Milky Way Formation History project

***Classification Part Report**

AmirHosein Masoudnezhad

1. Introduction

In this part of project, I tried to build a classifier to classify variable stars, mainly LPVs. I used $OGLE(III, IV)$ survey datasets here. These datasets include 9 different class of variable stars' light curves in LMC and SMC.

$OGLE-III$		$OGLE-IV$	
LMC	SMC	LMC	SMC
Anomalous Cepheids $(ACEP)$	Classical Cepheids	Anomalous Cepheids	Anomalous Cepheids
Classical Cepheids (CEP)	Eclipsing and Ellipsoidal Binary Systems	Classical Cepheids	Classical Cepheids
Double-Period Variables (DPV)	Long Period Variables	Eclipsing and Ellipsoidal Binary Systems	Eclipsing and Ellipsoidal Binary Systems
Eclipsing and Ellipsoidal Binary Systems (ECL)	RR-Lyrae	RR-Lyrae	RR-Lyrae
Long Period Variables (LPV)	Type-II Cepheids	Type-II Cepheids	Type-II Cepheids
R-CrB (RCB)			
RR-Lyrae $(RRLYR)$			
Type-II Cepheids $(T2CEP)$			
δ-Scuti $(DSCT)$			

2. Data

I concatenated each class data in a dataframe of this format:

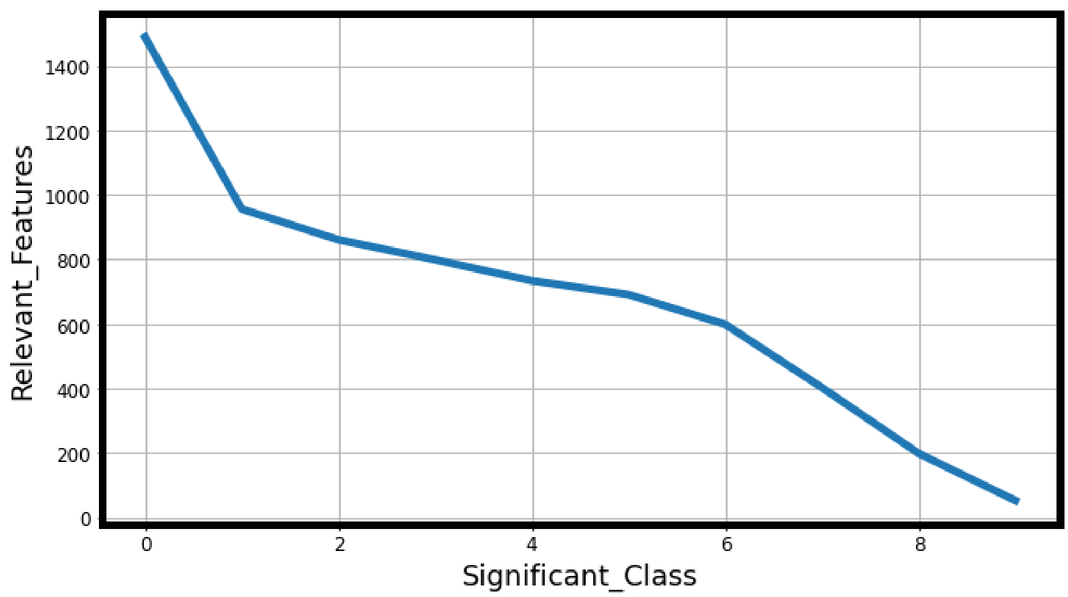
	Time*	value	id	kind	class
0	2088.91535	14.977	III-OGLE-SMC-LPV-00001	I	LPV
1	2090.87535	14.978	III-OGLE-SMC-LPV-00001	I	LPV
2	2103.92958	14.980	III-OGLE-SMC-LPV-00001	I	LPV
3	2104.94812	14.986	III-OGLE-SMC-LPV-00001	I	LPV
4	2106.84596	14.998	III-OGLE-SMC-LPV-00001	I	LPV

Columns' information

Column Name	unit	description
Time	Days	Heliocentric Julian Day - 2450000
Value	Mag	Magnitude
ID	-	Star's OGLE Series - Name
Kind	-	Observation Filter
Class	-	Stars' Classes

3. Feature extraction

I used *tsfresh* library to extract features from the data. It gives us about 1575 features per star. It would be a large number of data, which is too hard to work with. Accordingly, I decided to select optimum relevant features with small sample of data. Finally, the extracted features per star became about 50 features, which have significant effects on classifying all 9 classes.



4. Preprocessing

After all, I concatenate all collected data together and also encoded the output with *One-Hot Encoding* and *LabelEncoder*. Afterwards, I imputed missing data by *iterative_imputer* and shuffle them.

The main problem was different object classes were too imbalance to train a model.

Class	*LPV*	*ECL*	*RRLYR*	*CEP*	*DSCT*	*T2CEP*	*ACEP*	*DPV*	*RCB*
Number	111379	78343	74935	17582	2788	586	349	137	23

In order to deal with this problem, I defined 7 distinct problems to check the model performance on them. I used KNN-Classifier in this part.

