--------------------------------------------------------------------------------------------------------------

Open Source SW Contribution Project Document

--------------------------------------------------------------------------------------------------------------

Class **EasyCombination**

This class and methods automatically generate a combination of various sclaers and encoders, and calculates the scores according to each combination.

Contain Methods: encodeAndSplit, Scale, Estimate, printAllResult, findBestScore, findWorstScore

WARNING: encodeAndSplit, Scale, Estimate must be executed first before printAllResult, findBestScore, findWorstScore

***Constructor***

**\_\_init\_\_:**

\_\_init\_\_(self, dataset)

**Implementation Note:**

After the dataset is loaded, this constructor can be called to use the class.

**Parameters:**

*dataset*: Dataset you will use

***Method Details***

|  |  |
| --- | --- |
| Method | Description |
| encodeAndSplit(self, target\_name) | **Parameters**: self, target\_name  You have to deliver target\_name to estimate.  **Returns**: void  **Description**:  Encodes dataset with One-Hot, Label encoder.  Then split dataset into train and test.  Recommend call the *scale* method after this method call  **Examples**:  >>>import opensource  >>>os = opensource.EasyCombination(newDatset) >>>os.encodeAndSplit('IsBadBuy') |
| scale(self) | **Parameters**: self  **Returns**: void  **Description**:  Executes scaling: standard, MinMax, MaxAbs, Robust  Recommend call the *estimate* method after this method call  **Examples**:  >>>import opensource  >>>os = opensource.EasyCombination(newDatset) >>>os.encodeAndSplit('IsBadBuy')  >>>os.scale() |
| estimate(self, option) | **Parameters**: self, option  # option 1: DecisionTree # option 2: KNN # option 3: Random Forest # option others: Error  **Returns**: void  **Description**:  Calculates estimation scores for each combination:  #Label Encoding + Standard Scale  #Label Encoding + MinMax Scale  #Label Encoding + MaxAbs Scale  #Label Encoding + Robust Scale  # OneHot Encoding + Standard Scale  # OneHot Encoding + MinMax Scale  # OneHot Encoding + MaxAbs Scale  # OneHot Encoding + Robust Scale  **Examples**:  >>>import opensource  >>>os = opensource.EasyCombination(newDatset) >>>os.encodeAndSplit('IsBadBuy')  >>>os.scale()  >>>os.estimate(1) # option 1: DecisionTree  Selected Random Forest  Score using Label Encoding and Standard Scaling: 0.9009181855557078  Score using Label Encoding and MinMax Scaling: 0.9002786533278516  Score using Label Encoding and Robust Scaling: 0.8998675254670869  Score using One-Hot and Standard Scaling: 0.9005527385683614  Score using One-Hot Encoding and MinMax Scaling: 0.9008268238088712  Score using One-Hot Encoding and Maxabs Scaling: 0.9007354620620346  Score using One-Hot Encoding and Robust Scaling: 0.9005984194417798 |
| printAllResult(self) | **Parameters**: self  **Returns**: void  **Description**:  Prints all the results of the estimate method.  **Examples**:  >>>import opensource  >>>os = opensource.EasyCombination(newDatset) >>>os.encodeAndSplit('IsBadBuy')  >>>os.scale()  >>>os.estimate(1) # option 1: DecisionTree  >>>os.printAllResult()  Selected Random Forest  Score using Label Encoding and Standard Scaling: 0.9009181855557078  Score using Label Encoding and MinMax Scaling: 0.9002786533278516  Score using Label Encoding and Robust Scaling: 0.8998675254670869  Score using One-Hot and Standard Scaling: 0.9005527385683614  Score using One-Hot Encoding and MinMax Scaling: 0.9008268238088712  Score using One-Hot Encoding and Maxabs Scaling: 0.9007354620620346  Score using One-Hot Encoding and Robust Scaling: 0.9005984194417798  Best result score is: 0.9009181855557078  Using Label Encoding and Standard Scaling  Worst result score is: 0.8998675254670869  Using Label Encoding and Robust Scaling |
| findBestScore(self) | **Parameters**: self  **Returns**: void  **Description**:  Outputs the highest score of the estimation  **Examples**:  >>>import opensource  >>>os = opensource.EasyCombination(newDatset) >>>os.encodeAndSplit('IsBadBuy')  >>>os.scale()  >>>os.estimate(1) # option 1: DecisionTree  >>>os.findBestScore()  Best result score is: 0.9009181855557078  Using Label Encoding and Standard Scaling |
| findWorstScore(self) | **Parameters**: self  **Returns**: void  **Description**:  Outputs the Lowest score of the estimation  **Examples**:  >>>import opensource  >>>os = opensource.EasyCombination(newDatset) >>>os.encodeAndSplit('IsBadBuy')  >>>os.scale()  >>>os.estimate(1) # option 1: DecisionTree  >>>os.findWorstScore()  Worst result score is: 0.8998675254670869  Using Label Encoding and Robust Scaling |