Pulsar prediction with HTRU2 data set

Ruben R. Kazumov

6/6/2019

Contents

Acronyms, variables and abbreviations

DM - Dispersion measure

SNR - Signal-to-noise ratio

RFI - radio frequency interference

LOFAR - low frequency array

FAST - Five hundred metre Aperture Spherical Telescope

SKA - Square Kilometre Array

ALFA - Arecibo L-band Feed Array

ANN - artificial neural network

PMPS - Parkes multi-beam pulsar survey

HTRU2 - High Time Resolution Universe Survey. Name of the data set. Collection of pulsar candidates.

P - Pulsar (in legend)

NP - Not a pulsar (in legend)

μ - Mean

σ - Standard deviation

K - Excess kurtosis

S - Skewness

Listen!
If stars are lit
It means there is someone who needs it,
It means someone wants them to be,
That someone deems those specks of spit
Magnificent!

— *Vladimir Mayakovsky, Listen!*

Executive summary

The data set HTRU2 is a well known public data set, which has been studied with different approaches [1].

Many authors attempted to improve the classification accuracy by various methods, including, but not limited to Random Forest, Generalized Linear Model, Support Vector Machine, ANN [2].

In this project, we will study the HTRU2 data set with several methods of classification, we will compare the methods of classification by accuracy of prediction, and the calculation time. We will find the errors of classification and study the reasons for these errors. We will try to understand the origins of the classification errors and will make the conclusion about the possibility to make the model more precise in general.

Also, we will overview the concept of feature selection as the original events derivative.

The data set

The data set HTRU2 is a collection of the pulsar candidate star observations. The features of the data set describe the parameters of continuous signal from numerous stars, perfected by station. The original set has nine features which are listed in Table 1.

Table 1: List of features in HTRU2 data set

#	Feature name	Symbol	Column
1	Mean of the integrated profile	$Prof_\mu$	profMu
2	Standard deviation of the integrated profile	$Prof_\sigma$	profSigma
3	Excess kurtosis of the integrated profile	$Prof_k$	profK
4	Skewness of the integrated profile	$Prof_S$	profS
5	Mean of the DM-SNR curve	DM_μ	dmMu
6	Standard deviation of the DM-SNR curve	DM_σ	dmSigma
7	Excess kurtosis of the DM-SNR curve	DM_k	dmK
8	Skewness of the DM-SNR curve	DM_S	dmS
9	Class of star		starClass

All the features are the derivatives of two pre-processed objects:

1. Integrated profile of the star, and
2. Dispersion measure signal-to-noise ratio (DM-SNR) curve [3][4].

Integrated profile

The integrated profile feature group, describes the shape of the radio signals from the stars, intercepted by stations. The signals intercepted by stations, like LOFAR, SKA, ALFA etc. are passed through multiple pre-processing procedures and folded into the "integrated profile" of a star [5][6][7].

The integrated profile describes the change of the shape and magnitude of impulse-like radio-signal series from the star. Integrated profile statistically describes the history of the star signals.

The statistical description in the form of mean ($Prof_\mu$), standard deviation ($Prof_\sigma$), excess kurtosis ($Prof_K$), and skewness ($Prof_S$), are expected to be similar for each of the different star classes.

Since the pulsar is the sub-class of the neutron star, there expected to be similarity in the integrated profile of pulsars.

DM-SNR curve

Signal from the star comes in the continuous radio-frequency band [8]. The speed of the signal disperses for different frequencies with the different speed. Immediately after the impulse, the station starts to receive the same signal but on a different frequency bands. It is a continuous process. The high frequency signal comes with less delay, but low frequency signal comes with more delay. Station continuously receives signals with lower and lower frequencies. It is the same signal, but cloned with the delay on different radio-frequency bands.

Since the cloned signals can be described statistically, the DM-SNR curve can be described in terms of the mean (DM_μ), standard deviation (DM_σ), excess kurtosis (DM_K), and skewness (DM_S), as well as integrated profile.

Class of star

The feature "Class of star" is our target feature for the prediction. We will describe it as a factor with the values "pulsar" and "not a pulsar".

Statistical overview of the features

We scale the eight features of the data set to make them visually more defined (Figure1). One should notice, the features have distant averages, but the feature values for the different star classes are heavily overlapped for each feature.

One can study the features correlation matrix (Table2). We can split the correlation matrices for pulsars and not the pulsar classes and present them in the graphical correlation matrix (Figure2). The plot describes the correlation between features of the data set. The red color highlights not a pulsar stars, but the green one highlights pulsars.

Table 2: The data features correlation.

	profMu	profSigma	profK	profS	dmMu	dmSigma	dmK	dmS	starClass
profMu	1.00	0.55	-0.87	-0.74	-0.30	-0.31	0.23	0.14	-0.67
profSigma	0.55	1.00	-0.52	-0.54	0.01	-0.05	0.03	0.03	-0.36
profK	-0.87	-0.52	1.00	0.95	0.41	0.43	-0.34	-0.21	0.79
profS	-0.74	-0.54	0.95	1.00	0.41	0.42	-0.33	-0.20	0.71
dmMu	-0.30	0.01	0.41	0.41	1.00	0.80	-0.62	-0.35	0.40
dmSigma	-0.31	-0.05	0.43	0.42	0.80	1.00	-0.81	-0.58	0.49

	profMu	profSigma	profK	profS	dmMu	dmSigma	dmK	dmS	starClass
dmK	0.23	0.03	-0.34	-0.33	-0.62	-0.81	1.00	0.92	-0.39
dmS	0.14	0.03	-0.21	-0.20	-0.35	-0.58	0.92	1.00	-0.26
starClass	-0.67	-0.36	0.79	0.71	0.40	0.49	-0.39	-0.26	1.00

One can notice the grouping of the pulsar and not a pulsar stars. Also one can see the values of correlation coefficient between the pairs.

As we can see in Table 3, the data set is heavily unbalanced. The number of `not a pulsar` observations is ten times bigger than `pulsar` ones. We can suggest, that the data set builder expected the heavy prevalence of a `not a pulsar` classes over `pulsar` ones in reality and tried to recreate this proportion.

Table 3: Proportion of `pulsar` and `not a pulsar` classes in the data set.

	Count
not a pulsar	16259
pulsar	1639

However, as one may suspect, the unbalanced data may cause problems in a prediction system results. Later we will see the affect of this disproportion in confidence matrix.

