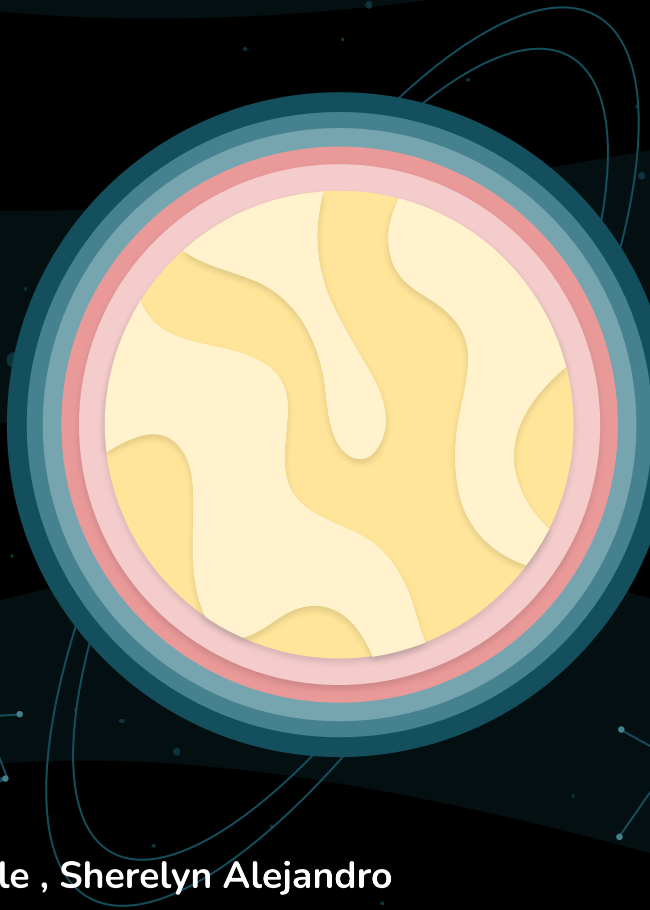
Astronomy Transients



<u>ASTRAL EIGHT</u>

Chirag Rathi , Elise P, Martina Cadiz , Sara Federle , Sherelyn Alejandro

TA- Paula Llanos

Project Overview

Transients from ZTF
~ 40,000 objects and ~44 features

Exploratory data analysis

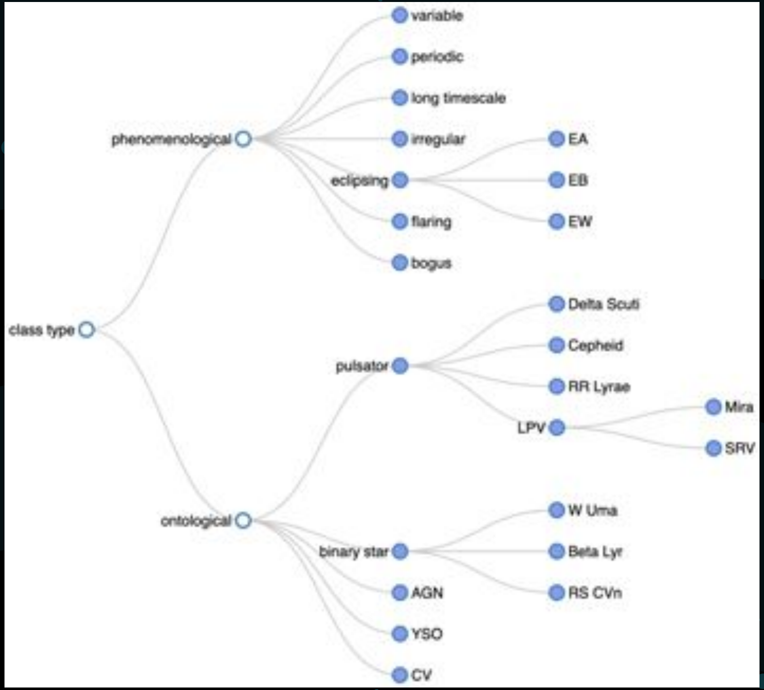
Multiclass supervised classification

Compare the performance of multiple machine learning models

Data from Zwicky Transient Facility

- Observes entire Northern Sky since 2018
- Scans every 2 days in the *g*, *r*, and *i* filters
- 1.2 m Samuel Oschin Schmidt telescope
- Magnitude limit: m_r=20.5

