YSOC-VIS - A Visualization of the Classification of Young Stellar Objects

Lorenz J. Linhardt*

University of Vienna

ABSTRACT

This document introduces YSOC-VIS, a Java application for visualizing the classification of young stellar objects. Its purpose is to support interactive, visually guided model and feature selection. First the problem is introduced, then the general design and specific choices are discussed and lastly future improvements are suggested.

Index Terms: Visualization, Clasification, Feature Selection, Model Selection, Young Stellar Objects

1 MOTIVATION

The development of young stellar objects(YSOs) has been subject to recent research, and until now their classification has been conducted manually by astronomers. This involves looking at images of the stellar observations coloured with respect to different infra-red wavelength spectra, looking up the observation in color-color plots, where the strength of the infra-red emissions of single observations are pairwise plotted against each other, and looking at so-called spectral energy distribution curves for each observation. For researching YSOs it is important to know which stellar observations are YSOs and to which class they belong, but it is understandable that the large quantity of observations cannot all be manually labelled by humans; at least not in a reasonable time and with the current work flow. To combat this limitation and to free up additional resources, astronomers are trying to automate the classification process so that the majority of the work is done by machines and only unsure labels have to be reviewed by the domain experts. YSOC-VIS is designed to support the domain experts in finding a feasible classification model.

Users The target user of YSOC-VIS are astronomers without in depth knowledge of machine learning and whose interests lie in classifying stellar objects. At present there are the two specific astronomers at the University of Vienna who are the intended users and in collaboration with whom this tool is being designed.

Goals The users' ultimate goal is the classification of stellar objects based on data from various telescopes and data derived therefrom. They are interested in the distinction between the classes of YSOs and everything that does not belong to these classes. The goal of YSOC-VIS to support them in finding a suitable model for classification without having to posses extensive knowledge about machine learning. It is designed to enhance the current workflow of finding classification models by letting the user explore a space of multiple possible models instead of trying out different models one-by-one.

Data It has been attempted to formulate a data abstraction[10, chapter 2], in which the general properties of the available data are described.

The data is stored as table in a static file, where each table row, or item, represents a stellar observation and each column an attribute of the observations. There are about 800.000 rows and 200 columns, but only a small fraction of the observations is labelled, so that there are only a few hundred rows per class of YSO and only a few thousand classified as certainly not being a YSO. The attributes in the table are comprised of various data types, containing numeric types of different bit-lengths, single characters and character string. Most of them represent measurements at different infra-red wavelength regions and attributes derived from them. The contained attributes are of different types, covering both categorical and ordered data. Also not all attributes are semantically independent, e.g. for some columns there are additional attributes describing how uncertain their value is. For the proposed solution, additionally to this heterogeneous table, there is derived data, namely the models that are being built and especially their performance and confidence measures, which are represented as quantitative, sequential floating point numbers.

Notation To keep this document consistent and understandable, the word "observation" or "source" will denote a row of the data table and "attribute" a feature of a table row represented by a column. Also "parameter" will be used for the parameters and hyper parameters of the classification algorithm; in the current implementation this is the k of the k-nearest-neighbour algorithm, as this is the classification algorithm being used. "Variables" will stand for both parameters and attributes. Regarding classification, "model" will be used for the combination of a classifier, a configuration of its parameters, and a list of attributes being used for training.

Tasks It follows an attempt to generalize the tasks the user needs to perform, to provide a clearer picture of what has to be achieved and also as guidance for further design steps. The Task abstraction is based on [10, chapter 3] and abstractions proposed in this book are written in *italic*.

Consume: The user needs to *discover* which subset of attributes, and which parameters result in the most accurate prediction. To do this it is essential to be able to compare the performance of variables or models. This could be done by either listing them ranked by some performance metric or displaying their performance against a common scale.At least for the latter case, the user also needs to be able to filter out interesting subsets of the data for more visual clarity. This can be accomplished by either manually selecting parts of the displayed models or variables, or by restricting the range of parameters and attributes being considered for display.

Search: In the process of discovery, the user has to be able to *explore* the data, most likely looking for clusters, outliers or similar patterns. Finding patterns, even though this does not necessarily mean finding variables that perform well, can help the user by pointing out inherent properties of their data or its interaction with the classifier. This knowledge can be useful to steer the further investigation or even raise questions outside the scope of classification. Hence the dimensions in which patters could emerge and be of interest should be encoded in a way that makes it easy to spot those patters and it should be possible to find out what caused the pattern, e.g. by selecting the data points forming it, and showing details about them on demand.

