Classification with SVM

Zander Bonnet

Aug 28, 2024

The chosen data set is the dry beans dataset from the UCI machine-learning repository. His dataset is a collection of 13611 observations of 16 features derived from dry bean pictures. I will use this data set to classify the beans based on the data from their photographs.

SVM is a supervised machine-learning algorithm used for clustering a regression. What sets SVM apart from other algorithms is that it utilizes a kernel algorithm to find the optimal hyperplane to separate the data. It can do this by increasing the dimensionality of the data to a point that it can separate the data most optimally. The algorithm first uses the kernel to increase the dimensions of the data and then it finds the optimal separation of the classes.

In SVM the kernel is a function that maps the input data to a higher dimension. The hyperplane is the plane that most effectively separates the classes in the higher dimension space. A decision boundary is the value that the hyperplane separates the classes at in the N-dimensional space.

The maximum margin is the margin at which the hyperplane creates the largest possible distance between groups. The decision vectors are the data points that are the closest to the hyperplane. The maximum margin hyperplane is the hyperplane that best separates two classes.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, KFold, StratifiedKFold, cr
from sklearn.svm import SVC
from sklearn.pipeline  import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix, accuracy_s
```

In [2]:
```
from ucimlrepo import fetch_ucirepo

# fetch dataset
dry_bean = fetch_ucirepo(id=602)

# data (as pandas dataframes)
X = dry_bean.data.features
y = dry_bean.data.targets

# metadata
print(dry_bean.metadata)

# variable information
print(dry_bean.variables)
```

{'uci_id': 602, 'name': 'Dry Bean', 'repository_url': 'https://archive.ics.uci.edu/dataset/602/dry+bean+dataset', 'data_url': 'https://archive.ics.uci.edu/static/public/602/data.csv', 'abstract': 'Images of 13,611 grains of 7 different registered dry beans were taken with a high-resolution camera. A total of 16 features; 12 dimensions and 4 shape forms, were obtained from the grains.', 'area': 'Biology', 'tasks': ['Classification'], 'characteristics': ['Multivariate'], 'num_instances': 13611, 'num_features': 16, 'feature_types': ['Integer', 'Real'], 'demographics': [], 'target_col': ['Class'], 'index_col': None, 'has_missing_values': 'no', 'missing_values_symbol': None, 'year_of_dataset_creation': 2020, 'last_updated': 'Thu Mar 28 2024', 'dataset_doi': '10.24432/C50S4B', 'creators': [], 'intro_paper': {'title': 'Multiclass classification of dry beans using computer vision and machine learning techniques', 'authors': 'M. Koklu, Ilker Ali Özkan', 'published_in': 'Computers and Electronics in Agriculture', 'year': 2020, 'url': 'https://www.semanticscholar.org/paper/e84c31138f2f261d15517d6b6bb8922c3fe597a1', 'doi': '10.1016/j.compag.2020.105507'}, 'additional_info': {'summary': 'Seven different types of dry beans were used in this research, taking into account the features such as form, shape, type, and structure by the market situation. A computer vision system was developed to distinguish seven different registered varieties of dry beans with similar features in order to obtain uniform seed classification. For the classification model, images of 13,611 grains of 7 different registered dry beans were taken with a high-resolution camera. Bean images obtained by computer vision system were subjected to segmentation and feature extraction stages, and a total of 16 features; 12 dimensions and 4 shape forms, were obtained from the grains.', 'purpose': None, 'funded_by': None, 'instances_represent': None, 'recommended_data_splits': None, 'sensitive_data': None, 'preprocessing_description': None, 'variable_info': '1.) Area (A): The area of a bean zone and the number of pixels within its boundaries.