Supervised Learning

Dry Beans Dataset

Unidade Curricular: Inteligência Artificial

Grupo 36:
Diogo Filipe de Oliveira Santos
Jéssica Mireie Fernandes do Nascimento
João Vítor Freitas Fernandes

Créditos: KOKLU, M. and OZKAN, I.A., (2020), â€œMulticlass Classification of Dry Beans Using Computer Vision and Machine Learning Techniques.â€ Computers and Electronics in Agriculture, 174, 105507.

U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Descrição do problema

O objetivo deste trabalho é utilizar *Supervised Learning* para prever o tipo de *dry bean*. Para isso, contamos com um *dataset* de 13611 amostras.

Estas amostras contêm os seguintes atributos:

- Area
- Perimeter
- MajorAxisLength
- MinorAxisLength
- AspectRation
- Eccentricity
- ConvexArea
- EquivDiameter
- Extent
- Solidity
- Roundness
- Compactness
- ShapeFactor1
- ShapeFactor2
- ShapeFactor3
- ShapeFactor4

Este dataset é constituído por 7 tipos de feijões: *Seker*, *Barbunya*, *Bombay*, *Cali*, *Horoz*, *Sira*, *Dermason*.

Tirando partido dos diferentes valores destes atributos, utilizaremos vários classificadores para avaliar a possível classe de cada feijão, com uma taxa de acerto aceitável.

Ferramentas e Algoritmos utilizados

Para desenvolver este projeto utilizou-se a linguagem de programação *Python3* com o auxilio do *Jupyter Notebook* e das seguintes *libraries*: *Pandas, Numpy, Scipy, Scikit-Learn, Matplotlib* e *Seaborn*.

Durante um pré processamento dos dados foram removidos outliers, duplicados e valores nulos. Decidimos utilizar um sampling de 450 dados de cada classe, uma vez que o número total de dados de cada classe, depois desta “limpeza de dados”, era variável e o menor continha 500 dados (classe: *Bombay*).

Foram implementadas 5 tecnicas de classificação:

- Decision Tree
- Nearest Neighbor
- Naïve Bayes
- Support Vector Machines
- Neural Networks

Para todas estas tecnicas foi efetuado o grid-search, com o objetivo de obter as melhores parametrizações possíveis para os modelos neste contexto.

Decision Tree

Funções utilizadas

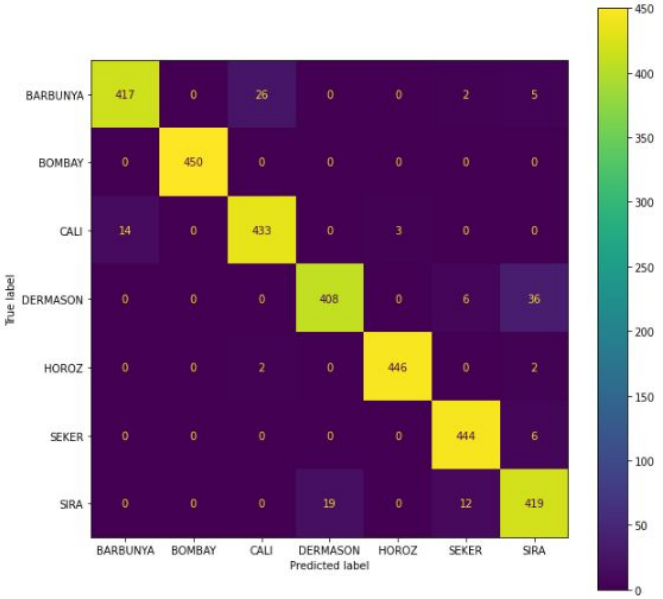
```
decision_tree_classifier = DecisionTreeClassifier()

parameter_grid = {'criterion': ['gini', 'entropy'],
                  'splitter': ['best', 'random'],
                  'max_depth': [1, 2, 3, 4, 5, 6, 7, 8],
                  'max_features': [1, 2, 3, 4, 5, 6, 7, 8]}

cross_validation = StratifiedKFold(n_splits=10)

grid_search=gridSearchScore(decision_tree_classifier, parameter_grid,cross_validation, labels, inputs)
```

```
Best score: 0.933968253968254
Best parameters: {'criterion': 'entropy', 'max_depth': 7, 'max_features': 7, 'splitter': 'best'}
```

Métricas

```
Accuracy:  0.9577777777777777
Precision:  0.9582567956720535
Recall:  0.9577777777777777
F1:  0.9577601677119991
```

Confusion Matrix



Nearest Neighbor (K-NN)