Query: To enhance the exploration process, the VIS-system should provide a way to *summarize* the displayed data, providing an overview for the user, who can then look into the portion of the data they deem interesting. This can be accomplished by providing

*e-mail: pflaenzchen@hotmail.com

an additional view that aggregates the displayed data and provides a way to select or filter on aggregations to display. Also the user should be able to *identify* data points, e.g. models, returned by the search process, as they are ultimately interested in the properties that led to that item being selected.

Produce: Furthermore the VIS-system needs to *produce* an output, to be used as input for the classifier. This could be by *deriving* a list of attributes and model parameters that the user found to perform well.

2 Related work

Unsurprisingly there are already a number of different approaches to interactive visual classification proposed and implemented.

One interesting approach is StarClass[16]. This visualization tool represents the training data as points, coloured according to their class in star coordinates[8] and allows the user to construct classification rules, by dragging the coordinate axes and thus changing the projection of the data, until the classes are visually separated. Unfortunately StarClass only works for numeric attributes and scalability beyond 19 attributes has not been evaluated, but it seems unlikely that this approach would scale up to the amount attributes in the stellar observations dataset.

A different approach has been proposed for interactively constructing decision trees[3]. Again the authors lay out the attributes radially, but not in the form of axes, but as triangular segments. Pixel positions within each segment correspond to a value of the respective attribute, and the color of the pixel is determined by the class of the data point with the corresponding attribute value. The user can then iteratively split on segments were they see a separation of classes dependent on the attribute value. This approach also is likely to lead to scalability problems with large amounts of attributes.

The solution proposed in this document is built upon ideas and visualization concepts proposed by other authors, which are discussed in the rest of this section.

Scented widgets[18] are an enhancement of ordinary GUI widgets. The idea is to visually encode additional information directly at the widget, without changing the functionality of the widget and ideally without using up much more space than the base widget. A similar approach, but more specialized approach are Data Visualization Sliders[4], which specifically use the inside of sliders to provide a visualization of the data belonging to the slider.

The representation of uncertainty has been discussed and empirically analysed by a number of authors[9] . Often with a focus on discrete data points in an uncluttered environment. For those cases blurriness seems to be a good choice of encoding uncertainty. One way of encoding uncertain in two dimensions, given multiple estimations of the true value have been made, are standard deviational ellipses[17], where ellipses are fit to the variance of the data points in both scale and rotation, and additionally scaled by a factor to represent an arbitrary confidence interval. This approach has been chosen to represent uncertainty in respect to two performance measures in YSOC-VIS, as it allows to give the representation direction and also can easily be added to and removed from the visualization to adapt to clutter without changing the basis representation for the objects that are uncertain.

The idea of coupling widgets for conducting queries and visualizations has been around for quite some time[2] and the aspects of tight coupling have been discussed and implemented in the famous FilmFinder[1], which the proposed solution resemble to, even though the problem is not exactly the same. One of the ideas regarding tight coupling is that the state of the widget and the state of the visualization should always be connected, i.e. if the user changes the state of the widget, this should immediately be reflected in visualization and changing aspects of the visualization should promptly be propagated to the widgets corresponding to these aspects. Also interactions with tightly coupled components should be reversible and allow for incremental refinement of the results.

Another idea YSOC-VIS is incorporating is visual parameter space analysis and especially some ideas introduces as a conceptual framework[12]. In this framework a general abstraction of visual parameter space analysis is proposed in form of a data flow model and strategies for navigating the resulting data space are identified. YSOC-VIS uses some of the ideas presented in the framework, such as replacing a workflow based on informed trial and error, e.g. testing different classification models one-by-one and adapting to what worked, with an expensive, but fully offline process, consisting of sampling the space of possible inputs and generating corresponding outputs, which are then analysed in bulk, based on the inputs used to produce them. Another idea formalized in this framework and included in YSOC-VIS is the *global-to-local* exploration of the result space, starting from an overview of all generated outputs and from there on identifying and focusing on the more promising ones.

3 Methodology

The methodological approach roughly follows the steps suggested in[13], in which a nine-stage design study methodology framework has been proposed. The stages are grouped in three categories: *Preconditioning* is the phase of preparing for the design study and finding suitable collaborators. The *Core* phase is where solutions are developed in collaboration with the chosen domain experts, then implemented and evaluated. In the *Analysis* phase the knowledge won in the design study is being identified and brought into a structured form so that others may benefit from it.