\r\n2.) Perimeter (P): Bean circumference is defined as the length of its border.\r\n3.) Major axis length (L): The distance between the ends of the longest line that can be drawn from a bean.\r\n4.) Minor axis length (l): The longest line that can be drawn from the bean while standing perpendicular to the main axis.\r\n5.) Aspect ratio (K): Defines the relationship between L and l.\r\n6.) Eccentricity (Ec): Eccentricity of the ellipse having the same moments as the region.\r\n7.) Convex area (C): Number of pixels in the smallest convex polygon that can contain the area of a bean seed.\r\n8.) Equivalent diameter (Ed): The diameter of a circle having the same area as a bean seed area.\r\n9.) Extent (Ex): The ratio of the pixels in the bounding box to the bean area.\r\n10.)Solidity (S): Also known as convexity. The ratio of the pixels in the convex shell to those found in beans.\r\n11.)Roundness (R): Calculated with the following formula: (4piA)/(P^2)\r\n12.)Compactness (CO): Measures the roundness of an object: Ed/L\r\n13.)ShapeFactor1 (SF1)\r\n14.)ShapeFactor2 (SF2)\r\n15.)ShapeFactor3 (SF3)\r\n16.)ShapeFactor4 (SF4)\r\n17.)Class (Seker, Barbunya, Bombay, Cali, Dermosan, Horoz and Sira)\r\n', 'citation': None}}

	name	role	type	demographic	\
0	Area	Feature	Integer	None	
1	Perimeter	Feature	Continuous	None	
2	MajorAxisLength	Feature	Continuous	None	
3	MinorAxisLength	Feature	Continuous	None	
4	AspectRatio	Feature	Continuous	None	
5	Eccentricity	Feature	Continuous	None	
6	ConvexArea	Feature	Integer	None	
7	EquivDiameter	Feature	Continuous	None	
8	Extent	Feature	Continuous	None	
9	Solidity	Feature	Continuous	None	
10	Roundness	Feature	Continuous	None	
11	Compactness	Feature	Continuous	None	
12	ShapeFactor1	Feature	Continuous	None	
13	ShapeFactor2	Feature	Continuous	None	
14	ShapeFactor3	Feature	Continuous	None	
15	ShapeFactor4	Feature	Continuous	None	
16	Class	Target	Categorical	None	

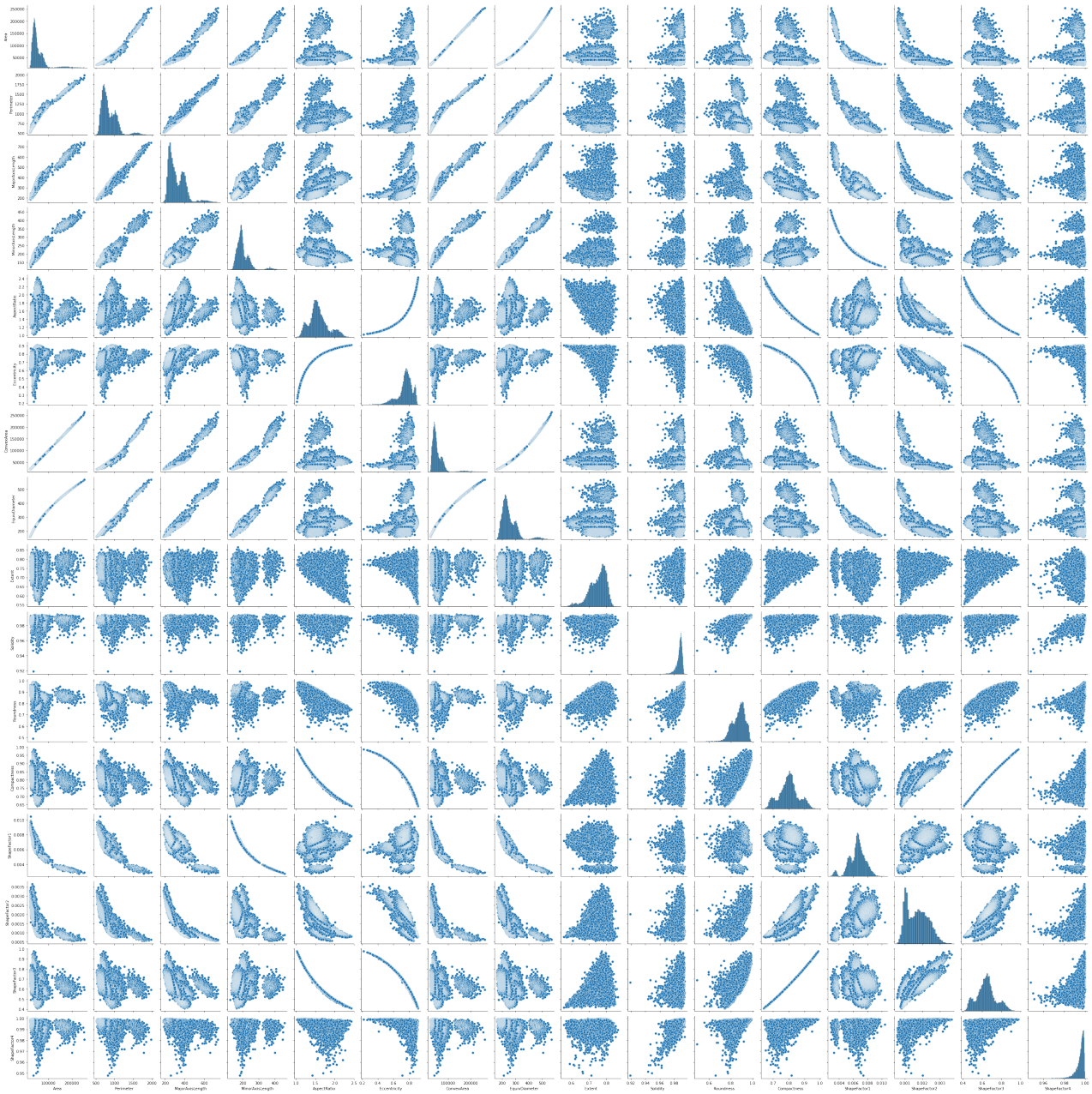
	description	units	missing_values
0	The area of a bean zone and the number of pixe...	pixels	no
1	Bean circumference is defined as the length of...	None	no
2	The distance between the ends of the longest l...	None	no
3	The longest line that can be drawn from the be...	None	no
4	Defines the relationship between MajorAxisLeng...	None	no
5	Eccentricity of the ellipse having the same mo...	None	no
6	Number of pixels in the smallest convex polygo...	None	no
7	Equivalent diameter: The diameter of a circle ...	None	no
8	The ratio of the pixels in the bounding box to...	None	no
9	Also known as convexity. The ratio of the pixe...	None	no
10	Calculated with the following formula: (4piA)/...	None	no
11	Measures the roundness of an object	Ed/L	no
12		None	no
13		None	no
14		None	no
15		None	no
16	(Seker, Barbunya, Bombay, Cali, Dermosan, Horo...	None	no

In [3]:
```
X.head()
```

Out[3]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRatio	Eccentricity	ConvexArea	E
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	28715	
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	29172	
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	29690	
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	30724	
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	30417	

In [4]:
```
sns.pairplot(X)
plt.show()
```



No unusual features and no missing data.

How accurately can we predict the class of a dry bean based on its photograph?

Will some classes be too similar to tell apart?

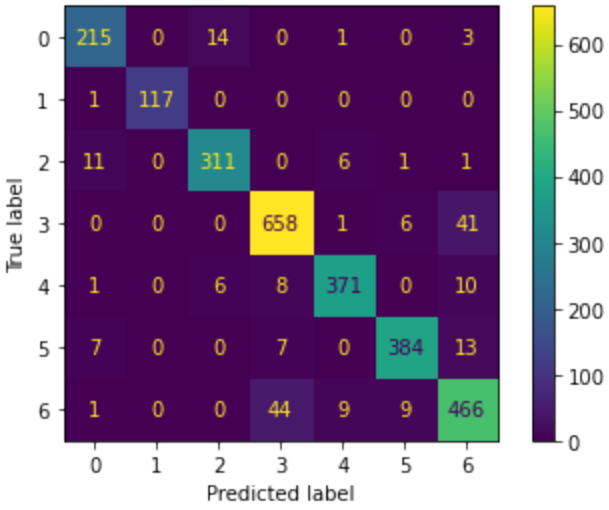
In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size= .2, random_st
```