Classification

For the classification we split the observations into training and testing sets:

```
n = dim(stars)[1] # total number of observations

trainIdx <- sample(x = n, size = n * 0.9) # 90% of observations for the training set, and
                                          # 10% for the test set

train <- stars[trainIdx, ]

test <- stars[-trainIdx, ]

summary(train$starClass) # check the content of the training set
# > summary(train$starClass)
# not a pulsar       pulsar
#        14635         1473

summary(test$starClass) # check the content of the testing set
# > summary(test$starClass)
# not a pulsar       pulsar
#         1624          166

x <- train %>% select(-starClass)

y <- train$starClass

xTest <- test %>% select(-starClass)
```

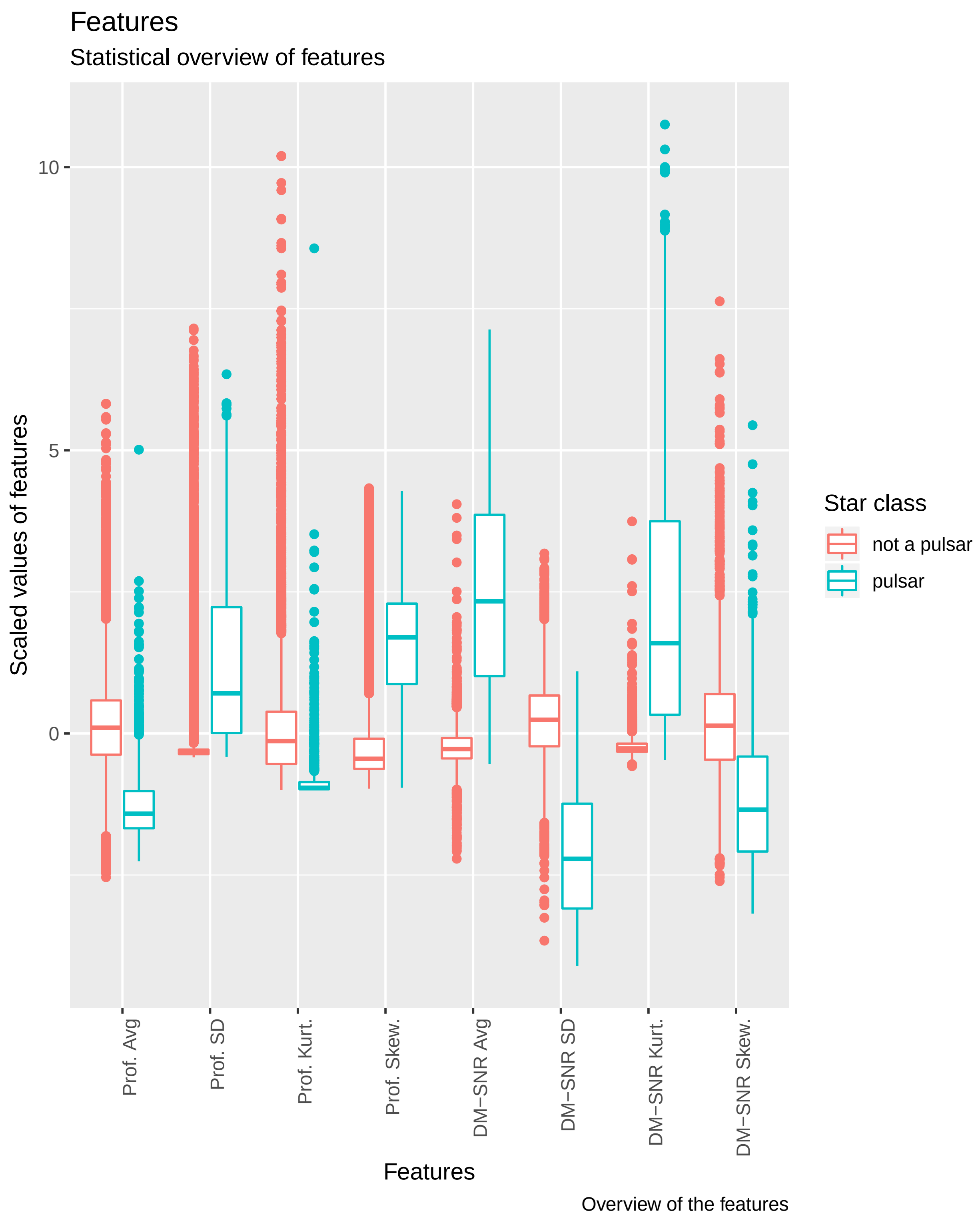


Figure 1: Features overview

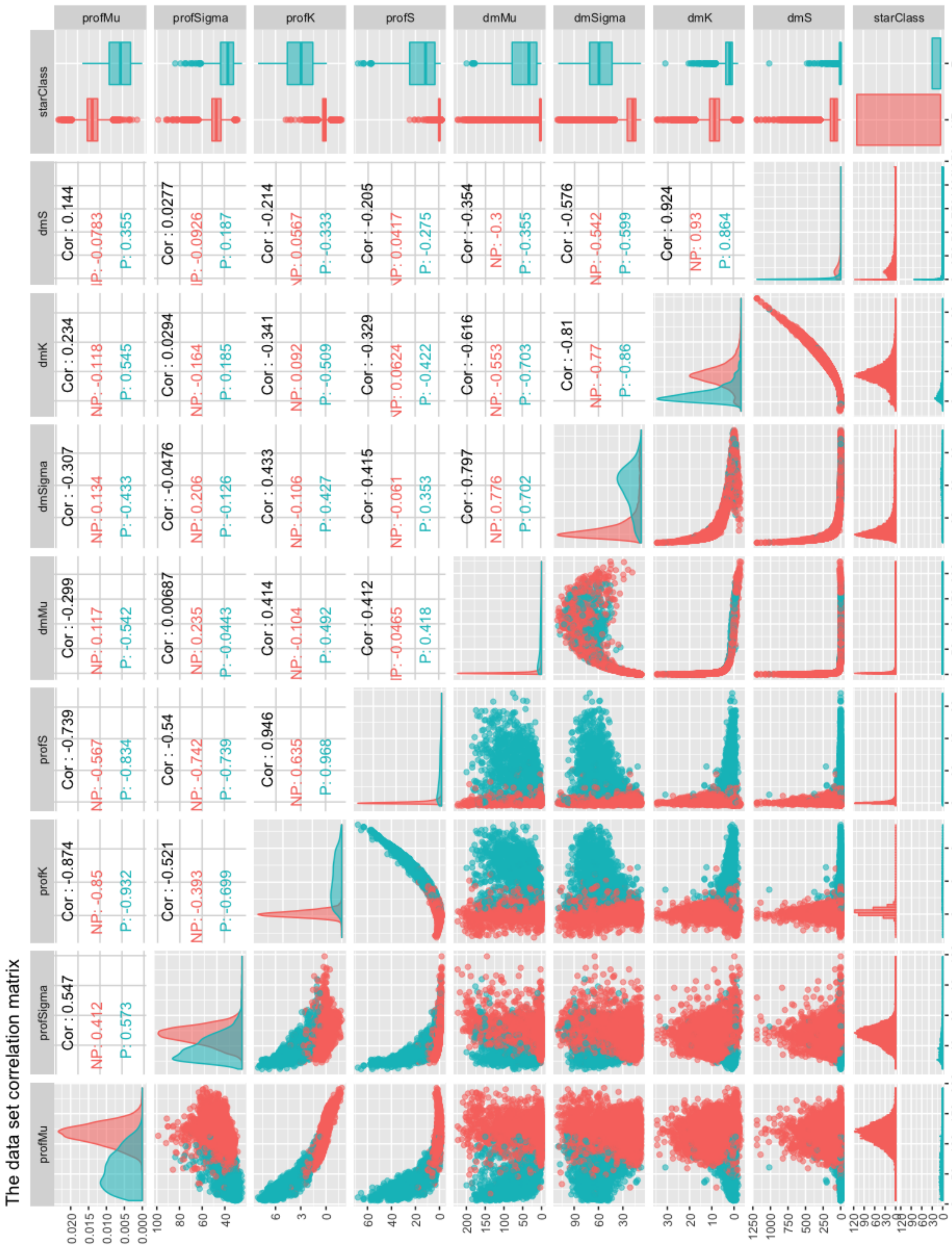


Figure 2: The data set correlation matrix.

```r
yTest <- test$starClass
```

As one can see, the test set consists only of 166 observations of confirmed pulsar star class.

Quick classification with Rborist package

We can try to fit the model with the Rborist algorithm. The Rborist library should show us the lower level of possible prediction. The serious advantage of Rborist library is possibility to work with the maximum observations population.

```r
fitRborist <- Rborist(x = x, y = y)
```

The confusion matrix of the prediction:

```
## $positive
## [1] "not a pulsar"
##
## $table
##               Reference
## Prediction     not a pulsar pulsar
##   not a pulsar         1612     29
##   pulsar                 12    137
##
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.770950e-01   8.573239e-01   9.690540e-01   9.835140e-01   9.072626e-01
