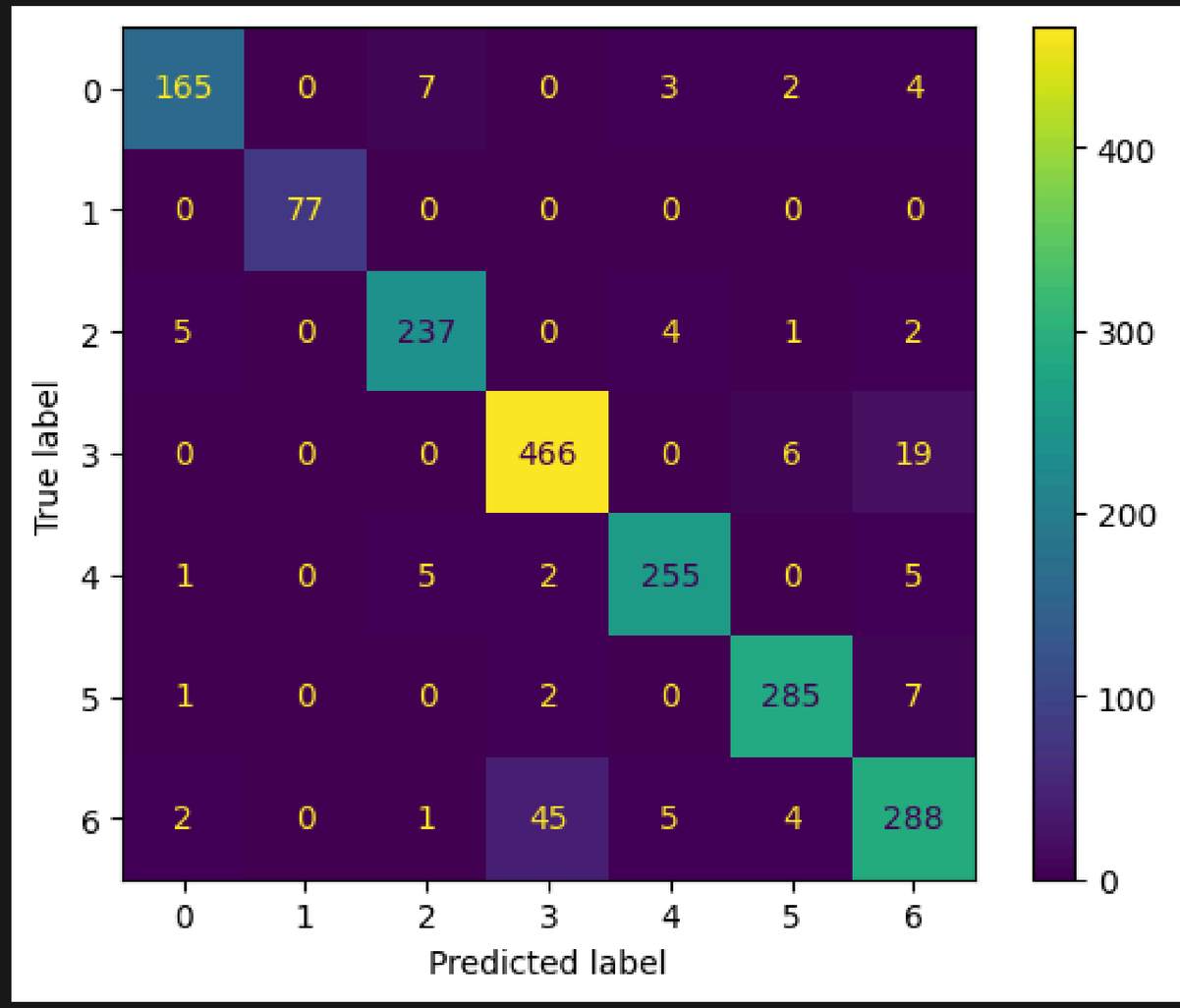
7. Final model selection

Model 1 (tuned SVC)

Model 1:

	precision	recall	f1-score	support
0	0.95	0.91	0.93	181
1	1.00	1.00	1.00	77
2	0.95	0.95	0.95	249
3	0.90	0.95	0.93	491
4	0.96	0.95	0.95	268
5	0.96	0.97	0.96	295
6	0.89	0.83	0.86	345
accuracy			0.93	1906
macro avg	0.94	0.94	0.94	1906
weighted avg	0.93	0.93	0.93	1906