Funções utilizadas

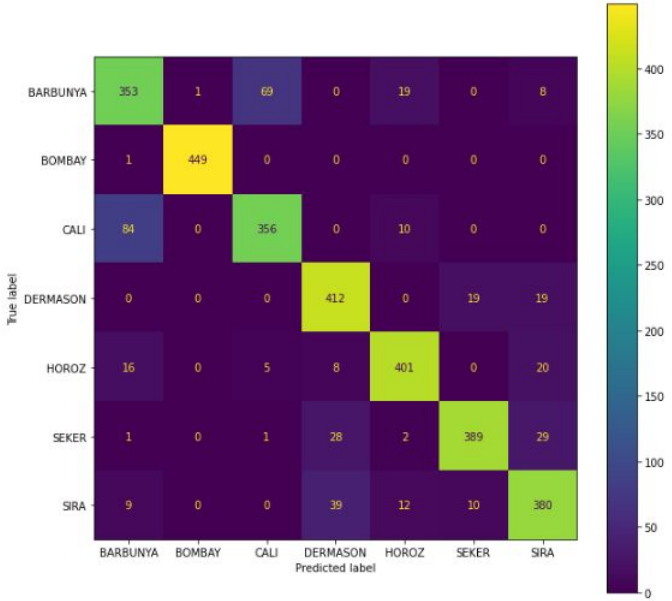
```
parameter_grid = {"n_neighbors": [1, 2, 3, 4, 5, 6, 7, 8],
                  "weights": ['uniform'],
                  "algorithm": ['auto', 'ball_tree', 'kd_tree', 'brute'],
                  "p": [1, 2, 3]}

cross_validation = StratifiedKFold(n_splits=10)

knn = neighbors.KNeighborsClassifier()

grid_search=gridSearchScore(knn,parameter_grid,cross_validation,labels,inputs)
```

Best score: 0.7498412698412699
Best parameters: {'algorithm': 'auto', 'n_neighbors': 3, 'p': 1, 'weights': 'uniform'}

Métricas

Accuracy: 0.8698412698412699
Precision: 0.8710920154303912
Recall: 0.8698412698412699
F1: 0.8700055321025822

Confusion Matrix



Naïve Bayes (NB)

Funções utilizadas

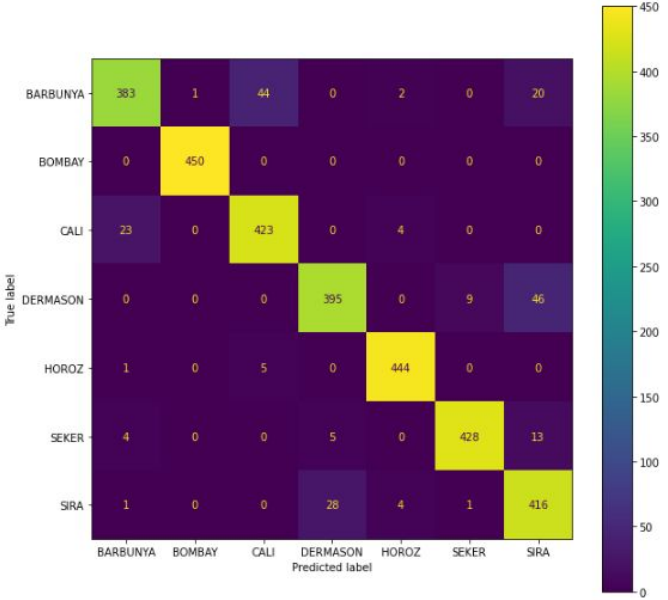
```
parameter_grid = {"var_smoothing": [1e-9,1e-10,1e-11,1e-12,1e-13,1e-14,1e-15,1e-16,1e-17,1e-18]}

cross_validation = StratifiedKFold(n_splits=10)

gnb = GaussianNB()

grid_search=gridSearchScore(gnb,parameter_grid,cross_validation,labels,inputs)
```

Best score: 0.9311111111111112
Best parameters: {'var_smoothing': 1e-15}

Métricas

Accuracy: 0.933015873015873
Precision: 0.9345749358546472
Recall: 0.933015873015873
F1: 0.9330788643427723

Confusion Matrix



Support Vector Machines (SVM)