Although this document is an artefact of the *Analysis* phase, the majority of work on YSOC-VIS has been conducted in the *Core* phase. Specifically in the *discover*, *design*, and *implement* stages.

Discover In the discovery stage an attempt was made in collaboration with the domain experts, to find out what the concrete problem is that has to be solved, how this problem can be abstracted to higher level problems, what data is available and who the potential user of YSOC-VIS are. This has been done via discussion with the domain experts.

Design In the design phase, paper prototypes of possible solutions were made and iteratively refined based on feedback from the domain experts. It was tried to first present a broad space of possible solutions to the domain experts and to then narrow it down through discussion until it finally converges to a potentially good solution.

Implement The result of the previous phase has been implemented in software and progressively improved through feedback from the domain experts and additional usability considerations.

The next steps would be to deploy the application and gather additional feedback, and after some further improvements, to evaluate its usefulness.

4 Approach

The high level concept of the proposed solution is to sample an user-constrained space of possible classification models, evaluate their performance and to visualize them dependent on this performance. The sampling is being done quite coarsely, as the space to cover is so large, that finer or even exhaustive sampling is not possible. After a seizable amount of models has been built, the user should be able to interactively explore the visualization and investigate the best performing models. Based on their investigation, the user can try to identify causes for good model performance, change the sampling constraints accordingly and rerun the sampling until they have found a satisfying solution.

What should be noted is that the sampling and evaluation of models is a very time consuming process, but it requires no further user

interaction, so the user does not have to invest time to actively take part in this step.

Furthermore the application should enable the user to take the same preprocessing steps they are conducting in their current workflow. That is, defining so called error-cuts on the input table. Error-cuts are simple rules that are used to filter out observation with undesired properties. An example would be disregarding all observations where the value of an attribute that is semantically associated with pollution of the observation, is above a certain threshold. These error-cuts are made by the experts based on domain knowledge.

In the current workflow these error-cuts reduce the data used for model selection dramatically - usually to a few thousand observations - which is due to the fact that the vast majority of the data is unlabelled and additionally significant cuts are made on noise and pollution attributes.

In this project, defining the class membership of observations is also seen as part of the preprocessing, since taking appropriate error-cuts is dependent on this definition.

Figure 1 schematically shows the proposed workflow steps. The steps with white boxes require user interaction, the one in a grey box is done offline.

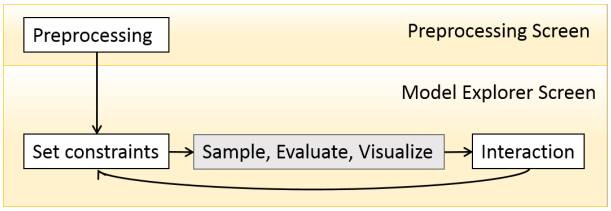

Figure 1: workflow

5 Implementation

The implementation has been done in Java, which was chosen for its portability and direct access to various toolkits. For the modules regarding machine learning the Weka toolkit[6] has been used. The GUI has been created using the GUI widget toolkit Java Swing and extended with plots made with JFreeChart[5].
Furthermore, since the .fits[11] file format, which has been chosen because it is already used in the users' workflow, is the format of the data table, the STIL and STILTS[15] libraries, that are specialized in handling this format, have been used.

6 Results

6.1 Preprocessing Screen

The preprocessing screen shown in figure 2 is the first part of the program the user comes into contact with when starting the application. After loading the data table from their file system, the user can define inputs to the preprocessing on the left side and see its results on the right side. These kinds of grouping of either functionality or semantics are done throughout the design, even though this is not always done by spacial grouping of elements.

The definition of the target class, which is the class that the resulting models should be able to identify, is done at the top of the left panel via a Boolean expression, as can be seen in figure 3. This form of input has been chosen, because it is simple and yet allows to define very complex subset selection. Furthermore this kind of subset-selection of tables is already used in topcat[14], a tool for editing tables of astronomical data, which is frequently used by the domain experts.