## AccuracyPValue  McnemarPValue
##   3.066641e-33   1.246216e-02
##
## $byClass
##          Sensitivity          Specificity       Pos Pred Value
##            0.9926108            0.8253012            0.9823278
##       Neg Pred Value            Precision               Recall
##            0.9194631            0.9823278            0.9926108
##                   F1           Prevalence       Detection Rate
##            0.9874426            0.9072626            0.9005587
## Detection Prevalence    Balanced Accuracy
##            0.9167598            0.9089560
##
## $mode
## [1] "sens_spec"
##
## $dots
## list()
##
## attr(,"class")
## [1] "confusionMatrix"
```

One can notice the Rborist predicts star class with the overall accuracy 98% for the `not a pulsar` positive.

Classification with library Random Forest

We will perform the prediction with Random Forest algorithm three times:

i) With default parameters;
ii) With adjusted `ntree` parameter; and
iii) With adjusted `ntree` and `mtry` parameter.

The first run with default parameters:

```
fitRandomForest <- randomForest(x = x, y = y)
```

The result of the prediction with the default train parameters:

```
## $positive
## [1] "not a pulsar"
##
## $table
##                 Reference
## Prediction        not a pulsar pulsar
##   not a pulsar            1613     27
##   pulsar                    11    139
##
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.787709e-01   8.681374e-01   9.709770e-01   9.849343e-01   9.072626e-01
## AccuracyPValue  McnemarPValue
##   3.347403e-35   1.496102e-02
##
## $byClass
##          Sensitivity          Specificity       Pos Pred Value
##            0.9932266            0.8373494            0.9835366
##       Neg Pred Value            Precision               Recall
##            0.9266667            0.9835366            0.9932266
##                   F1           Prevalence       Detection Rate
##            0.9883578            0.9072626            0.9011173
## Detection Prevalence    Balanced Accuracy
##            0.9162011            0.9152880
##
## $mode
## [1] "sens_spec"
##
## $dots
## list()
##
## attr(,"class")
## [1] "confusionMatrix"
```

The optimization of `ntree` parameter

We will optimize `ntree` by iterating the possible values from 50 to 500 by 10:

