

A Virtual Research Assistant for the ATLAS transient sky survey

Technical Manual **VRA v1.1**

April 11, 2025

Dr. Heloise Stevance



Contents

Preface

5

I Production Systems

6

1 Quick References

7

1.1 Current VRA

7

1.1.1 Policies

7

1.1.2 Triggers to Mokoodi

7

1.1.3 Features

8

1.2 Eyeballer Cheat Sheet

9

1.2.1 First Eyeballer

9

1.2.2 Second Eyeballer

9

1.2.3 Reminders

9

1.3 Dev Checklists

10

1.4 Change History

11

1.4.1 VRA 1.1 - 2025-03-06 - Duck 1.1

11

1.4.2 VRA 1.0 - 2025-02-03 - Duck1.0

11

1.4.3 VRA beta0.1 - 2024-12-06 - Crabby

12

1.4.4 VRA alpha0.1 - 2024-13-08 - BMO

12

2 Data Processing

13

2.1 ATLAS data processing pipeline

13

2.1.1 On-site Processing

13

2.1.2 Centralise processing: Hawaii

13

2.1.3 Centralised processing - Queen's University Belfast

13

2.2 VRA data structures

15

2.3 VRA Packages and environments

16

2.4 Recording Passive Feedback

16

II First data and code release: Crabby and Duck

18

3 Duck - Dealing with auto-garbage

19

3.1 Duck 1.1 - Fixed

19

3.1.1 In Production Performance

20

3.2 Duck 1.0 - How NOT to deal with the Auto-garbage

22

3.2.1 Summary

22

3.2.2 The Data

22

3.2.3 The auto-garbage mishandled

23

4 Crabby - Finer Sherlock features	25
4.1 Summary	25
4.2 The data	25
4.2.1 Re-eyeballing	27
4.3 Models and rankings	29
4.3.1 Hyperparamter tuning	30
4.3.2 The start of the new eyeballing strategy	30
 III Early prototypes: Arin and BMO	 32
5 BMO - The first production VRA	33
5.1 Raw data, cleaning and making training data	33
5.1.1 Jupyter notebooks locations	33
5.1.2 Data Overview	34
5.1.3 Balancing the data sets	35
5.1.4 Split into train and test sets	35
5.1.5 Features	35
5.2 Day-1 Scoring Models	36
5.2.1 Model description	36
5.2.2 Out of the box performance	36
5.2.3 Hyper-parameter Tuning	38
5.2.4 Tuned Scoring Models	41
5.2.5 Conclusions	42
5.3 Ranking	44
5.3.1 Effect of varying the fudge factor	44
5.3.2 Conclusions	45
5.4 Day 1 Models In production performance	46
5.4.1 Are the real scores different from the Real Bogus scores from the CNN?	46
5.4.2 Looking at some individual objects	48
5.5 Update Scores Models	49
5.5.1 Features	49
5.5.2 Some discussions	49
5.5.3 Results	51
5.5.4 Permutation Importance	52
5.5.5 What Next?	53
5.6 Updates to scores and ranks - testing the new eyeballing strategy	54
5.6.1 Selecting the VRA rank threshold to eyeball	54
5.6.2 Selecting automated garbaging conditions	54
5.6.3 Eyeballing strategy performance	55
5.6.4 Eyeballing Strategy for Galactic Transients	56
5.7 Further Feature analysis	57
5.7.1 Using the Galactic latitude as a feature	57
5.7.2 Culling features in day 1 model	57
5.7.3 Retraining Update Scores Models including day 1 features	58
 6 Arin - benchmarks and first tests	 59
6.1 Raw data gathering and cleaning	59
6.2 Exploring Data and Setting Benchmarks	61
6.2.1 Alert type fractions	61
6.2.2 Human decision timescales	61
6.3 Training set features	62
6.3.1 Creating Light Curve history features	62
6.3.2 List of features used for training	64



- 6.4 Assigning "real" and "galactic" scores 65
 - 6.4.1 First model 65
- 6.5 Ranking the alerts 66
 - 6.5.1 Ranking Algorithm 66
 - 6.5.2 Performance in train and test set 67
 - 6.5.3 Performance in production 67

Preface

This is a technical manual to the ATLAS Virtual Research Assistant and history of the development of the VRA. It is mostly a resource for developers and eyeballers, although for external collaborators it can also be supplementary information on the development of the VRA - information that either does not fit in the paper or is not part of a public data release. For all other resources (codes, papers, data, talks) related to the VRA, please check the ATLAS [VRA Zenodo community](#).

