Dry Bean Classification

Problem Definition

What is the problem about?

Dry bean is the most popular pulse produced in the world. The main problem dry bean producers and marketers face is in ascertaining good seed quality. Lower quality of seeds leads to lower quality of produce. Seed quality is the key to bean cultivation in terms of yield and disease. Manual classification and sorting of bean seeds is a difficult process. Our objective is to use Machine learning techniques to do the automatic classification of seeds.

Why is this problem important to solve?

Ascertaining seed quality is important for producers and marketers. Doing this manually would require a lot of effort and is a difficult process. This is why we try to use machine learning techniques to do the automatic classification of seeds.

Business/Real-World impact of solving this problem?

- Saves hours of manual sorting and classification of seeds.
- We can do it in real-time.

Dataset

Source of dataset

The dataset is downloaded from the UCI Machine learning repository.
Link: https://archive.ics.uci.edu/ml/datasets/Dry+Bean+Dataset

Explanation of each feature

There are a total of 16 features - 12 dimensional and 4 shape features

1. Area (A): The area of a bean zone and the number of pixels within its boundaries.

   $$A = \sum_{r,c \in R} 1$$
2. Perimeter (P): Bean circumference is defined as the length of its border.
3. Major axis length (L): The distance between the ends of the longest line that can be drawn from a bean.

4. Minor axis length (l): The longest line that can be drawn from the bean while standing perpendicular to the main axis.
5. Aspect ratio (K): Defines the relationship between L and l.
   $K = \frac{L}{l}$
6. Eccentricity (Ec): Eccentricity of the ellipse having the same moments as the region.
7. Convex area (C): Number of pixels in the smallest convex polygon that can contain the area of a bean seed.
8. Equivalent diameter (Ed): The diameter of a circle having the same area as a bean seed area.
   $d = \sqrt{\frac{4 * A}{\pi}}$
9. Extent (Ex): The ratio of the pixels in the bounding box to the bean area.
   $Ex = \frac{A}{B}$ B = Area of bounding rectangle
10. Solidity (S): Also known as convexity. The ratio of the pixels in the convex shell to those found in beans
    $S = \frac{A}{C}$
11. Roundness (R): Roundness is the measure of how closely the shape of an object approaches that of a mathematically perfect circle. Calculated with the following formula:
    $R = \frac{4\pi A}{P^2}$
12. Compactness (CO): Measures the roundness of an object. The formula is:
    $CO = \frac{Ed}{L}$

The shape features are:

1. $ShapeFactor1(SF1) = \frac{L}{A}$
2. $ShapeFactor2(SF2) = \frac{l}{A}$
3. $ShapeFactor3(SF3) = \frac{A}{\frac{L}{2} * \frac{L}{2} * \pi}$
4. $ShapeFactor4(SF4) = \frac{A}{\frac{L}{2} * \frac{l}{2} * \pi}$

Dataset size and challenges

1. The dataset is in the form of an excel sheet with 13,611 rows, each referring to one example of a seed with 16 features.
2. The last column of the sheet is “Class” which has 7 unique classes: ***Barbunya, Bombay, Cali, Dermason, Horoz, Seker and Sira***.
3. The dataset is imbalanced, “BOMBAY” class has only 522 examples whereas “DERMASON” has 3546 examples. So, we need to deal with this imbalance.

Tools to process the data

I will use ‘Pandas’ to process the data. There’s also ‘Dask’, but Pandas should be good enough to handle 13k rows in the excel sheet

Data Acquisition

1. Data is openly available and collected from the UCI Machine learning repository
2. Other than primary source no more data can be acquired as it has been acquired through a specialized camera system only available with the authors of the paper.

Key Performance Indicator *(KPI)*

Business Metric Definition

1. We will be using *Confusion Matrix* and using it for calculating per-class *F1-score.*
2. We will average the calculated *F1-score* for all the 7 classes.

Why is the metric used?

Since the dataset is imbalanced, if we simply use a metric like accuracy, it will give us a very skewed idea of the performance of the model. This is why we need metrics that are robust to imbalanced data.

Alternative metrics that can be used?

Accuracy is another metric we can use but that is not going to help us much. We have fewer examples of BOMBAY class which constitutes about 3.8% of the dataset. So, let's say our model doesn't predict the BOMBAY class at all, that would still give us an accuracy close to 96%

Pros and cons of metric used?

F1 - score

Pros :

1. Takes data distribution into account, so it's useful in-case

Cons :

1. Less interpretable. Precision and recall are more interpretable than the f1-score since it measures the type-1 error and type-2 error. However, f1-score measures the trade-off between this two.
2. When positive class is minority class, the score is quite sensitive when there is switching where the ground truth is positive.

We will also using other metrics: Recall, Precision, Sensitivity, Balanced Accuracy, ROC - AUC inorder to mitigate the cons of F1-score.

Implementation of F1-score from scratch

```python
import numpy as np
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split


def get_basic_metrics(y_true: np.array, y_pred: np.array):

    Tp = sum((y_true == 1) & (y_pred == 1))
    Fp = sum((y_true == 1) & (y_pred == 0))
    Fn = sum((y_true == 0) & (y_pred == 1))
    Tn = sum((y_true == 0) & (y_pred == 0))

    return {
        "True_positive": Tp,
        "False_positive": Fp,
        "False_negative": Fn,
        "True_negative": Tn
    }


def f1_score(y_pred: np.array, y_true: np.array):
    metrics = get_basic_metrics(y_true=y_true, y_pred=y_pred)

    recall = metrics['True_positive'] / (metrics['True_positive'] + metrics['False_negative'])
    precision = metrics['True_positive'] / (metrics['True_positive'] + metrics['False_positive'])

    f1 = (2 * precision * recall) / (precision + recall)

    return f1

X, y = make_classification()
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.33, random_state=42
)

lr = LogisticRegression()
lr.fit(X_train, y_train)

y_pred = lr.predict(X_test)

print("F1-score is: {f1_score(y_pred=y_pred, y_true=y_test)}")
```

Real world challenges and constraints

Minimum requirements of the system

The minimum requirements for our system is that:

1. Highly accurate
2. Must be fast

How are similar problems solved in literature ?

1. KOKLU, M. and OZKAN, I.A., (2020) apply SVM, Decision Tree, MLP and KNN on the above 16 features and produce a detailed analysis of the results on each model in their paper
2. Słowiński, Grzegorz in his paper “Dry Beans Classification Using Machine Learning.” trains an ANN, Multinomial Naive Bayes, SVM, Decision Tree, Random Forest and Voting Classifier on the 16 features and provides a detailed analysis of the results.

Mapping into a ML problem

1. Our problem is a multiclass classification problem, with 7 classes
2. We have 16 features, which we already discussed.

Solution approaches

1. We will use the provided features, and train classical machine learning models like SVM, Knn, DT and analyse their performance
2. We will also apply deep learning techniques like ANN to the features and analyse the performance

References

1. KOKLU, 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.
2. Słowiński, Grzegorz. "Dry Beans Classification Using Machine Learning." Proceedings http://ceur-ws. org ISSN 1613 (2021): 0073.
3. https://datascience.stackexchange.com/questions/65341/f1-score-vs-accuracy-which-metric-is-more-important
4. https://www.educative.io/edpresso/what-is-the-f1-score
5. https://www.datascienceblog.net/post/machine-learning/specificity-vs-precision/

6. https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/