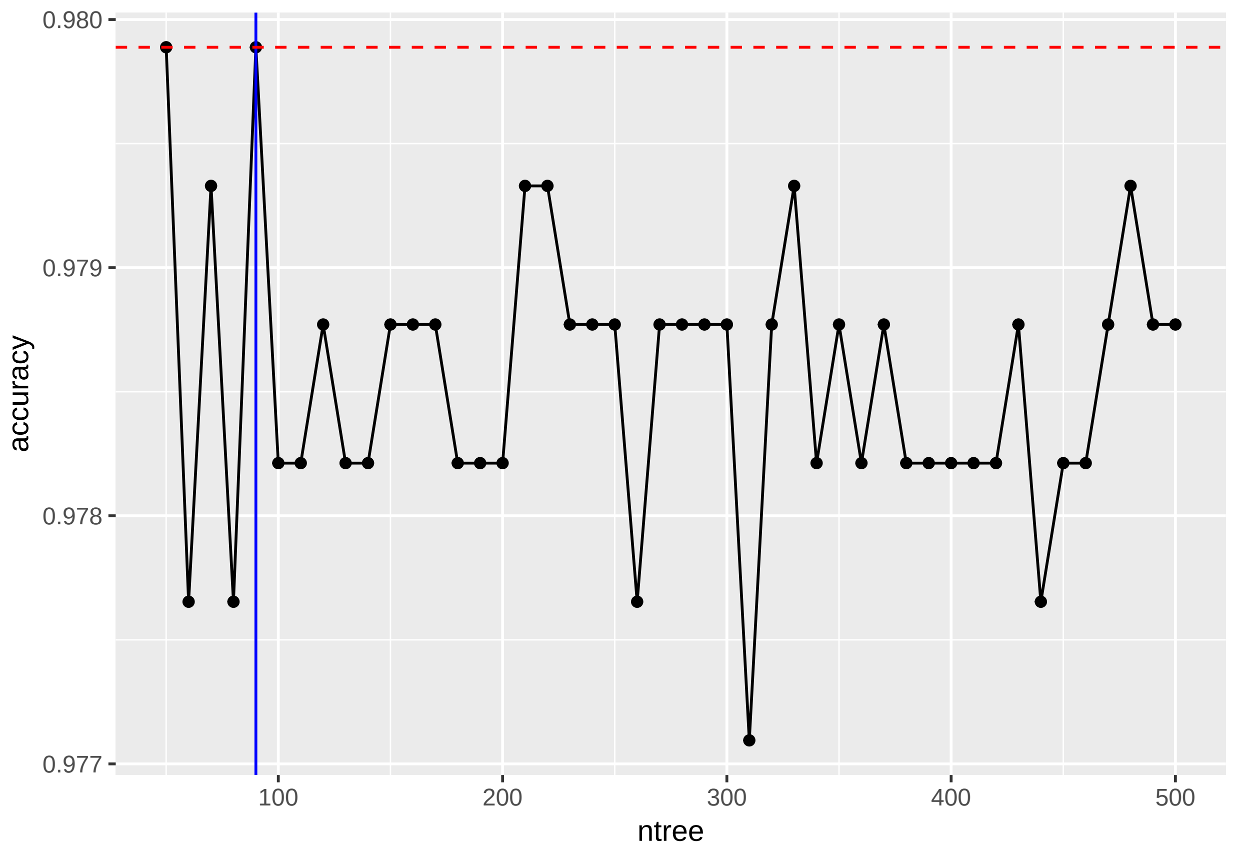


Figure 3: Adjustment of `ntree` parameter.

```r
ntree <- seq(from = 50, to = 500, by = 10)

nTreeVariants <- lapply(ntree, function(ntree){
  fit <- randomForest(x = x, y = y)
  yHat <- predict(object = fit, newdata = xTest)
  cm <- confusionMatrix(yHat, yTest)
  return(
    data.frame(ntree = ntree, accuracy = cm$overall["Accuracy"])
  )
}) %>% bind_rows()

nTreeSelected <- nTreeVariants %>%
  group_by(ntree) %>%
  summarise(maxAccuracy = max(accuracy)) %>%
  arrange(maxAccuracy) %>%
  tail(1) %>% .$ntree
```

The `ntree` over `accuracy` dependency plot displays the unstable, random-like change of the accuracy parameter in interval (0.977 . . . 0.980) dependent on `ntree`. We will select the minimal possible value of `ntree` with the maximum `accuracy` value (Fig. 3). The optimal value of `ntree` highlighted by blue vertical line.

The optimization of `mtry` parameter

With the defined optimal `ntree` value, we will find the optimal `mtry` paramenter:

```
mtry <- 1:8 # possible values

mtryVariants <- lapply(mtry, function(mtry){
  fit <- randomForest(x = x, y = y)
  yHat <- predict(object = fit, newdata = xTest, ntree = nTreeSelected, mtry = mtry)
  cm <- confusionMatrix(yHat, yTest)
  return(
    data.frame(mtry = mtry, accuracy = cm$overall["Accuracy"])
  )
}) %>% bind_rows()

mtrySelected <- mtryVariants %>%
  group_by(mtry) %>%
  summarise(maxAccuracy = max(accuracy)) %>%
  arrange(maxAccuracy) %>%
  tail(1) %>% .$mtry
```

The resulting dependency has no any noticeable trend (Fig. 4). The best `mtry` parameter is highlighted on the plot with a blue line.

Optimized approach

Finally we will perform prediction whith Random Forest algorithm and will use the defined `ntree` and `mtry` parameters:

```
fitRandomForestAdjusted <- randomForest(x = x,
                                        y = y,
                                        ntree = nTreeSelected,
                                        mtry = mtrySelected)
```

Now we can overview the results of optimization in confusion matrix:

```
## $positive
## [1] "not a pulsar"
##
## $table
##                 Reference
## Prediction      not a pulsar pulsar
##   not a pulsar          1611     26
##   pulsar                  13    140
##
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.782123e-01   8.658052e-01   9.703349e-01   9.844620e-01   9.072626e-01
## AccuracyPValue  McnemarPValue
##   1.548175e-34   5.466394e-02
##
## $byClass
```

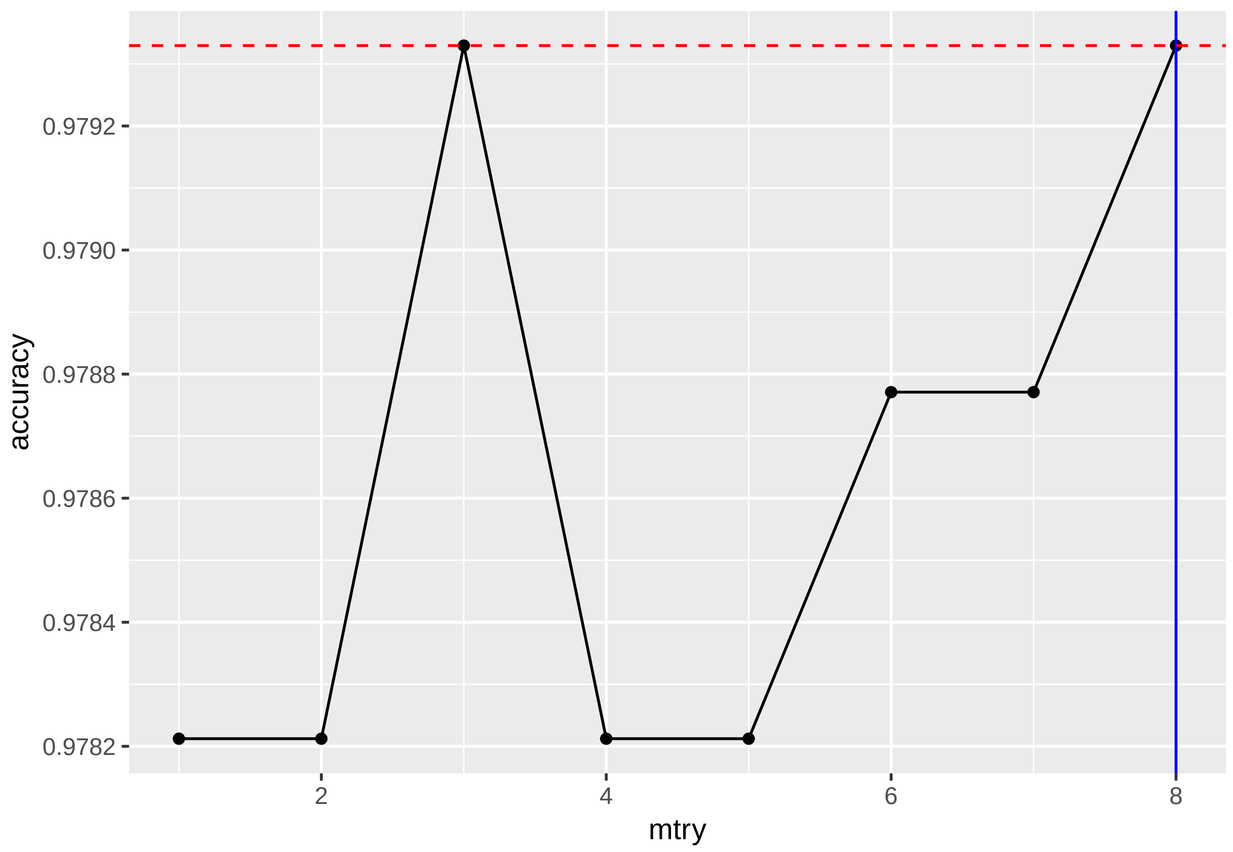


Figure 4: Adjustment of `mtry` parameter.

