Project :- Adopt a Pet Project

Author :- Sartaj Ahmad Hajam

Approach & Outlines:

The given dataset that is used to predict whether a pet is going to be adopted during the next 30 days or not, contains a wide variety of data types:

- **Structured Data**:
    - **Numerical Variables**: 'Age', 'Fee'
    - **Categorical Variables**: 'Type', 'Gender', 'Breed', 'Color1', 'Color2', 'Color3', 'MaturitySize', 'FurLength', 'Vaccinated', 'Dewormed', 'Sterilized', 'Health'
- **Unstructured Data:**
    - **Text**: 'Description'
    - **Image**: 'Images'

Thus, we need to perform feature extraction that efficiently extracts useful features for the training process. This step is implemented in the pipeline using different techniques for each type of variables.

ML Scikit-Learn Pipeline:

The machine learning pipeline used in this project is composed of 3 main steps:

1. Data Transformation (Feature extraction)
2. Dimensionality Reduction (Feature Selection)
3. Training (Making prediction)

The first step in this pipeline, which is Data Transformation, is conducted using 4 preprocessors to extract features from different types of the available data:

1. One Hot Encoder (For Categorical Variables)
2. Standard Scaler (For Numerical Variables)
3. TF-IDF Vectorizer (For Textual Variable)
4. BOF Extractor (For Image Variable)

The second step of the pipeline, which is Dimensionality Reduction, is performed using Principal Component Analysis or PCA. This technique allows us to reduce the number of features used in the training step, without losing much information.

Hyperparameters Tuning:

The process of Hyperparameter tuning is known to be time consuming, because a large number of models needs to be tested over the hyperparameter space. In addition, the BOF Extractor, which is used to extract features from the image data, has also a huge time cost (1 hour to perform over the whole training set).

For these reasons the Hyperparameter tuning approach that I implemented, uses a fraction of the training dataset. By using this approach, we can set a bigger hyperparameter space to search on, therefore we have higher probability to find the optimal set of hyperparameters.

Hyperparameter space:

Hyperparameter		Variable Name in Notebook	Range
Number of Image Features		nb_img_features	[3,5,10,15,20]
Number of PCA Vectors		n_components	[50,100,200,500,1000,2000]
Logistic Regression	Solver	solver	[‘saga’,’lbfgs’]
	L1 Ratio for Saga Solver	l1_ratio	[0,0.5,1]
	Regularization Coef.	C	[1,0.9,0.8]
Gradient Boosting	Learning Rate	learning_rate	[0.1,0.01]
	Number of Estimators	n_estimators	[100,150]
	Maximum Depth	max_depth	[3,5,7,10]

Hyperparameter Tuning Process:

The process of hyperparameter tuning is performed using the ‘HalvingGridSearchCV’ class from Scikit-learn on 3000 instances of the training set, which represent around 36% of the training data.

The ‘HalvingGridSearchCV’ class performs a modified version of the traditional exhaustive grid search over the whole space of hyperparameters, where the search is performed iteratively, and in each iteration the number of training data used increases while the number of models to be fitted decreases by a certain factor compared to the previous iteration.

This method is more time efficient, however it is still in the experimental module of scikit-learn for now.

The HalvingGridSearchCV class I used to perform hyperparameter tuning is the following:

```
gs = HalvingGridSearchCV(model, param_dic, cv=2, scoring='accuracy', verbose=2)
```

Where:

- ‘param_dic’ is the dictionary containing the different hyperparameters.
- ‘model’ is the pipeline object.
- ‘cv’ is the parameter for the number of cross-validation folds.

Results:

The Halving Grid Search process took about 9 hours to finish, and the set of optimal hyperparametrs is the following:

Hyperparameter	Best value
Number of Image Features	20
Number of PCA Vectors	100
Learning Algorithm	Gradient Boosting
Learning Rate	0.01
Number of Estimators	150
Maximum Depth	7

The training and test accuracy obtained by training a pipeline using the optimal set of hyperparameters, over the whole training dataset are the following:

Set	Accuracy
Train	0.844
Test	0.608