Note that the order of the Chapters relating to the VRA versions is **inversely chronologic**. This allows users of this manual to have access to the most recent information at the top of the file.

This research and development of these systems was supported by Schmidt Sciences.

Part I

Production Systems

1

Quick References

1.1 CURRENT VRA

1.1.1 Policies

- Extra-gal eyeball list threshold = 7
- Galactic candidate == True if within 0.4 of coordinate (1,1) (fudge factor = 0.9).
- On day1 Garbage anything with VRA rank < 1.0
- On the second visit garbage anything with maximum VRA rank < 2
- On subsequent visits garbage anything with mean VRA rank < 3 .
- Purgatory sentinel runs everyday
- Weekly report runs once a week on Friday
- TNS x-match runs once a week on Friday

1.1.2 Triggers to Mokoodi

Since the end of 2024 Fast Track objects with **VRA scores** > 8.5 (reduced from 9 on 2025-02-28) are added to the Mokoodi custom list (if their declination is below +15 degrees). This allows the Mokoodi telescope to check their custom list with a cron job and automatically trigger.

1.1.3 Features

The full list of features currently used by the day 1 and day N models are listed below in Table 1.1.

Table 1.1: [Features](#) used by the day 1 and day N models - the column name are given as they are in the code so that technical users can reference this Table.

Model	Column name	Type	Description
day 1 + day N	Nnondet_std	float	Standard deviation of the number of non detections between detections
day 1 + day N	Nnondet_mean	float	Mean of the number of non detections between detections
day 1 + day N	magdet_std	float	Standard deviation of the magnitude of the historical detections
day 1 + day N	DET_Nsince_min5d	float	Number of detections between phase -5 days and day 1
day 1 + day N	NON_Nsince_min5d	float	umber of detections between phase -5 days and day 1
day 1 + day N	log10_std_ra_min5d	float	log 10 of the Standard deviation of the RA in the detections from phase -5 days
day 1 + day N	log10_std_dec_min5d	float	log 10 of the Standard deviation of the Dec in the detections from phase -5 days
day 1 + day N	ra	float	RA
day 1 + day N	dec	float	Dec
day 1 + day N	rb_pix	float	Real/Bogus Score from the CNN from [Weston et al., 2024]
day 1 + day N	z	float	Spectroscopic redshift (if known, else NaN)
day 1 + day N	photoz	float	Photometric redshift (if known, else NaN)
day 1 + day N	ebv_sfd	float	Extinction E(B-V) calculated using <code>dustmaps</code> SFD
day 1 + day N	log10_sep_arsec	float	log 10 of the projected separation between the best matched source (in arcsec)
day 1 + day N	CV	bool	(sherlock classification) if CATAclysmic Variable
day N only	max_mag	float	Maximum (median) magnitude seen since phase -5 d
day N only	max_mag_day	float	Day of the maximum magnitude
day N only	DET_N_total	float	Number of detections since phase -5 d
day N only	NON_N_total	float	Number of non detections since phase -5 d
day N only	DET_mag_median	float	Median magnitude of the detections since phase -5 d
day N only	NON_mag_median	float	Median magnitude of the non detections since phase -5 d

1.2 EYEBALLER CHEAT SHEET

Also note there is an [ATLAS Eyeballing Handbook](#)

1.2.1 First Eyeballer

Make sure you are in the #vra channel on slack. Follow the links sent by the st3ph3n on that channel. There are different levels of priority:

1. Fast Track list: Eyeball Now
2. Extragalactic candidates: Eyeball ASAP (within 1h)
3. Galactic candidates: Eyeball within 24 to 48h
4. Eyeball the purgatory list when you can

Obviously these timings apply for when you're awake - don't be setting alarms during the night.