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.9919951            0.8433735            0.9841173
##       Neg Pred Value            Precision               Recall
##            0.9150327            0.9841173            0.9919951
##                   F1           Prevalence       Detection Rate
##            0.9880405            0.9072626            0.9000000
## Detection Prevalence    Balanced Accuracy
##            0.9145251            0.9176843
##
## $mode
## [1] "sens_spec"
##
## $dots
## list()
##
## attr(,"class")
## [1] "confusionMatrix"
```

As one can see, the optimization process does not produce any sufficient accuracy improvement.

Classification with KNN

Since Rborist and Random Forest have the same algorithmic roots, we will try to apply a radically different fitting method.

```r
library(caret)

ct <- trainControl(method = "repeatedcv", repeats = 3)

fitKnn <- train(x = x, y = y, method = "knn", trControl = ct, preProcess = c("center", "scale"), tuneLen
```

One can see, the KNN algorithm self-adjusted to minimal node (Fig. 5).

As a result, we can see the confusion matrix of KNN approach:

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     not a pulsar pulsar
##   not a pulsar         1612     29
##   pulsar                 12    137
##
##                Accuracy : 0.9771
##                  95% CI : (0.9691, 0.9835)
##     No Information Rate : 0.9073
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8573
##
##  Mcnemar's Test P-Value : 0.01246
##
##             Sensitivity : 0.9926
##             Specificity : 0.8253
```

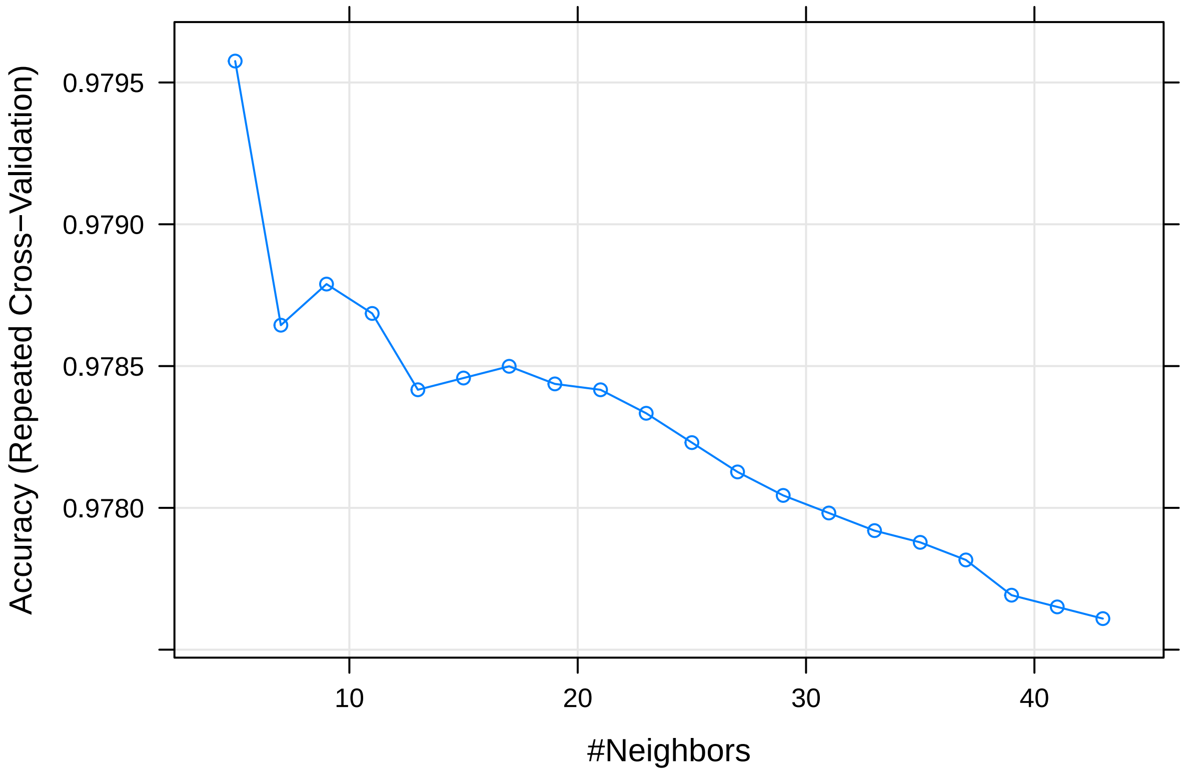


Figure 5: KNN method adjustment.

```
##          Pos Pred Value : 0.9823
##          Neg Pred Value : 0.9195
##              Prevalence : 0.9073
##          Detection Rate : 0.9006
##    Detection Prevalence : 0.9168
##       Balanced Accuracy : 0.9090
##
##        'Positive' Class : not a pulsar
##
```

Summary

We performed tree types of fitting. Three methods, Rborist, Random Forrest and KNN are performing fitting with noticeble different calculation time (Table 4), but without noticeable accuracy difference (Table 5).

Table 4: Fitting time comparison.

	test	replications	elapsed	relative	user.self	sys.self	user.child	sys.child
3	KNN	10	4005.213	239.103	1574.844	110.133	0	0
2	Random Forest	10	16.751	1.000	16.252	0.500	0	0
1	Rborist	10	23.539	1.405	88.540	2.973	0	0

Table 5: Prediction accuracy comparison.

test	Not a pulsar	Pulsar	Errornous Pulsar	Missed Pulsar
Rborist	1612	137	12	29
Random Forest	1611	140	13	26
KNN	1612	137	12	29

Now we should notice the fact, that the accuracy cannot be improved by classification method change.

Origin of errors

We consolidate the prediction errors in table 6.

Table 6: Possible types of prediction error.