Consideration have been made to replace this form of input with a less error prone and more intuitive approach, but so far no input

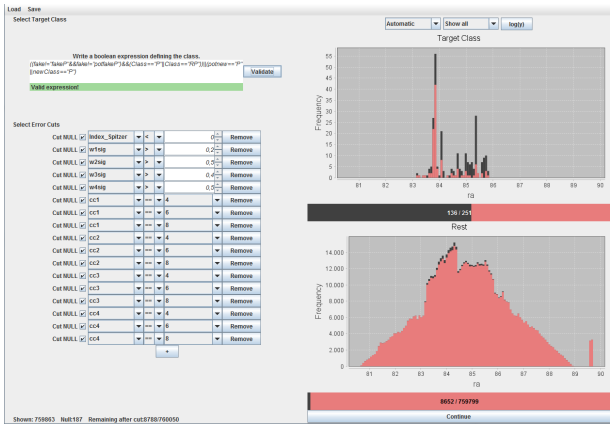

Figure 2: Preprocessing Screen

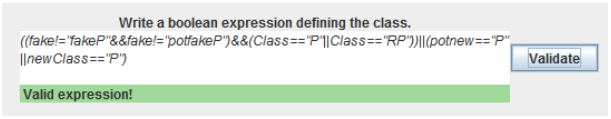

Figure 3: The area for defining the target class. The syntax can be checked for correctness with the button on the right.

interaction has been found that could easily be used to produce inputs of the required complexity for the given amount of attributes in the data, without dramatically increasing the time the user needs to spend on the interaction.

Figure 4: A row of GUI elements for defining an error-cut.

The majority of the left panel leaves room for defining error-cuts. This is done via rows of GUI elements, as shown in figure 4. In each row, the user can define a simple Boolean rule for filtering out observations. Observations that fulfil at least one of the rules are 'cut' from the table.

To define an error-cut, the user can select an attribute to base the error-cut on from a combo-box. Next to this they can select one of the following operators $<$, $>$, !=, ==, where the first two can only be chosen if a numeric attribute has been selected. Further on the right appears a text field where the user can type in a value for numeric attributes, or select a predefined value for nominal attributes, i.e. attributes that have been saved as character or character string in the table.

For each error-cut the user can choose to also cut away all observations that have an illegal, or NULL value for the selected attribute. This could be because the table cell is empty or the value in the table cell is set to a value defined as NULL-value in the file header.

Below the bottom most error-cut row there is a button for adding new error-cut rows. Also, next to each error-cut row there is a button for deleting it.

For the simple expressions used in error-cuts it is feasible to use more advanced GUI elements for defining the expression, as there are no complex connections between or groupings of statements. In this case restricting the input through the GUI elements adds the benefits that the user cannot make syntactically incorrect expressions and also for categorical attributes it is easier to select a value

from a predefined list than to type it.

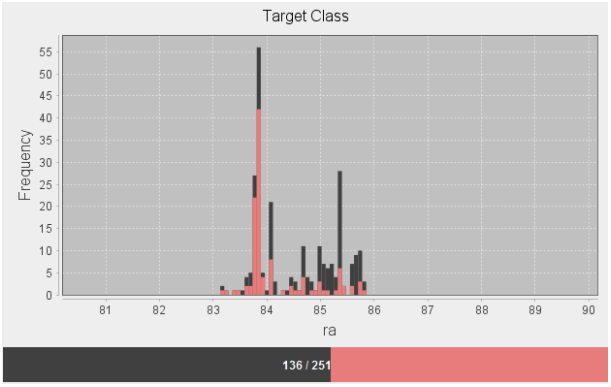

Figure 5: Histograms act as visual support for the error-cut generation.

On the right side of the preprocessing screen the user can see the distribution of observations over the attribute they are cutting on. Superimposed in red is the distribution over the same attribute, but just for the observations that have been cut-away. There is one histogram for the observations defined as target class and one for the rest.

Additionally there is a bar at the bottom of each histogram that shows the ratio of observations before and after the error-cuts. Again, the part that is cut-away is encoded in red.

The reason for splitting the histograms into target class and rest is that the target class, which is the sensitive part when creating error cuts, has usually only so few members, that it would be hard to make them visible in the same plot as the rest.

Also the additional bar below the histogram was placed to explicitly make it visible how many observations are remaining. What also plays a role here is that the domain experts already know roughly how many observations to expect after a cut, so by placing the exact number in the bar, it makes it easier for them to tell if the error-cuts they have made match up with what they usually do.

The plot for the target class has been placed above the other one, as it semantically belongs more to the target class definition, which is at the top of the left panel.