In [6]:
```
svm = make_pipeline(StandardScaler(),SVC(kernel='linear'))
svm.fit(X_train, y_train.values.ravel())
```

Out[6]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('svc', SVC(kernel='linear'))])
```

In [7]:
```
pred = svm.predict(X_test)
```

In [8]:
```
cm = confusion_matrix(y_test, pred)
ConfusionMatrixDisplay(confusion_matrix = cm).plot()
plt.show()
```



In [9]:
```
accuracy_score(y_test, pred)
```

Out[9]:
```
0.9261843554902681
```

In [10]:
```
kf = KFold(n_splits=5, shuffle=True, random_state=100)
score = cross_val_score(make_pipeline(StandardScaler(),SVC(kernel='linear')),
                        X, y.values.ravel(), cv= kf, scoring="accuracy")
print(f'Scores for each fold are: {score}')
print(f'Average score: {"{:.2f}".format(score.mean())}')
```

```
Scores for each fold are: [0.92691884 0.9283615  0.92799412 0.93277002 0.9162380
6]
Average score: 0.93
```

In my analysis, we can see that not only is the model very effective in predicting the classes of the dry bean based on the data from the photos, but it is also very accurate. In the confusion matrix, we can see that almost all groups have distinct differentiation, except for groups 3 and 6. These two groups seem to have a small level of overlap between them causing some false classification. The accuracy score of about 92% shows the model is performing very well. We cannot perform an ROC-AUC curve as there is more than one class, but the cross-validation metrics show that the model performs well across multiple subsets of the data. Showing that no data is visibly creating bias in the model.

Reference

Dry Bean. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C50S4B.