Classification error	Color code	Affected observations
Pulsar was missed by the classifier.	blue	13
The star was mistaken for a pulsar.	red	26

We will structurize the error types according to the confusion matrix types and create data set `starsAndErrors`:

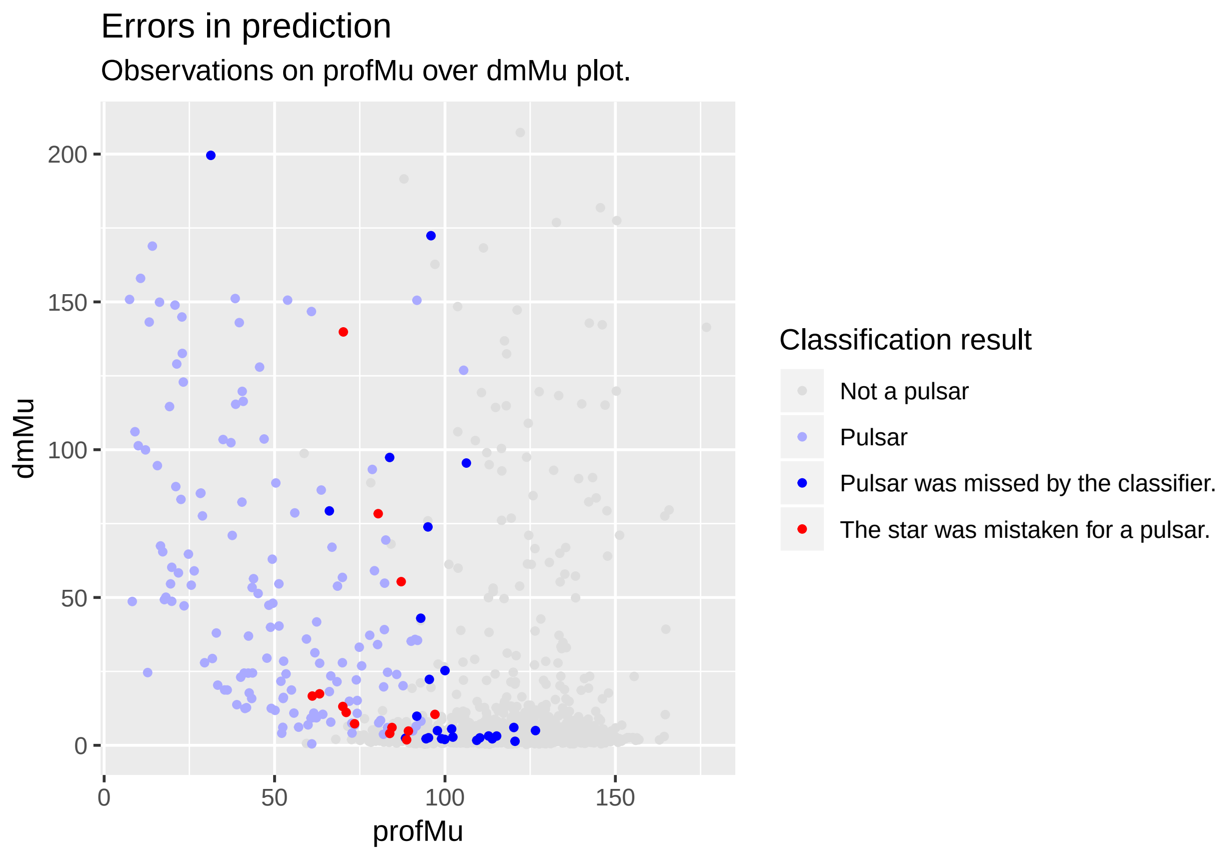


Figure 6: Erroneous predictions on $Prof_{\mu}$ vs. DM_{μ} plot.

```
resultLevels = c("Not a pulsar",
                 "Pulsar",
                 "The star was mistaken for a pulsar.",
                 "Pulsar was missed by the classifier.")


starsAndErrors <- stars %>%
  mutate(result = case_when(
  err == TRUE & starClass == "not a pulsar" ~ resultLevels[3], # it is actually not a pulsar, but we ma
  err == TRUE & starClass == "pulsar" ~ resultLevels[4], # it is actually a pulsar, but we made a mista
  err == FALSE & starClass == "not a pulsar" ~ resultLevels[1], # it is not a pulsar
  err == FALSE & starClass == "pulsar" ~ resultLevels[2], # it is a pulsar
))
```

As one can see, the `starsAndErrors` data set contains the vector `result` with the observation description.

Then we will visualize three of all possible pairs of features as 2D plots and will highlight the star classes with the colors, encoded by result level (see Fig. 6, Fig. 7, and Fig. 8).

As one can see, as the values of erroneous observations features are laying in the region of values of opposite star class, the classification method can not recognize the star class right.

Finally, we should conclude, that the source of errors is lying in the features set only. The only way to increase the prediction accuracy is by creating the independent feature, based on unused before parameters