6.2 Model Explorer Screen

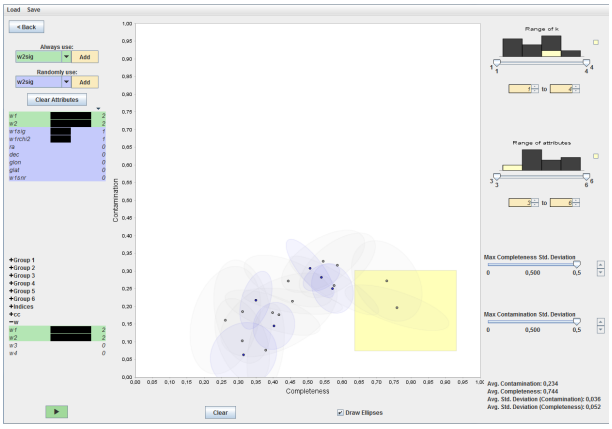

Figure 6: Model Explorer Screen

The model explorer screen is where most of the interactions take place. It contains widgets that allow the user to steer the sampling process and to filter out uninteresting models. All widgets that change the sampling process are coloured in the same orange tone. This color marking has been done to signal that those widgets form some kind of unit, even though their placement is not grouped.

Most prominently is the performance plot in the middle. Here all generated models are plotted according to their performance.

6.2.1 Attribute Panel

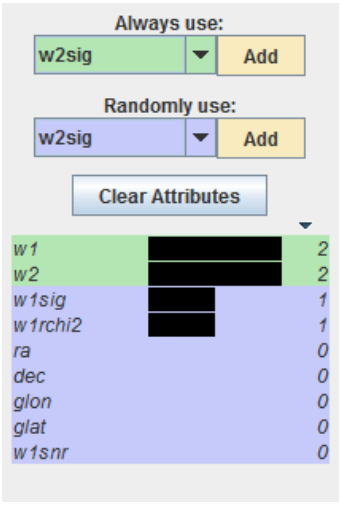

Figure 7: The selection of attributes to be sampled from.

The attribute panel on the left side of the model explorer screen, as seen in figure 7, enables the user via combo boxes to define which attributes should always be used in the the creation of models and which should be randomly sampled from. Any attribute for which the user does not specify the sampling behaviour is never used in the sampling process. All attributes for which the user has defined the sampling behaviour are listed below the combo boxes. The background color for each attribute signifies if it was added to the 'always use' or the 'randomly use' list. This is again an attempt to visually link elements that have a connection - in this case they are sharing the same state. The user can also change the list membership of any attribute displayed via a context-menu, activated via the right mouse button. As soon as models are being created, a bar appears next to each attribute in the list, encoding how often the attribute has been used in all created models. This is also explicitly shown as a number next to the bar.

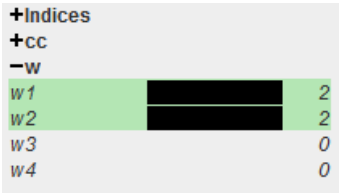

Figure 8: Semantic groups for easier navigation selection of attributes.

In figure 8 we can see the lower part of the attribute panel. This part only appears if the user loads a file in which attribute groups are specified. As the application cannot automatically detect those groups and it is not made to only work with one single, never changing set of attributes, the user has to specify semantic groups and attribute memberships to these groups manually in an external file.

Each group can be expanded and collapsed on click and contains a list of attributes like the one describe previously. The reason for using semantic groups is that they make it easier for the user to find specific attributes than looking through all possible attributes in the combo-box and they allow to specify the sampling behaviour for whole groups at once via a context menu.

There are two additional buttons in the attribute panel which are not connected to attribute selection. One is the 'Play' button at the bottom left, which starts the sampling process. Its placement at the bottom of the screen has been chosen because in western societies humans are used to doing things from top to bottom, and starting the sampling is not something users would want to start their interaction with. So it makes sense to put the button at the 'end of the page' and create an environment with a more familiar logic.

The placement of the second button at the top left has similar reason. It is the 'Back' button, leading to the preprocessing screen. Its placement to the left is due to the fact that at least in western societies, humans are used to progress from left to right and also most applications and web-pages place back-buttons on the left side. The choice of vertical alignment was made to avoid that users accidentally hit the back button when they actually want to start or stop the sampling.