Problem	Classification Classes	No. of Classes	Precision	Recall	F1_Score
1.1	$All\ Classes:$ $\{ACEP, CEP, T2CEP, DPV,$ $DSCT, ECL, LPV, RCB, RRLYR\}$	9	0.68	0.49	0.51
1.2	$\{CEP, DPV, DSCT,$ $ECL, LPV, RCB, RRLYR\}$	7	0.70	0.62	0.64
2	$Binary\ Clf.: \begin{cases} 1: & LPV \\ 0: & O.W \end{cases}$	2	0.99	0.99	0.99
3.1	$\{LPV, ECL, RRLYR, CEP, Other\}$ $Other: \{DPV, RCB, DSCT\}$	5	0.87	0.81	0.83
3.2	$\{LPV, ECL, CEP,$ $RRLYR, DSCT, Other\}$ $Other: \{DPV, RCB\}$	6	0.80	0.69	0.72
4.1	$\{LPV, ECL, RRLYR, CEP\}$	4	0.91	0.90	0.90
4.2	$\{LPV, ECL, RRLYR, CEP, DSCT\}$	5	0.85	0.81	0.82

The best performance was on problem 2, which is a binary classification. On the other hand, I was inclined to classify the most number of classes with acceptable score. According to the table above, I decided to solve problem 3.1. It is a multi-class problem in which I put the DPV, RCB and $DSCT$ classes in one class and entitled to '$Other$' class.

5. Fitting models

Finally, I tried to solve the classification problem using KNN, Decision Tree Classifier and LDA, which their results are shown below.

Model Name	Tuned Hyper-parameters	Precision	Recall	F1_Score	Best Score
K-Nearest Neighbors ($\boldsymbol{KNN}$)	$algorithm$: "kd_tree" $weights$: "$distance$"	0.87	0.81	0.83	92%
Decision Tree Classifier	$criterion$: "entropy" $splitter$: "$best$"	0.86	0.87	0.87	92%
Linear Discriminant Analysis ($\boldsymbol{LDA}$)	$shrinkage$: "auto" $solver$: "lsqr"	0.72	0.67	0.68	83%

- **Best Model :** Decision Tree Classifier

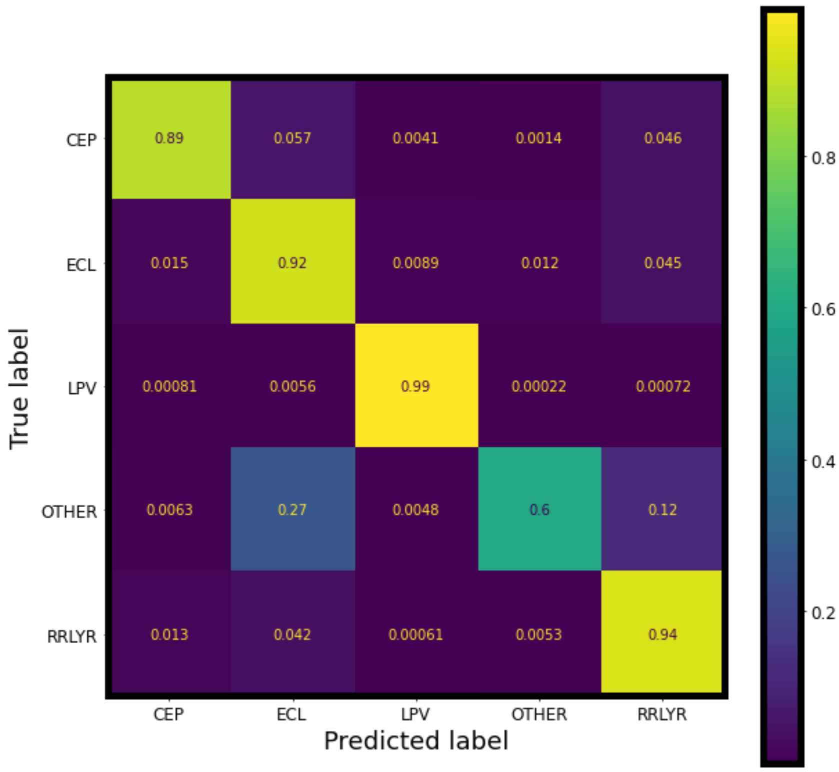
```
In [ ]: # model evaluation

        clf = DTC(criterion='entropy',splitter='best',max_features=None)
        model_reports(clf)
```

```
              precision    recall  f1-score   support

         CEP       0.88      0.89      0.89      3640
         ECL       0.93      0.92      0.92     15802
         LPV       0.99      0.99      0.99     22288
       OTHER       0.58      0.60      0.59       630
       RRLYR       0.93      0.94      0.94     14865

    accuracy                           0.95     57225
   macro avg       0.86      0.87      0.87     57225
weighted avg       0.95      0.95      0.95     57225
```



6. Saving model and Classifier function

At the end, I saved the best model (Decision Tree Classifier) and built a classifier function which can be found on the Github repository as a separate file.