Obtained Data for Billions of Sources



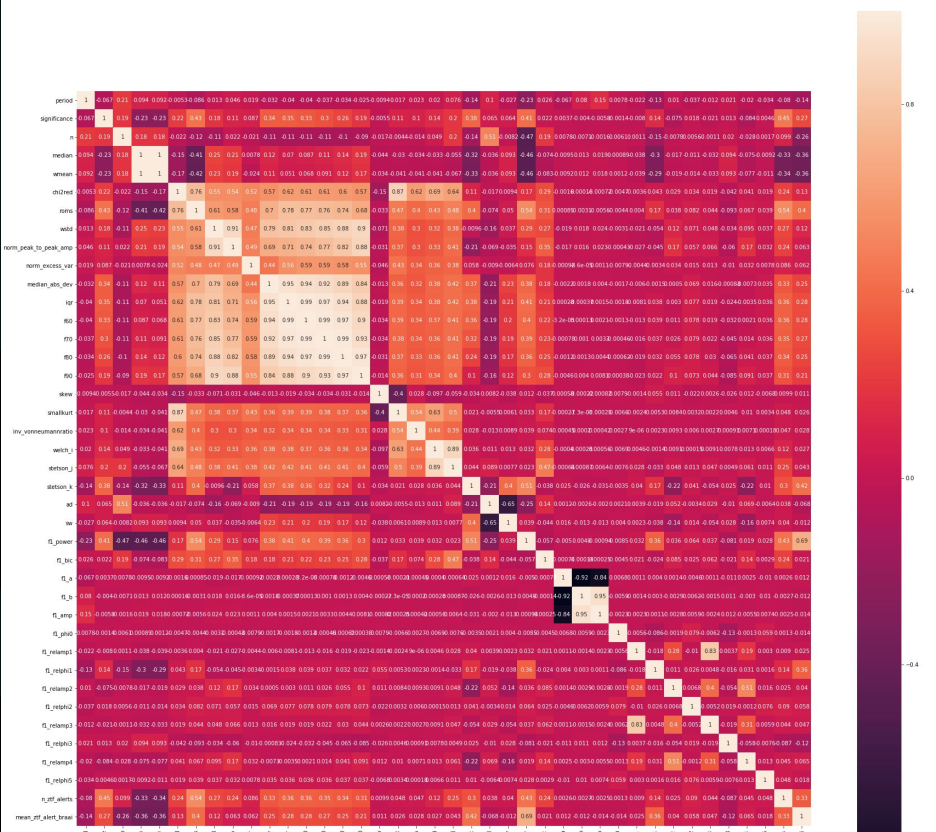
Roestel 2021

The Hurdles of Big Data

Supervised Machine Learning can help us analyze our large sample of data with known classifications more efficiently, allowing us to understand our sample even better

SF

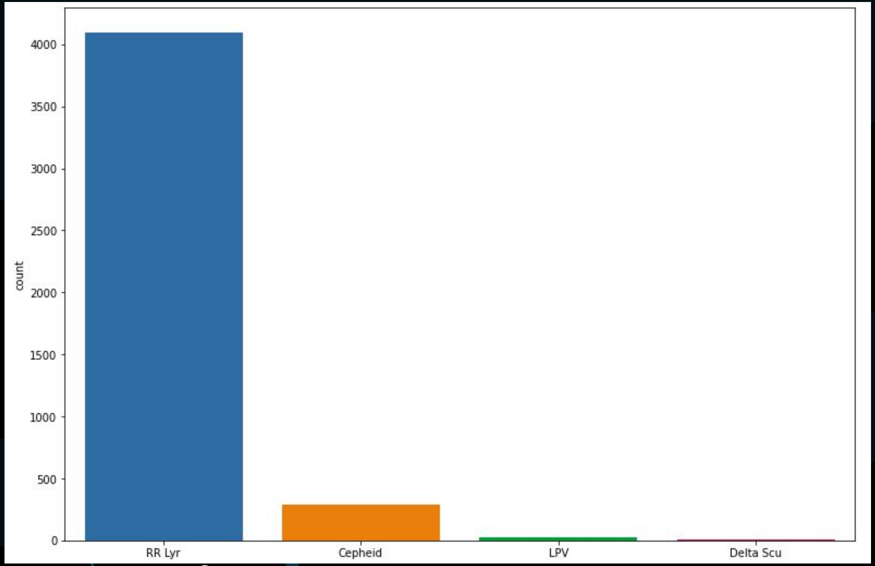
Exploratory Data Analysis: Correlation matrix

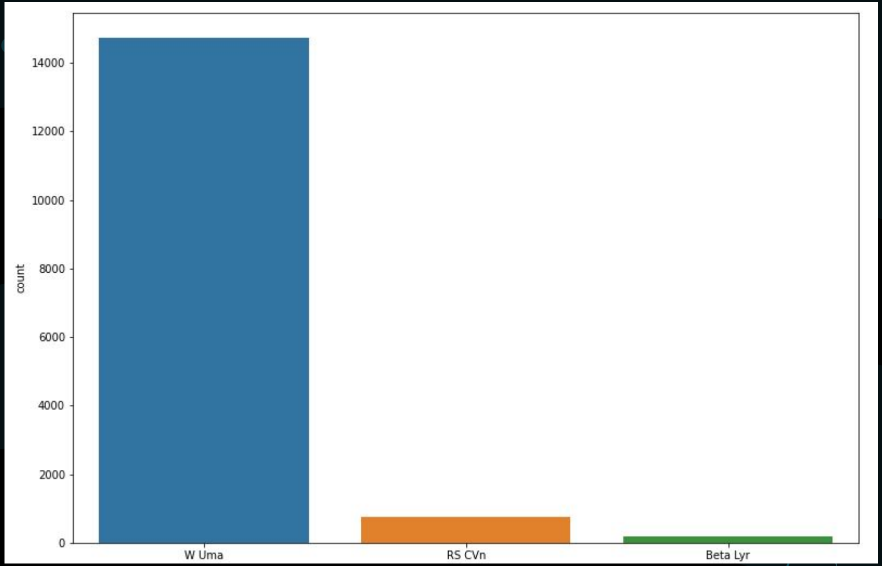


The matrix shows the correlation among the features

Orange denote strong correlation, whereas darker colors denote no correlation

Exploratory Data Analysis: Pulsators and Binaries





Count Plots for the different classes of pulsators (*left panel*) and binaries (*right panel*)