6.2.2 Performance Plot

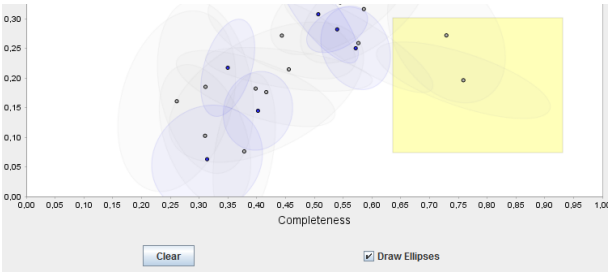

Figure 9: A section of the performance plot showing models as points with standard deviational ellipses and additional widgets for disabling the ellipses and deleting all contents of the plot.

Most of the area of the model explorer screen is covered by the performance plot as this is where smaller differences in the position of the marks are important. The the axes of the plot are 'Completeness', which equals to the probably more common *recall*, and 'Contamination', which corresponds to 1 minus *precision*. These measures have been chosen because the domain experts are already familiar with them and they are common in the astronomy community[7, p. 387].

When the user has started the sampling process, models will be created one-by-one under the user defined constraints and evaluated via cross-validation. They are then placed as point-marks in the performance plot (see figure 9) where their position encodes their performance and their color encodes the uncertainty of their position. Red stands for high uncertainty, gray for medium and blue for low uncertainty. Ideally the encoding would have been done by using a channel with intrinsic order, but as luminance and saturation are hard to distinguish for points so small and variability in size would increase clutter if many point are plotted, the most important channels have been ruled out and currently hue seems to be a reasonable choice as it is distinguishable and does not add to clutter.

Furthermore, standard deviational ellipses are shown around each point to signify a confidence interval in which the model would most likely lie if it would be evaluated again. These variances in performance for one model are due to the randomness involved in cross-validation.

As the ellipses tend to quickly clutter the plot, the can be turned off via a check box. Also the points are drawn with a black border, which makes them more visible against cluttered background.

The only direct interaction with the plot that is provided is the ability to draw a frame to select multiple models. This selection has an effect on many widgets. One example is the list on the attribute panel that only shows the attributes and their frequency for the models in the selection.

6.2.3 Filter Panel

The filter panel resides on the right side of the model explorer screen. It mainly allows for filtering the performance plot, but also for setting some of the sampling constraints, as indicated by the orange coloured widgets, and it also displays some data about the selected models.

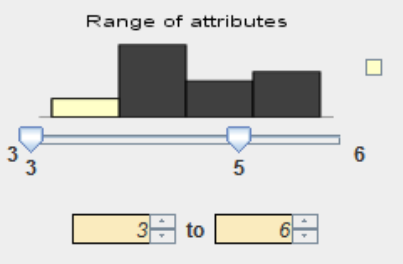

Figure 10: Semantic unit regarding the length of the attribute vector.

There are two widget groups as presented in figure 10. One for the range of the parameter k and one for the length of the attribute vector to create models from. Their spatial grouping should signify that they form a unit. The spinners at the bottom can be used to set constraints for the sampling process regarding this parameter. The range sliders can be used to filter out models not falling within a certain value range of the parameter represented by the group. Those models are not being displayed in the performance plot. The sliders are additionally enhanced by a histogram showing the distribution of the created models over the parameter.

The rational behind including the histogram was that it should make it easier to identify anomalies if they can be discovered in a histogram, rather than by moving the slider back and forth and observing how the performance plot changes.

Superimposed over the distribution of all the models is the distribution of the selected models in yellow. The color was chosen to match the color of the selection rectangle in the performance plot, so that the user can more easily see a connection between them.

Also it was decided to keep the overall distribution as context to make it easier to spot anomalies in the selection.

If the overall distribution is much higher than the distribution of the selection, the user can choose to sacrifice context for a clearer view on the selection and use the small yellow button next to the histogram to only show the yellow bars. The functionality of the button might not be clear at first, but again, the color should serve as a hint. Clicking the button also turns it dark gray to indicate a connection to the hidden bars.

The yellow bars of the histogram are framed in black to increase their visibility against the background, as this can be an issue on some screens.

Below the widget groups there are sliders to filter out models by the standard deviation of the performance of their cross validation results. They make it possible to hide models for which the performance estimation is too uncertain.

At the bottom of the filter panel there is the performance of the selected models displayed in concrete numbers. If no selection has

been made on the performance plot, the shown metrics are the average of all models.

Performance and Feedback As YSOC-VIS is still work in progress, the performance of the tool has not yet been evaluated and there were only minimal interactions of the users with the tool. Gathering results is one of the next steps in the development process.