Funções utilizadas

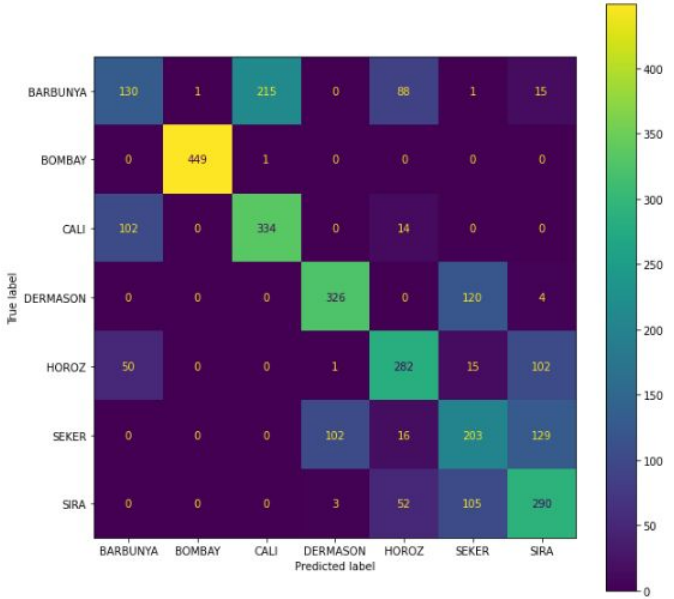
```
parameter_grid = {"kernel": ['rbf'],#poly
                  "degree": [1, 2, 3, 4, 5],
                  "gamma": ['auto', 'scale']}

cross_validation = StratifiedKFold(n_splits=10)

svm = SVC()

grid_search=gridSearchScore(svm,parameter_grid,cross_validation,labels,inputs)
```

```
Best score: 0.6387301587301587
Best parameters: {'degree': 1, 'gamma': 'scale', 'kernel': 'rbf'}
```

Métricas

```
Accuracy:  0.6393650793650794
Precision:  0.6341158702907902
Recall:  0.6393650793650794
F1:  0.632210358533379
```

Confusion Matrix



Neural Networks (ANN)

Funções utilizadas

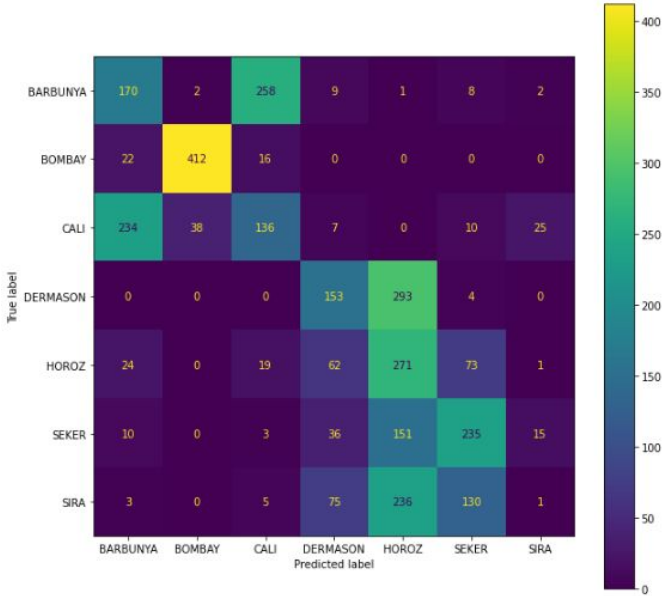
```
parameter_grid = {"solver": ['adam'],
                  "hidden_layer_sizes": [8, 9, 10, 11],
                  "alpha": [1e-8, 1e-9, 1e-10, 1e-11],
                  "max_iter": [10000],
                  "random_state": [1]}

cross_validation = StratifiedKFold(n_splits=10)

mlp = MLPClassifier()

grid_search=gridSearchScore(mlp,parameter_grid,cross_validation,labels,inputs)
```

```
Best score: 0.4238095238095238
Best parameters: {'alpha': 1e-09, 'hidden_layer_sizes': 11, 'max_iter': 10000, 'random_state': 1, 'solver':
'adam'}
```

Métricas

```
Accuracy:  0.43746031746031744
Precision:  0.40793099861678234
Recall:  0.43746031746031744
F1:  0.41229459142695535
```

Confusion Matrix



Conclusão

Com este trabalho concluímos que a fim de implementarmos Supervised Learning, conseguimos perceber que é necessário passar-se por várias etapas para obter um modelo aceitável. Nestas etapas inclui-se análise dos dados e um pré processamento dos mesmos, incluindo uma “limpeza” e balanceamento.

No que toca à avaliação da performance de cada modelo, utilizamos as várias tecnicas anteriormente apresentadas e podemos verificar que a Decision Tree e Naïve Bayes foram os classificadores que obtiveram melhores resultados com valores acima dos 90%.