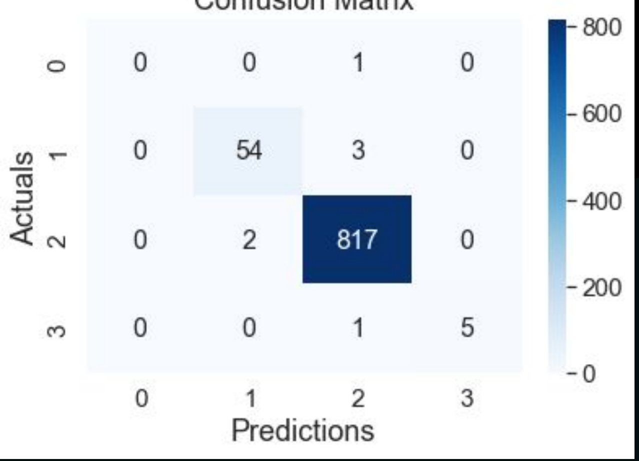
Dealing with Data Imbalances

- Stratified sampling
- Data augmentation techniques (SMOTE, random sampling)

Dimensionality reduction methods:

- Recursive Feature Elimination
- Principal component analysis (PCA)

Data Imbalance:



Method 1: Random oversampling or undersampling

Oversampling: produce exact (and random) copies of minority data

until minority = majority.

Undersampling: randomly remove majority data until majority = minority.

Method 2: SMOTE

Synthetic Minority Oversampling Technique.

Uses k-nearest neighbors to mimic data points in the minority class.

Module(s) used: *imblearn.over_sampling.smote*, *imblearn.oversampling.RandomOverSampler*

Dimensionality Reduction Methods:

METHOD 1: Recursive Feature Elimination

Too much data is overwhelming.

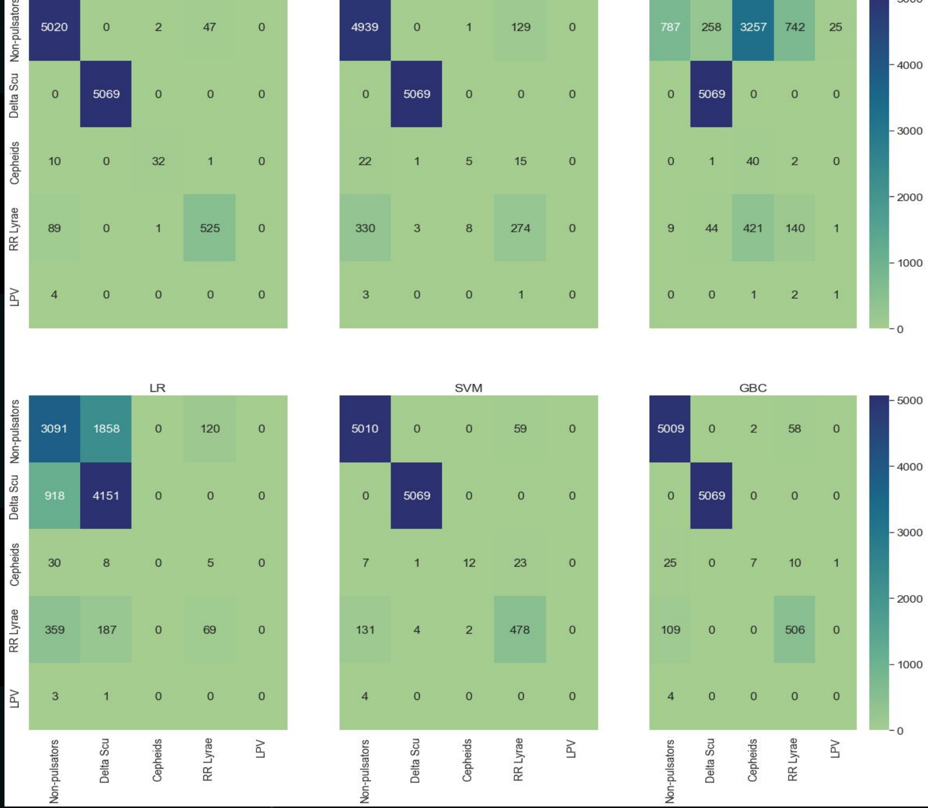
Hard to focus on anything.

RFE to the rescue! This process eliminates unnecessary features by taking smaller subsets of features and applying it to the training data.

Module(s) used: *sklearn.feature_selection.RFE, sklearn.ensemble.RandomForestRegressor*