Still there are some suggestions by the domain experts, most of which are not implemented in the current version of the tool, but are planned to be introduced in the future.

Probably the most important suggestion is the addition of views in which the domain experts can see the model evaluation results in views they are used to, and in relation to the original, real-world attributes contained in the data table.

Also it has been requested to add the functionality to derive attributes from already existing ones. This is due to the fact that the domain experts often work with the difference of magnitude between wavelength regions, which is not explicitly saved in the data table.

Feedback from the domain experts also led to the reconsideration of some notational choices, such as the discussed labelling of the performance plot axes.

Some observations about the current scalability of the tool have been made, although no rigorous testing has been conducted, as optimization of performance was not yet the focus of development. The model evaluation process works in reasonable speed for at least up to 10.000 observations remaining after the error cuts. It has to be noted though, that the time needed to evaluate a model is very much dependent on the size of the attribute vector that is used for training and also on the configuration of the cross validation, which is currently defined in the code as one run of ten folds. Also it has been reported that using 50.000 observations leads to infeasible model evaluation times of several minutes per model.

The amount of models already being built and displayed does not impair the interactions up to at least 3.000 models, given that the confidence ellipses are not being displayed.

7 Discussion

7.1 Strengths and Weaknesses

There are still many aspects of the tool that await implementation or improvement. Also, the list of strengths and weaknesses presented here will probably be subject change when handing the tool to the user.

The preprocessing screen will most likely have to undergo some changes, as it is a tedious process to define error cuts and creating a class definition much resembles to coding. The widgets used also provide no mental model for the actual value ranges. Even though it is obvious that this part of the tool is not an optimal solution, providing a light and efficient input interface for complex expressions is not a trivial task and improvements will be made in the future.

Another weakness is that there is still no view on the original attributes the domain experts are familiar with. Without those views it is hard to mentally set the model performances into the context of the real problem and observable patterns in misclassifications might just not be found.

Furthermore the low performance of the tool is definitely a weakness. Especially in the preprocessing screen there are noticeable delays when updating the plots, but also in the model explorer screen, although not visible from usability standpoint, higher performance would increase the amount of models built per time-step.

Also it is not possible to classify unknown samples wit a chosen model, as the current focus is exclusively on model and feature selection and not on the real-world application of the model.

Moreover, k-nearest neighbour is the only classification algorithm available to the user at the moment. As other algorithms might yield better results, this is a severe limitation.

On the positive side, YSOC-VIS uses a quite general approach and is mostly detached from the original domain, although it is being developed with a target application and dataset in mind. This makes the tool usable in other areas too, if it is found to be useful.

Also, even though it might not be 'intuitive' at first, the interactions in the model explorer screen are already in a state that makes it easy to manipulate performance plot. Although there are still refinements to be done, the current usability could be considered a strength.

The tool also allows for quickly trying out different attribute sets and parameter choices, so even if it is used to experiment with single model choices at a time, this should be quicker than making those tests in code.

7.2 Lessons Learned

One thing about the implementation of projects like this that became more and apparent is that writing clean code really is not that important, as most modules are frequently subject to change anyway. However, building a solid architecture from the start would have helped performing later changes more quickly.

Another aspect to consider is that conducting rigorous task abstraction and identification of the goals of the project early on is immensely important. If the task is unclear, much effort is wasted.

Also, while keeping a tight feedback loop might be something obvious, one should not underestimate the improvements possible through this practice. What was new to me in this regard is that the people to gather feedback from do not necessarily have to be potential users of the system. Showing the tool to non-users also catches some usability aspects that that would have stayed undetected otherwise.

Moreover, gathering feedback in general is not always an easy task. Especially if the people giving feedback are not used to evaluating software. Overcoming *experimental demand characteristic* effects[13] where one would get mostly positive feedback when working with the domain experts for a longer time, is also something to watch out for.

Furthermore, especially if one is new to the field, planning enough time for literature research is crucial. As there are many handy ideas already implemented and building upon knowledge of others is the only to avoid reinventing the wheel.

8 Future Work

As this project is ongoing, there are still many steps to be taken, some of which are listed in this section. To provide more clarity, the future steps are divided into implementation and analysis steps, though it has to be noted that this division is only to provide structure and does not mean that these two categories are independent of each other.

Also continuous improvement of the usability and the underlying abstractions is implied.