Some tips for Eyeballer # 1:

- Start with the Fast Track list. Also note that you need to **eyeball all ranks** (even the lower ones) in the Fast Track list.
- Go back to slack and click on the Extragalactic eyeball link provided by st3ph3n. This will take you to the curated eyeball list (ordered and cropped down to the eyeballing threshold).
- Scroll down to the first alerts that or not rank 10 and eyeball these. They are the most promising which haven't yet been pinched by another team.
- Some alerts are ambiguous when there is not a lot of data. If you're not sure what to do with them you can **ask for other opinions on slack** or **snooze** with the Possible list. These alerts will come back to eyeball when the telescopes have looked at that part of the sky again.
- The ATLAS ingest cycle is faster than it used to be so you may get alerts every 30 minutes when there is not a lot of data being ingested. If you do not snooze the data you will keep getting poked by st3ph3n even if nothing has changed.
- Clean up the rank 10 alerts.
- Now go do the Galactic candidates. If you are in a rush you can leave these for now and only do them once or twice a day.

1.2.2 Second Eyeballer

Check the good list for anything interesting to put in follow-up (particularly check the last week of objects). Alert the relevant people if rapid follow-up needed, or keep the source in your own list for later discussion.

1.2.3 Reminders

- CVs do not go in TNS if you're not sure if something is galactic snooze to wait for more data or ask on slack.
- Nuclear transients only go in TNS if you think they are TDEs. AGNs do not go to TNS
- The Attic/Galactic list is only for real galactic events. Bad subtractions due to high proper motion stars go in the HPM list.

1.3 DEV CHECKLISTS

Preliminary steps

- ☐ Push all new code to atlasvras github
- ☐ Make sure the new models have been copied over into the atlasvras/st3ph3n/models directory.
- ☐ PAUSE THE INGEST
- ☐ Pull the relevant branch on the server
- ☐ If the eyeball threshold has changed, update `eyeball_threshold` in the `data/bot_config_MINE.yaml`.
This needs to be updated locally as this file is not pushed to GH since it contains tokens

Move current VRA to OLD VRA codes

- ☐ Copy `initialiseVRA.py` to `OLDinitialiseVRA.py`
- ☐ Copy `updateVRA.py` to `OLDupdateVRA.py`
- ☐ Does any code need to change in `initialiseVRA.py` and `updateVRA.py`, e.g cropping some features that are now calculated in preprocessing but were not before?
- ☐ Check that the `OLDinitialiseVRA.sh` and `OLDupdateVRA.sh` are going to work on the python codes, e.g. have the inputs changed?

Test the new VRA - the test directory is located in `scripts/utils/python/test_vra`

- ☐ Copy `initialiseVRA.py` to `/test_vra/initialiseVRA.py` (may be the same as prod but check that you didn't edit things in prod directly)
- ☐ Copy `updateVRA.py` to `/test_vra/initialiseVRA.py`
- ☐ In `/test_vra/initialiseVRAScoresTest.py` and `/test_vra/updateVRAScoresTest.sh` change the version of the `model_name` parameter in `ScoreAndRank` object
- ☐ Apply any other change required.
- ☐ Run `/test_vra/initialiseVRAScoresTest.sh` with the options `RBSCORECSV [test_vra.csv]`, `RBTHRESH-OLD [0.2]`, `RANKCOLUMN [rank or rank_alt1]`
- ☐ Debug
- ☐ Run `/test_vra/updateVRAScoresTest.sh`. **Note: if there are not new lightcurve data since the previous ingest the features won't be calculated and you'll get a bunch of rank 10 cross matches and nothing else.**
- ☐ Debug
- ☐ When everything runs and you are ready to put in prod, clean up the `tcs_vra_scores` and `tcs_vra_todo` table to remove what you've added during your tests

Put new VRA codes in production

- ☐ Copy `/test_vra/initialiseVRA.py` to `initialiseVRA.py`
- ☐ Copy `/test_vra/initialiseVRA.py` to `updateVRA.py`
- ☐ If Auto-garbageing policies have changed, make relevant changed to `VRAGarbageCollection.py`

Check the code works

- ☐ Check the logs to make sure there are not errors in the first few days of ingest after you deployed the code!

1.4 CHANGE HISTORY

1.4.1 VRA 1.1 - 2025-03-06 - Duck 1.1

2025-03-28: realise there was a bug (dropped the column DET_mag_median) and the *day N* code didn't run properly! Fixed now.

2025-04-01: The auto-garbage was applying the "visit 3" logic to all objects because of two bugs: the logic read ≥ 2 for the visit count (instead of > 2) and the garbage collector counted the rows from the old AND new model so and day 1 each object has ≥ 2 visits... Fixed now

For more information see Chapter 3 and the code release.