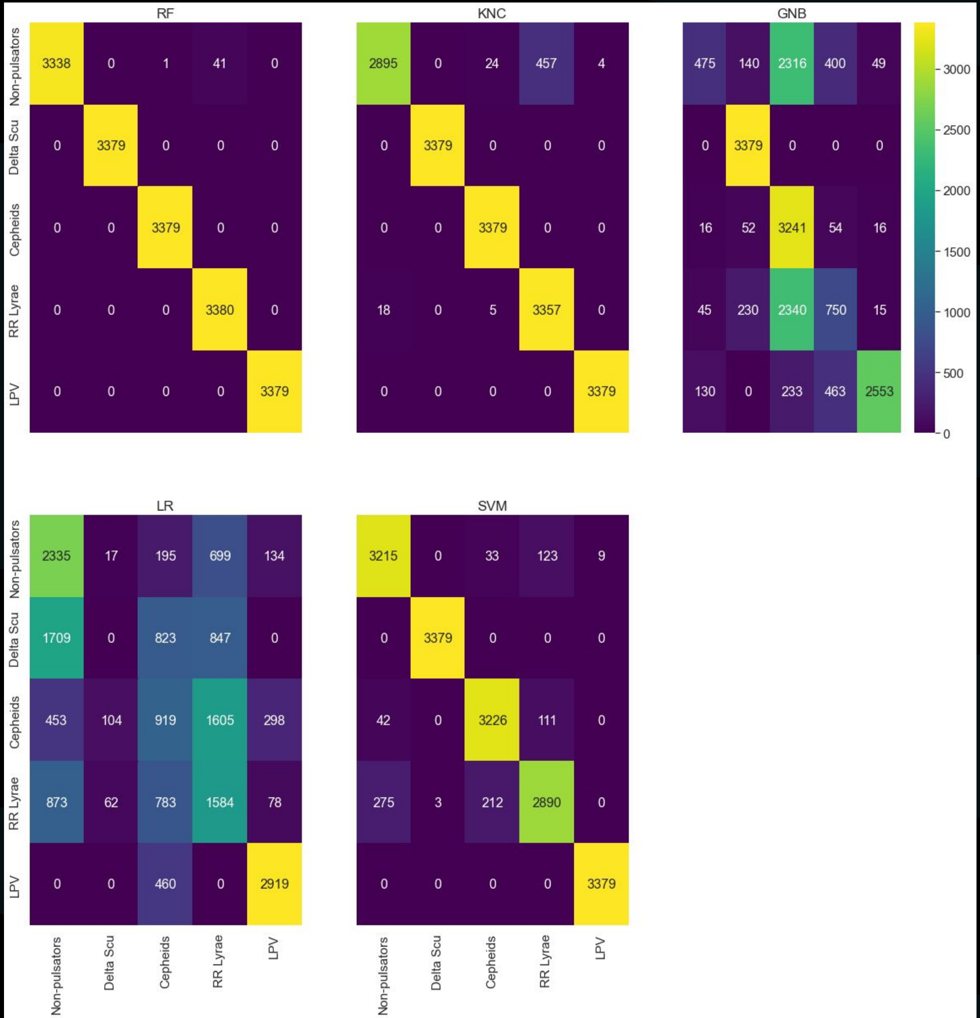
RFE + SMOTE Results:

Test size = 20%



Test size = 15%

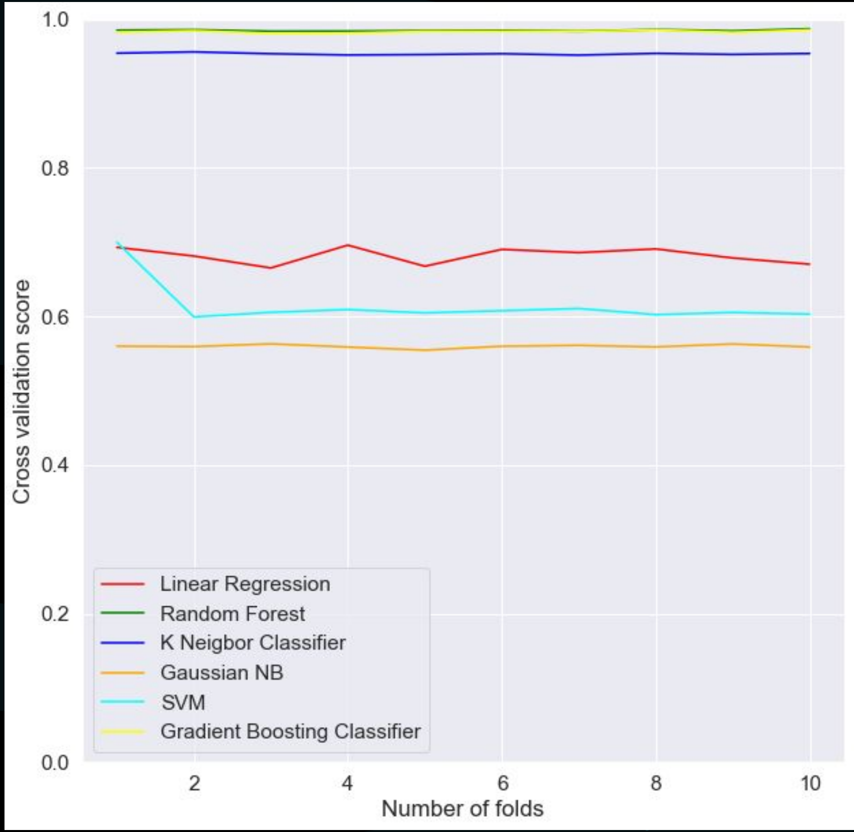
CR



CR

Test size = 10%

Comparing estimators' performances



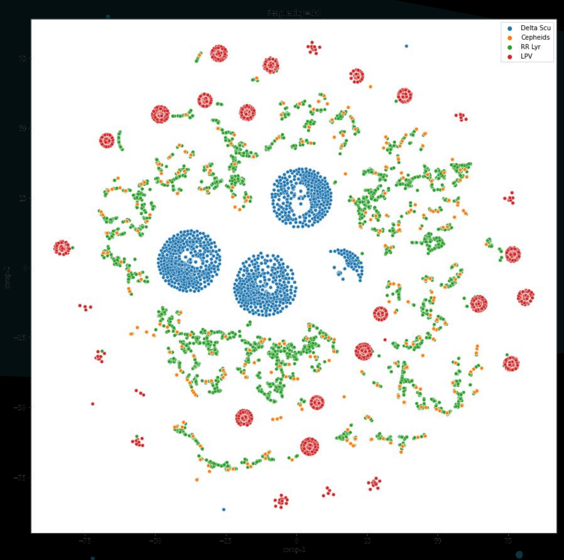
Scoring technique = accuracy
Folds = division of data into n folds
Cross validation score = predictor of performance.

Winner = GBC and RF

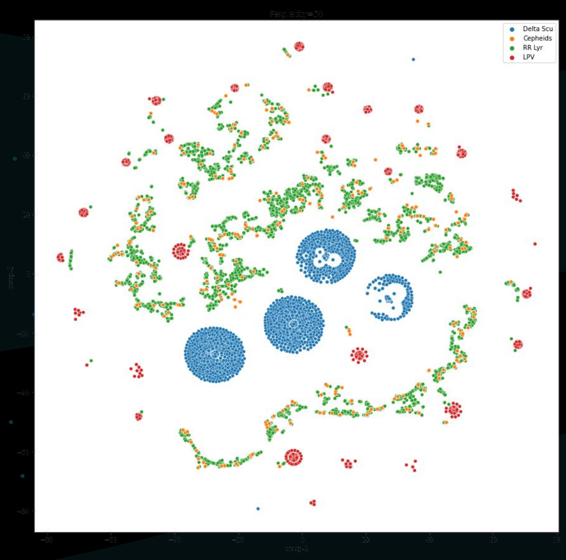
NOTE: Stratified folding!

CR

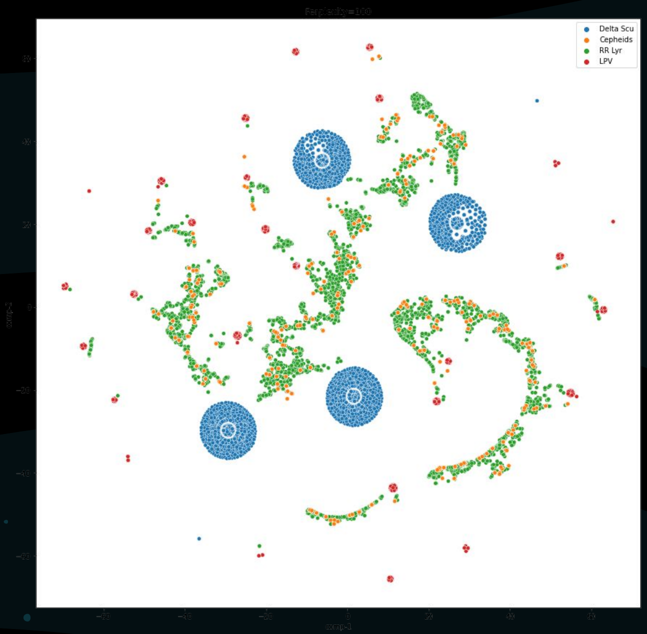
Unsupervised learning: TSNE



Perplexity = 30



Perplexity = 50



Perplexity = 100

Random Forest vs XGBoost

Random Forest

Fits a certain number of decision tree classifiers on various subsamples of the data

Improve predictive accuracy and control overfitting

XGBoost

Implementation of Gradient Boosting

Fit n classes of regression trees on the negative gradient of the loss function

Allows for optimization of differentiable loss function

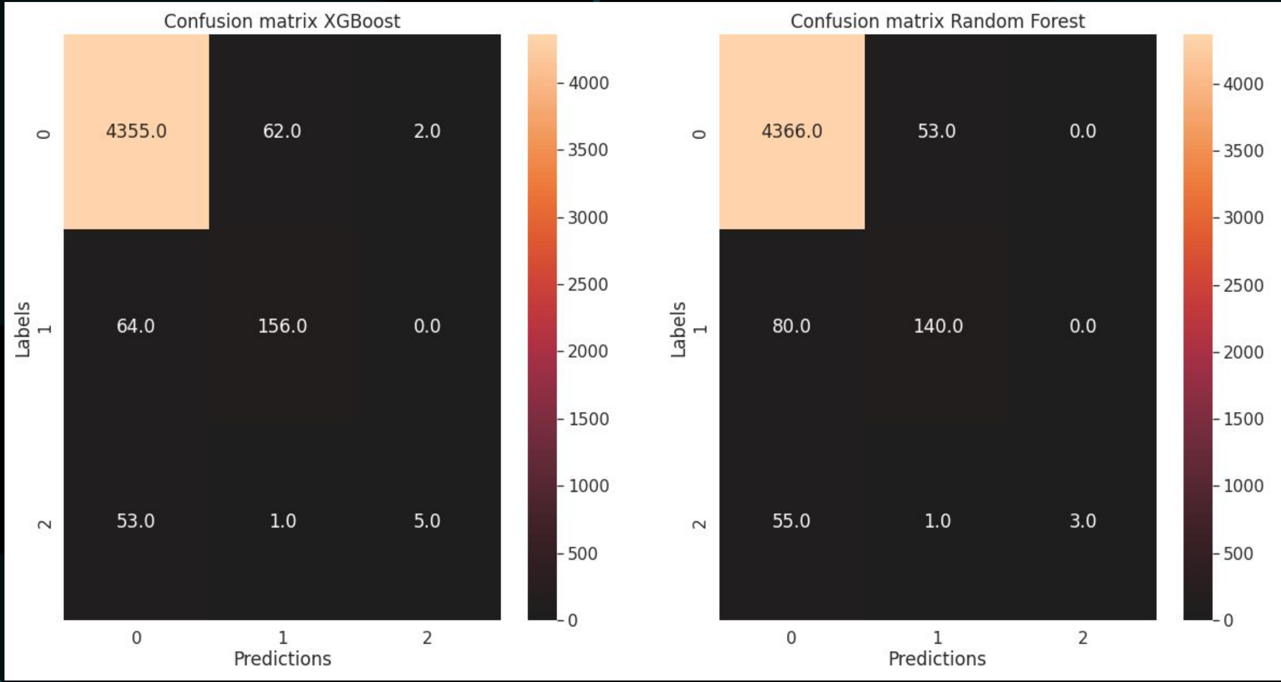
Confusion matrix (C)