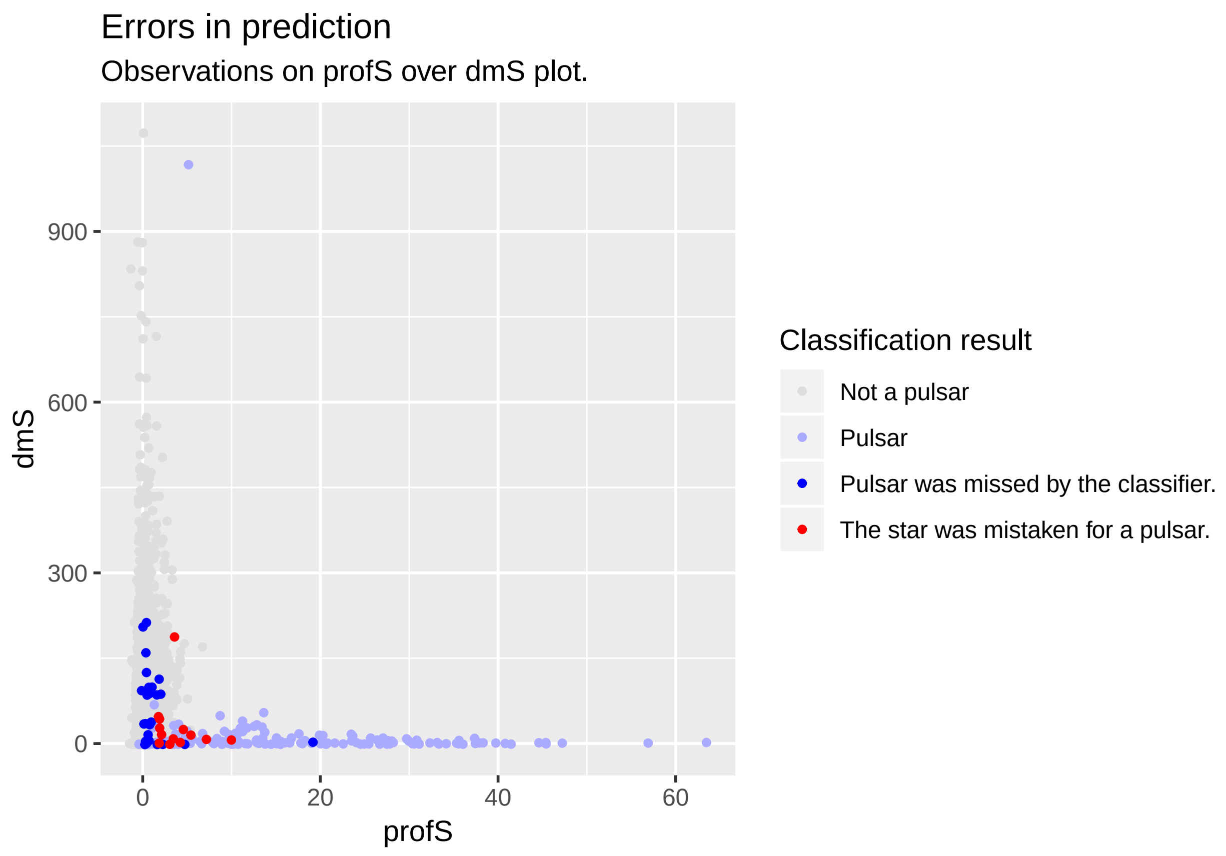


Figure 7: Erroneous predictions on $Prof_S$ vs. DM_S plot.

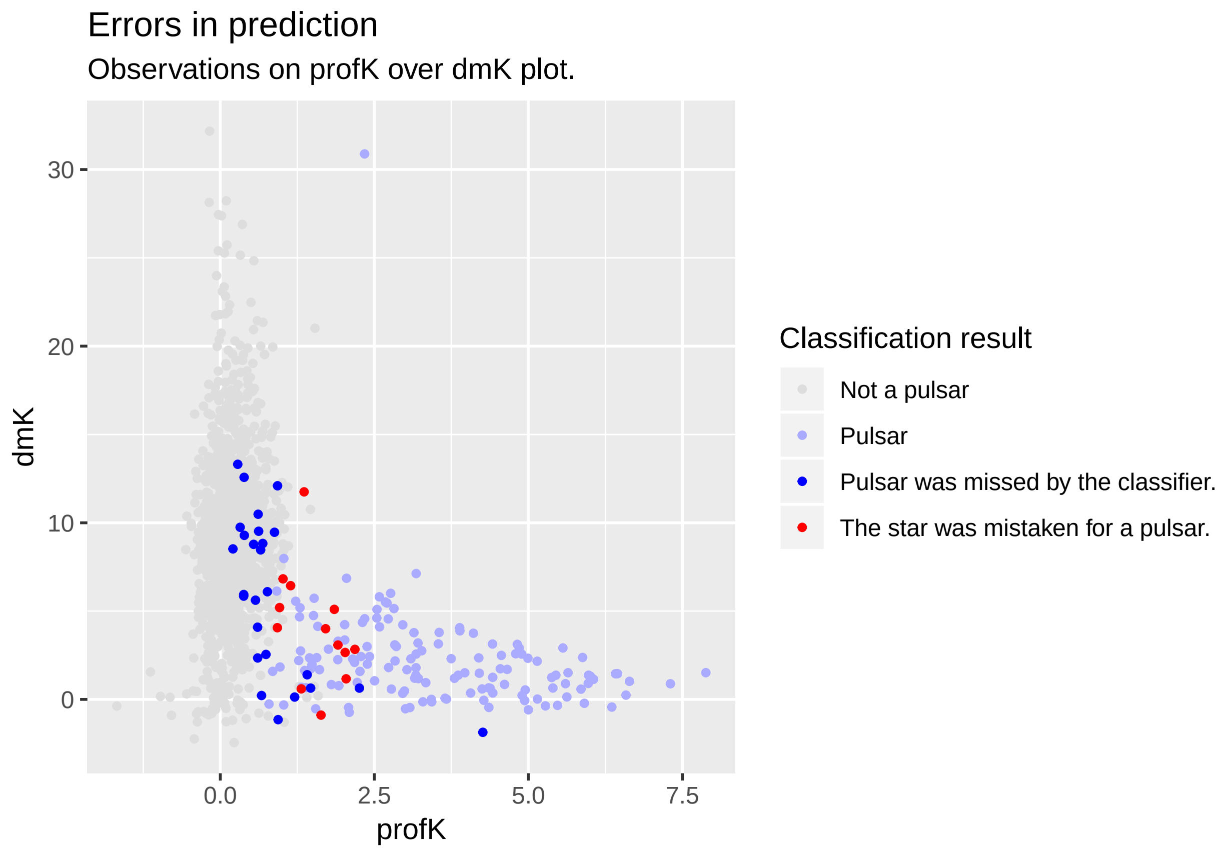


Figure 8: Erroneous predictions on $Prof_K$ vs. DM_K plot.

of the percepted signal.

3D interactive exploration

We use one dimension of a plot to visualize one feature. Since we use 2D plots, we can explore no more than tree features (including color as a plot dimension). The library `plotly` introduces 3D interactive plots. Three dimensions allows us to plot four features at once. The code for the dependency $result = f(Prof_K, DM_\mu, Prof_\mu)$ and $result = f(DM_K, Prof_\mu, DM_\mu)$ visualization:

```r
library(plotly)

levelsColors <- c("#dddddd", "#aaaaff", "red", "blue")

p3d <- plotly::plot_ly(data = starsAndErrors,
                       x = ~profK,
                       y = ~dmMu,
                       z = ~profMu,
                       color = ~result,
                       colors = levelsColors,
                       size = 1)

p3d

p3d <- plotly::plot_ly(data = starsAndErrors,
                       x = ~dmK,
                       y = ~profMu,
                       z = ~dmMu,
                       color = ~result,
                       colors = levelsColors,
                       size = 1)

p3d
```

On a Figure 9 you can see the visualization of the mentioned dependencies.

One can notice, the features values of the erroneously classified stars are deep into the values of the opposite class of stars. That is the only reason of mistake in the classification process. That is the reason, why the different classification algorithms do not sufficiently improve the results of classification.