Implementation Steps One major change that is planned, and based on feedback of the domain experts, is the introduction of additional views, where the user can see the distribution of correct and incorrect classifications plotted against an attribute of their choice. Also they should be enabled to view the results of the classification in the color-color plots they are used to. These views could help identify patterns in the misclassifications, which in turn could lead to different choices of error-cuts and sampling constraints.

Furthermore it is planned to let the user control the evaluation precess, especially the number of folds and the the number of cross validation runs used.

Also the sampling has to be improved, as it currently just uniformly samples the space of variables. More sophisticated sampling strategies could yield better results and using heuristics to first sample the most promising portion of the model space could accelerate the workflow.

Finally, optimization of the application should be conducted to increase the usability through better responsiveness and to accelerate the sampling and evaluation of models. This will be done by refactoring the code and additional exploitation of parallelism.

Analysis Steps One of the next steps to be taken is to deploy the YSOC-VIS to the domain experts and gather feedback about this early use of the tool. This will show if the work that has been done until now was successful and also provide direction for future improvements.

Eventually there will also be a more formal evaluation of the tool to determine if it really enhances the workflow of model selection in this context.

It is also planned to create a polished reflection about the project, the methods used and findings made by refining and improving on this document in the future.

REFERENCES

[1] C. Ahlberg and B. Shneiderman. Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, pages 313–317, New York, NY, USA, 1994. ACM.

[2] C. Ahlberg, C. Williamson, and B. Shneiderman. Dynamic queries for information exploration: An implementation and evaluation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, pages 619–626, New York, NY, USA, 1992. ACM.

[3] M. Ankerst, C. Elsen, M. Ester, and H.-P. Kriegel. Visual classification: An interactive approach to decision tree construction. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 392–396, New York, NY, USA, 1999. ACM.

[4] S. G. Eick. Data visualization sliders. In *ACM Symposium on User Interface Software and Technology*, pages 119–120, 1994.

[5] D. Gilbert. JFreeChart. http://sourceforge.net/projects/jfreechart.

[6] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[7] Ž. Ivezić, A. Connolly, J. Vanderplas, and A. Gray. *Statistics, Data Mining and Machine Learning in Astronomy*. Princeton University Press, 2014.

[8] E. Kandogan. Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In *In Proceedings of the IEEE Information Visualization Symposium, Late Breaking Hot Topics*, pages 9–12, 2000.

[9] A. M. MacEachren, R. E. Roth, J. O'Brien, B. Li, D. Swingley, and M. Gahegan. Visual semiotics & uncertainty visualization: An empirical study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2496–2505, 2012.

[10] T. Munzner and E. Maguire. *Visualization analysis and design*. AK Peters visualization series. CRC Press, Boca Raton, FL, 2015.

[11] W. D. Pence, L. Chiappetti, C. G. Page, R. A. Shaw, and E. Stobie. Definition of the Flexible Image Transport System (FITS), version 3.0. *Astronomy & Astrophysics*, 524:A42+, 2010.

[12] M. Sedlmair, C. Heinzl, H. Piringer, S. Bruckner, and T. Möller. Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization and Computer Graphics / Proceedings IEEE InfoVis 2014*, 20(12):pp. 2161–2170, 2014.

[13] M. Sedlmair, M. Meyer, and T. Munzner. Design Study Methodology: Reflections from the Trenches and the Stacks. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis)*, 18(12):2431–2440, 2012.

[14] M. B. Taylor. TOPCAT & STIL: Starlink Table/VOTable Processing Software. In P. Shopbell, M. Britton, and R. Ebert, editors, *Astronomical Data Analysis Software and Systems XIV*, volume 347 of *Astronomical Society of the Pacific Conference Series*, page 29, 2005.

[15] M. B. Taylor. STILTS - A Package for Command-Line Processing of Tabular Data. In *Astronomical Data Analysis Software and Systems XV - ASP Conference Series*, volume 351, pages 666–669, 2006.

[16] S. T. Teoh. StarClass: Interactive Visual Classification Using Star Coordinates - CiteSeerX. *Proceedings of the 3rd SIAM International Conference on Data Mining*, 2003.

[17] B. Wang, W. Shi, and Z. Miao. Confidence analysis of standard deviational ellipse and its extension into higher dimensional euclidean space. *PLoS ONE*, 10(3):e0118537, 2015.

[18] W. Willett, J. Heer, and M. Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 13:1129–1136, 2007.