- Although the clean data we use is the same, the training and validation sets have changed from Duck. Crabby data set is unchanged all changes below apply to the additional data gathered between august and january:
 - The training set no longer includes "guessed" labels.
 - The training set is balanced with 1600 auto-garbage and 400 garbage examples (for a total of 2k additional bogus examples). That subdivision reflects the fraction of auto-garbage and garbage in our additional data.
- When training the galactic classifier, the garbage and auto-garbage are no longer used during training. Instead only the PM, Galactic and Good alerts are used.
- When ranking the scalar (or fudge factor) has been changed back to 0.5.
- The Galactic flag claculation has been changed: the distance is now 0.4 (instead of 0.45) and the fudge factor on the galactic axis is 0.9 (instead of 1).
- The extragal eyeballing threshold is now 7
- The day 1 and day N feature DET_mag_median_min5dhas been added back in. In the previous training notebooks it looks like it was tried and abandoned but I found here that its permutation importance is non negligible for both day 1 and day N models.
- day N feature DET_mag_medianadded back in. Marginal gains but also consistent with keeping NON_mag_medianand removing both is worse.

1.4.2 VRA 1.0 - 2025-02-03 - Duck1.0

For more details check Chapter 3

- VRA real and galactic models trained on the Duck data-set. Ranges from 27-03-2024 to 22-01-2025
- The following day1 features were pruned: SN, ORPHAN, NT, UNCLEAR
- The following dayN features were pruned: SN, ORPHAN, NT, UNCLEAR, DET_N_today, NON_N_today, DET_mag_median
- Eyeballing now split into extragalactic and galactic
- Eyeball list now named extragalactic candidate list (although purgatory objects still linger at the bottom of that list)
- Extra galactic candidates defined as rank > 7.5 (instead of 4)
- Galactic candidates defined as within 0.45 of the top right corner of score space (fudge-factor = 1 instead of 0.4 used for the ranking used as threshold for the extra galactic candidates and the auto-garbage).

- ScoreAndRank now has the `is_gal_cand` property (boolean flag)
- Auto-garbaging strategy modified: day1: `max_rank < 1` (was 1.5), day2: `max_rank < 2` (was 3), day3+ : `mean_rank < 3` (unchanged).
- Number of alerts reported by slack-bot no longer overestimated
- el01z weekly reports and purgatory sentinel now run on the ATLAS server (Fridays)
- TNS cross-matching to garbage list done weekly on Friday mornings

1.4.3 VRA beta0.1 - 2024-12-06 - Crabby

For more details see Chapter 4

- VRA real and galactic models trained on the Crabby data-set. Ranges from 27-03-2024 to 13-08-2024
- Fudge-factor changed to 0.4 (from 0.5)
- added new sherlock features: `z`, `photoz`, `log10_sep_arsec`, `ebv_sfd`

1.4.4 VRA alpha0.1 - 2024-13-08 - BMO

First VRA in production. For more details see Chapter 5

2

Data Processing

The V.R.A. system is not just one piece of code running in isolation. It is comprised of new data structures and processes that have been added to the ATLAS transient database and data processing servers. This chapter summarises the journey of the data through the pipelines and the new data structures added to ATLAS. For exhaustive details on the ATLAS Transient Server check [Smith et al., 2020].

2.1 ATLAS DATA PROCESSING PIPELINE

2.1.1 On-site Processing

Just the data reduction

2.1.2 Centralise processing: Hawaii

Difference imaging and the processing that creates the measurements in the .ddc files are done in Hawaii. Note that the diffs are done not on the full ccd but on an 8x8 grid. There are 16 amplifiers in a ccd so 4 of those diff cells (in a column, not in a square).

The .ddc files are just that table on the webserver; text files that come down in addition to the images. **All 5 sigma detections in a field will have a row in the ddc file.** There is one ddc file per diff. Both the diff and associated ddc file are created on site and downloaded by the QUB servers.

2.1.3 Centralised processing - Queen's University Belfast

Subsequent data processing for all facilities occurs centrally at Queen's University Belfast (QUB).

ATLAS_ID: 19 digit integer which uniquely identifies an object in the database. As much as possible (with the need to be unique), it is the on sky coordinates of the object.