Accuracy: 0.9302

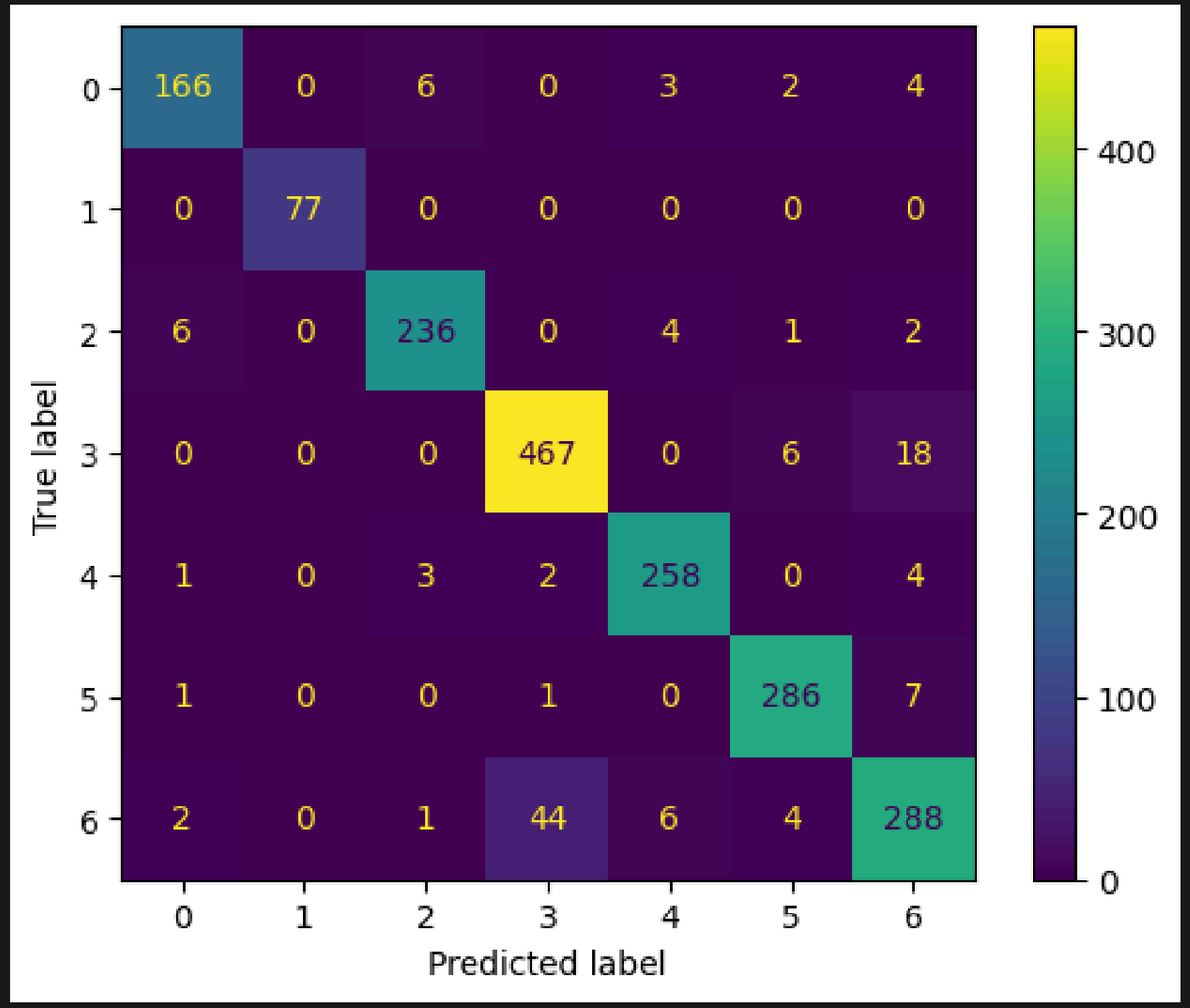


Model 2 (stacked model)

Model 2:

	precision	recall	f1-score	support
0	0.94	0.92	0.93	181
1	1.00	1.00	1.00	77
2	0.96	0.95	0.95	249
3	0.91	0.95	0.93	491
4	0.95	0.96	0.96	268
5	0.96	0.97	0.96	295
6	0.89	0.83	0.86	345
accuracy			0.93	1906
macro avg	0.94	0.94	0.94	1906
weighted avg	0.93	0.93	0.93	1906

Accuracy: 0.9328



Sira

Dermason

Business purposes

THE END