Evaluates the accuracy of the classification

C_{ij} = number of

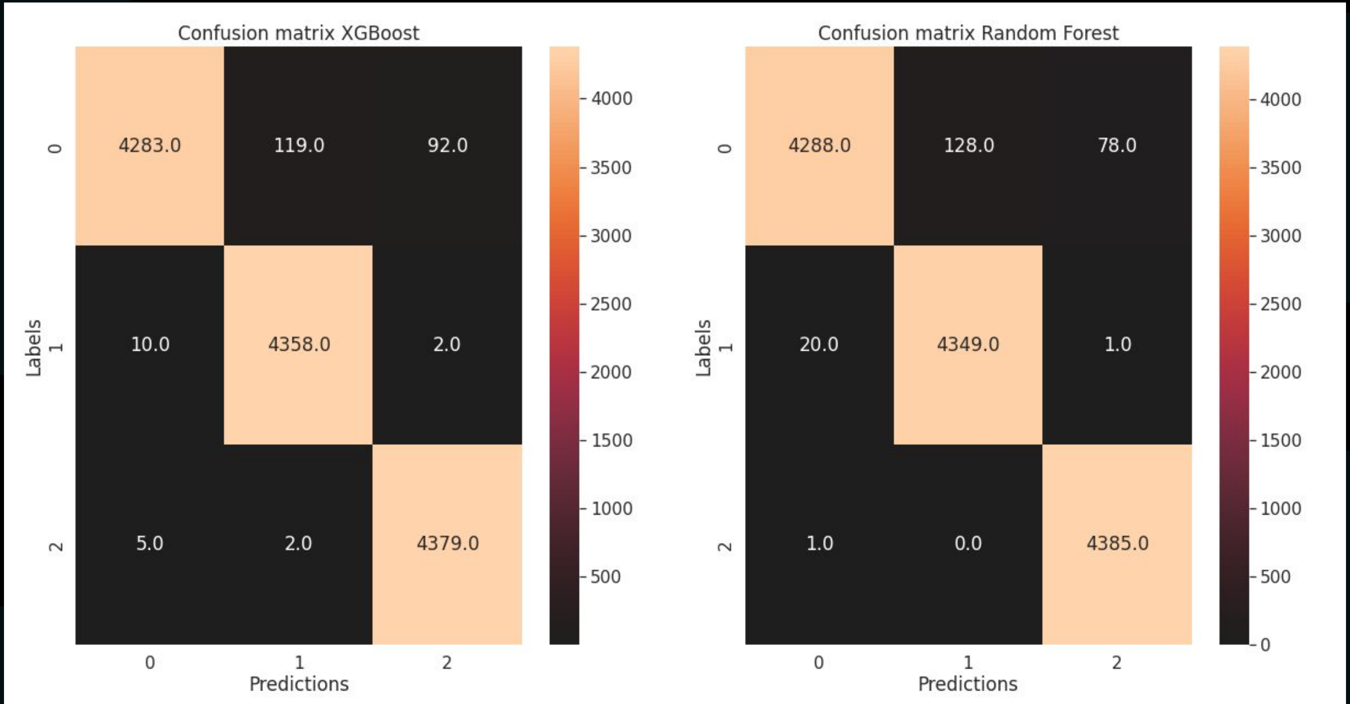
RESULTS

Machine Learning performance: unbalanced case



Confusion matrix for the XGBoost (*left panel*) and Random Forest Classifier (*right panel*) in the unbalanced case for the binaries

Machine Learning performance: balanced case



Confusion matrix for the XGBoost (*left panel*) and the Random Forest Classifier (*right panel*) in the balanced case for the binaries classification. In both cases the test size was 30%. The classes are W Uma (0), RS CVn (1)