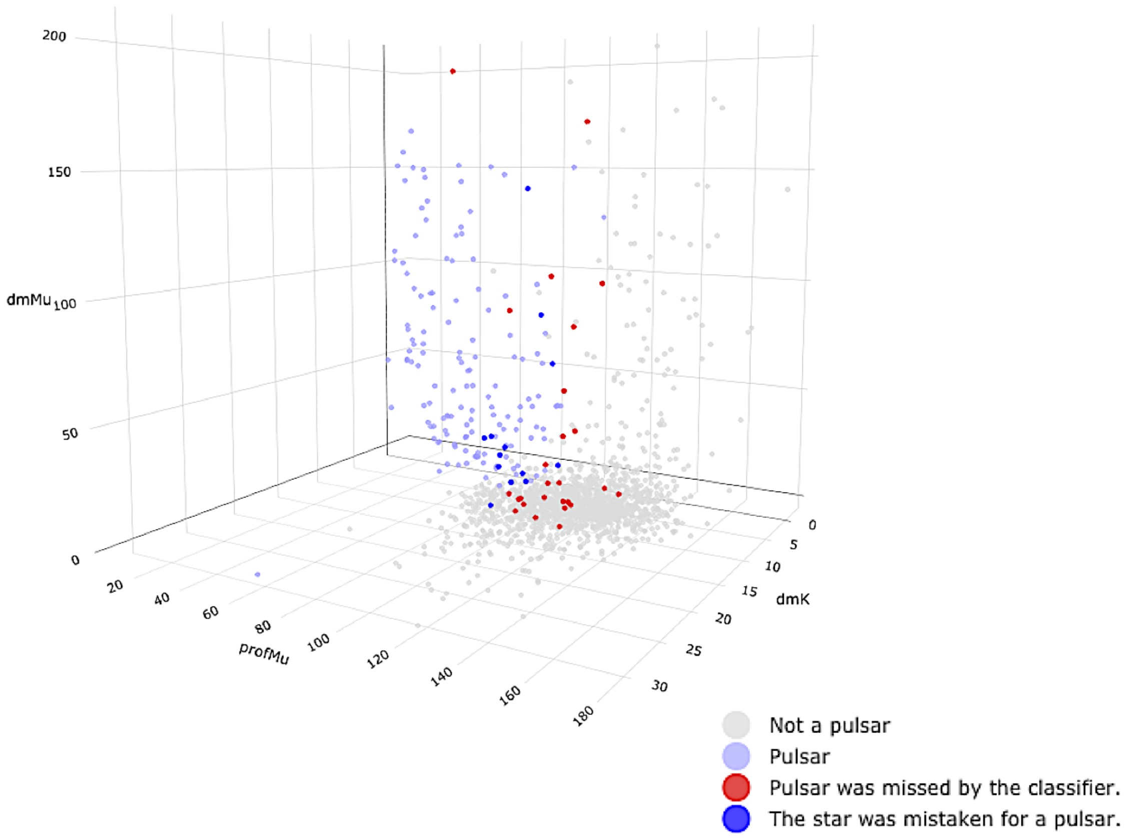
Conclusion

The machine learning methods allow us to classify different types of the objects with the ultimate precision. The recognition of the pulsar stars is one of the applications of the broadly accepted algorithms.

The ultimate condition of achieving the absolute precision goal is the ability to define features with grouped values.

The inability to separate the values of features into the groups creates the classification mistakes. In this case, the application of the different classification algorithms does not improve the quality of results.

In this project we achieved very impressive quality of results with fairly simple classification methods, we studied all the classification mistakes and achieved understanding of the errors origin.



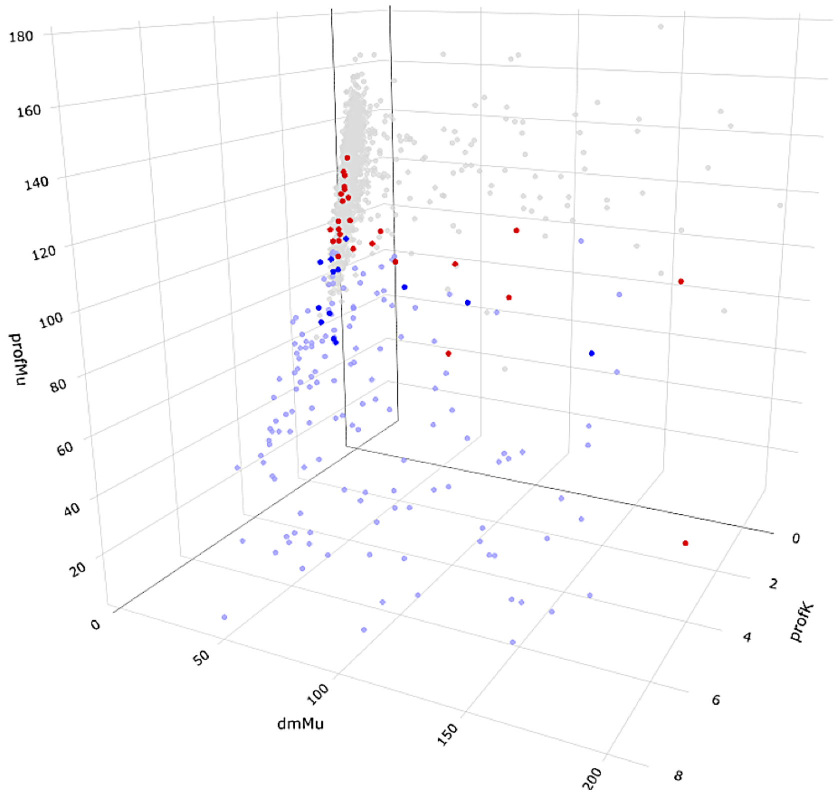


Figure 9: Erroneous observations on $DM_\mu, Prof_\mu, DM_K$ and $Prof_\mu, DM_\mu, Prof_K$ plots.

References

1. HTRU2 Data Set
2. John M. Ford, “Pulsar Search Using Supervised Machine Learning”
3. R. P. Eatough, N. Molkenthin, M. Kramer, A. Noutsos, M. J. Keith, B. W. Stappers, and A. G. Lyne, “Selection of radio pulsar candidates using artificial neural networks”
4. Ryan Shannon, “Pulsar Observation and Data Analysis”
5. Pulsar Timing
6. Chia Min Tan, “Using Machine Learning to Search for Pulsars with LOFAR”
7. Wikipedia: “Pulse Profiles”
8. Pulsar dispersion band
9. Ariel Goldberger, “Classifying pulsar stars using AI techniques”
10. Kuo Liu, “Introduction to Pulsar, Pulsar Timing, and measuring of Pulse Time-of-Arrivals”
11. Searching for and Identifying Pulsars
12. NASA: “Neutron Stars”
13. Wikipedia: “Pulsar”
14. Pulsar Properties
15. Duncan Lorimer ana Michael Kramer, “Handbook of Pulsar Astronomy”
16. Adam Deller, “Pulsars”
17. Gelu M. Nita, Aard Keimpema, Zsolt Paragi “Statistical discrimination of RFI and astronomical transients in 2-bit digitized time domain signals”
18. J.L. Han, R.N. Manchester, R.X. Xu, G.J. Qiao, “Circular polarization in pulsar integrated profiles”
19. Gelu M. Nita, Dale E. Gary, Zhiwei Liu, Gordon J. Hurford, and Stephen M. White, “Radio Frequency Interference Excision Using Spectral-Domain Statistics”