Machine learning: classification reports

```
----- XGBoost report -----
MAE (Mean-Absolute-Error): 0.04938271604938271
MSE (Mean-Squared-Error): 0.070242656449553
RMSE (Root-MSE): 0.2650333119620117
R2 score: 0.18822588761122128
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      4425
           1       0.70      0.65      0.67       221
           2       0.67      0.04      0.07        52

    accuracy                           0.96      4698
   macro avg       0.78      0.56      0.58      4698
weighted avg       0.96      0.96      0.95      4698

----- Random Forest report -----
MAE (Mean-Absolute-Error): 0.04938271604938271
MSE (Mean-Squared-Error): 0.070242656449553
RMSE (Root-MSE): 0.2650333119620117
R2 score: 0.18822588761122128
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      4425
           1       0.72      0.65      0.68       221
           2       1.00      0.06      0.11        52

    accuracy                           0.96      4698
   macro avg       0.90      0.57      0.59      4698
weighted avg       0.96      0.96      0.96      4698
```

```
----- XGBoost report -----
MAE (Mean-Absolute-Error): 0.024679245283018868
MSE (Mean-Squared-Error): 0.039320754716981134
RMSE (Root-MSE): 0.19829461595560566
R2 score: 0.9413230119973818
              precision    recall  f1-score   support

           0       1.00      0.95      0.97      4494
           1       0.97      1.00      0.98      4370
           2       0.98      1.00      0.99      4386

    accuracy                           0.98     13250
   macro avg       0.98      0.98      0.98     13250
weighted avg       0.98      0.98      0.98     13250

----- Random Forest report -----
MAE (Mean-Absolute-Error): 0.023169811320754716
MSE (Mean-Squared-Error): 0.03509433962264151
RMSE (Root-MSE): 0.18733483291326658
R2 score: 0.947629943529333
              precision    recall  f1-score   support

           0       1.00      0.95      0.97      4494
           1       0.97      1.00      0.98      4370
           2       0.98      1.00      0.99      4386

    accuracy                           0.98     13250
   macro avg       0.98      0.98      0.98     13250
weighted avg       0.98      0.98      0.98     13250
```

SF

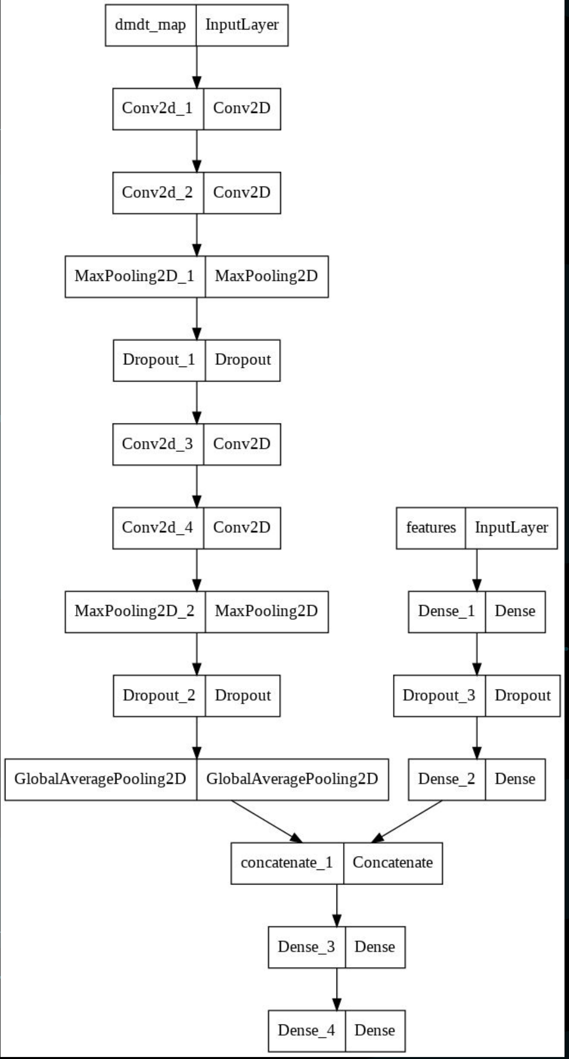
Machine Learning performances: scores and runtimes

Model	Score	Runtime Training	Runtime Prediction
Random Forest	96.104725	8.162835	0.072484
XGBoost	95.998297	8.334034	0.044336
SVC	94.870158	4.988982	1.064658
KNN	94.061303	0.004444	2.072916

Model	Score	Runtime Training	Runtime Prediction
XGBoost	98.264151	8.334034	0.044336
Random Forest	98.249057	22.581923	0.274669
KNN	94.075472	0.011730	17.398797

Comparison of different Machine Learning methods applied for the binaries classification for the unbalanced (*top panel*) and balanced (*bottom panel*) cases

Deep Learning classification



CNN architecture used

Aim to use the light curves image set

Data preparation

Train set (8K), test set (4K), validation set (2K)

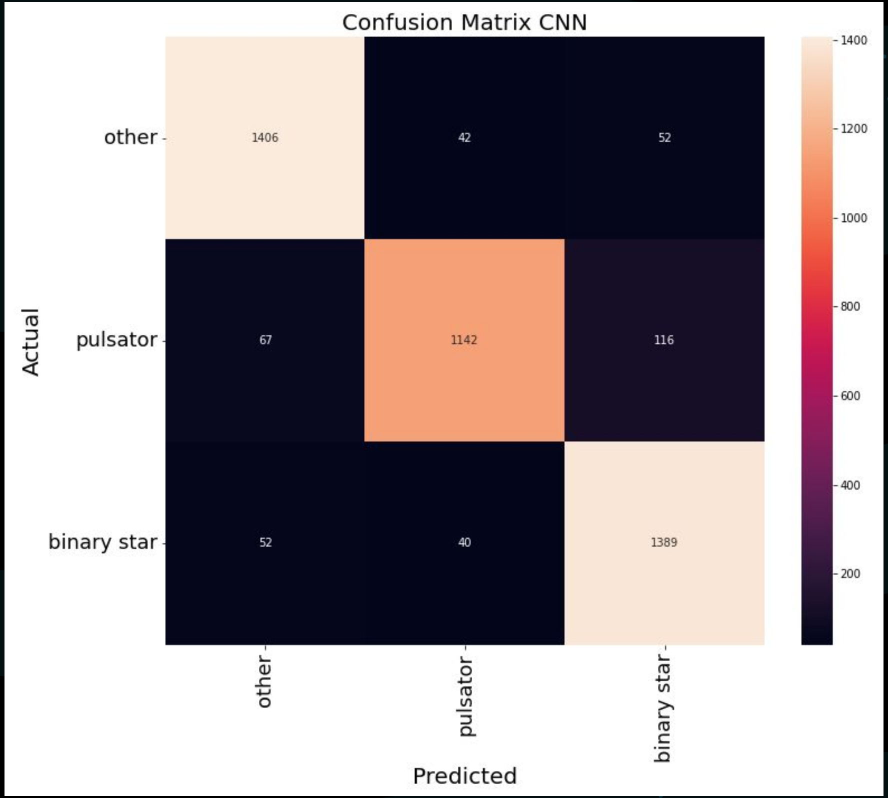
- PCA for dimensionality reduction
- Image resolution 28x28

Training stage

- EarlyStopping w/ 10 of patience
- Dropout
- Time consumed for training: 55.541 seconds

RESULTS

Deep Learning performance



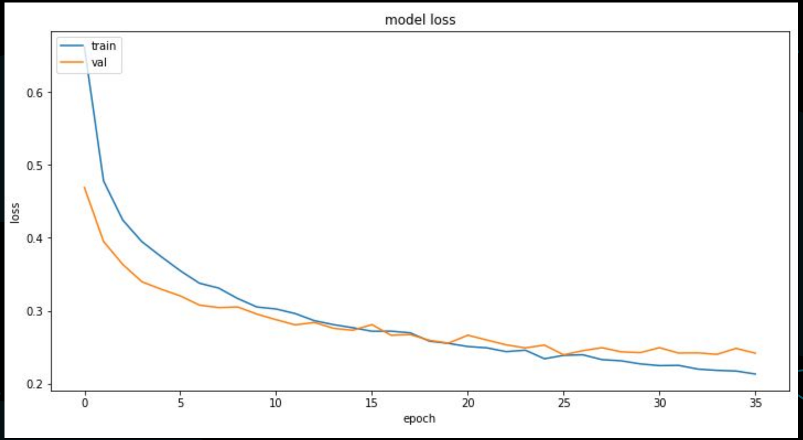
Confusion Matrix for the CNN

```
----- CNN report -----
MAE (Mean-Absolute-Error): 0.12378077101718532
MSE (Mean-Squared-Error): 0.17533673943334882
RMSE (Root-MSE): 0.41873230044188
R2 score: 0.746722162385355
              precision    recall  f1-score   support

           0       0.90      0.94      0.92      1500
           1       0.94      0.82      0.88      1325
           2       0.88      0.93      0.91      1481

    accuracy                           0.90      4306
   macro avg       0.90      0.90      0.90      4306
weighted avg       0.90      0.90      0.90      4306
```

CNN report



Learning curve

CONCLUSIONS & FUTURE WORK

- RESULTS
  - Classified the pulsators successfully using RFE process.
  - Classified binaries with different classification methods → Random Forest and XGBoost gave the best results
  - The machine learning models presented worked better than the CNN
- NEXT STEPS
  - Need more data for particular sub-classes of pulsators and binaries to improve the classification
  - Try another data augmentation technique

ACKNOWLEDGMENTS

Thank you to La Serena School For Data Science ! Niharika Sravan and Paula Llanos

REFERENCES

Jan van Roestel, Dmitry A. Duev, Ashish A. Mahabal, Michael W. Coughlin, Przemek Mróz, Kevin Burdge, Andrew Drake, Matthew J. Graham, Lynne Hillenbrand, Eric C. Bellm, Thomas Kupfer, Alexandre Delacroix, C. Fremling, V. Zach Golkhou, David Hale, Russ R. Laher, Frank J. Masci, Reed Riddle, Philippe Rosnet, Ben Rusholme, Roger Smith, Maayane T. Soumagnac, Richard Walters, Thomas A. Prince, & S. R. Kulkarni (2021). The ZTF Source Classification Project. I. Methods and Infrastructure. *The Astronomical Journal, 161(6), 267.*

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Guillaume Lemaitre, Fernando Nogueira, & Christos K. Aridas (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research, 18(17), 1-5.*

Hunter, J. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9(3), 90–95.*

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. *Array programming with NumPy*. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.

Wes McKinney (2010). Data Structures for Statistical Computing in Python . In *Proceedings of the 9th Python in Science Conference* (pp. 51 - 56).

Michael L. Waskom (2021). seaborn: statistical data visualization. *Journal of Open Source Software, 6(60), 3021.*

CREDITS:

This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**