THE MEGA SHELL SCRIPT: `atlas_ingester_dbc_cron.sh`

This is the main script that calls all the pieces of pipeline. It is set to run **every 30 minutes**. Note that if the previous run is still going (can check processing status, see below) it won't start a new run. So at worst the server is idle for 29min 59sec.

1. **Record current time stamp** and select how many days prior we want to check for ingest. This flag is usually 0 as we're only checking today's data. Ken will occasionally change it if there has been a delay in the ingest.
2. **Run** `applyAtlasCutsDDCMultiprocess.py` which applies the initial cuts in parallel. **TODO: Get full list of cuts from Ken - note that will change with nnc files**
3. **Record processing status** for bookkeeping. 1=Starting, 2=Finished, 5=Processing. Difference between 1 and 5 is that 1 is we are doing a database back up. Most of the time it switches very quickly from 1 to 5.

4. **rsync the .ddc files** from the on-site machines to the QUB servers. This is in addition to the background rsync process that runs in another script. This is to make sure we have the most up to date data. Note that only the .ddc files are rsync-ed here not the full image frames. Those are checked with rsync when stamps are created.
5. **\$MULTIPROCESSOR** kicks in on the .ddc files. Calls `ingesterWrapperMultiprocess.py` which is python wrapper to a C routine which:
 - Reads the contents of the ddc files
 - Checks the RA-Dec for a pre-existing source in the database (collects the previous ATLAS_ID).
 - For new sources, a new 19 digit ATLAS_ID is generated and the detections are aggregated under the source designation.
 - Insert the ATLAS_ID into the row with the rest of the .ddc information in the .ddc info table (`atlas_detectionsddc`).
 - Calculates the location in the hierarchical triangular mesh. Recorded at level 16 into the file (the size of the triangles at level 16 is roughly a few arcseconds by a few arcseconds).
 - **Diff logs** - legacy information - *TODO: Check with ken*
 - **Ingest cut**: if a difference imaging cell (see Section 2.1.2) has too many detections then none of the objects in that cell make the ingest cut. The data is still put in the DB and will appear in LC but it won't then go on to be ingested by sherlock and the rest of the stream processing.
6. **Post-ingest cuts**—(list of cuts later on). Just look at the detections from today (or N days if you set another value at the start of the file). *Everything that makes the cuts is now in the eyeball list (=4).*
7. **Recenter** the object by choosing the detection whose Ra and Dec is closest to mean Ra and Dec of all detections.
8. **Some stats** calculated with `atlasGenerateDailyStatsMultiprocess.py`.
9. *Is this were `updateVRAScores.py` lives?*
10. **Sherlock** is run on everything that doesn't have a context already.
11. **Move Var. Stars.** out of eyeball list into the variable star list
12. **Cross-match** other surveys (ztf, pan stars, tns)
13. **Pan-starrs finders** generated for anything in the good list with Dec above -30
14. **Make stamps** (obs., ref., diff – 200x200 pixels) for every recent detection up to 6 (no more than 6). Then finds a representative triplet from these that we can then see on the web server.
15. **Real/Bogus Scoring**: The name of the output .csv file is put in an environment variable. Then the classifier is run on all the diff stamps (up to 6) for each object. The final *score is the median* of all these and it is recorded alongside the 19 digit ATLAS_ID.
16. **RB cute <0.2**: everything with rb score <0.2 is put the garbage (list 0)
17. **Move known AGNs** to their list
18. **Initialise objects in VRA**: `addRBValuesToVRAScores.py` is called and adds to the VRA scores table the data saved in the RB scores output file. Uses the token of the vra and the server is called through the API so will appear under `apiusername` column as vra.
19. **Check ephemerides** for known movers and solar system objects. These are flagged but stay in eyeball list.
20. **Move objects to SMS/LMC, Fasttrack lists**
21. **Force photometry** on all good objects with VRA score (rank) > 4

When we say that we move an object to a list, all we are doing is changing with list (main) list number is associated with it. Those are mutually exclusive, unlike the cutsom list numbers.



Table 2.1: Main Lists on the ATLAS transient web server

Number	Name
0	Garbage
1	Follow-up
2	Good
3	Possible
4	Eyeball
5	Attic
6	Stars
7	Known AGN
8	Fast Track
9	Movers
10	Magellanic Clouds
11	High Proper Motion Stars
12	[NEW - February 2025] Galactic Candidates
13	[NEW - February 2025] Duplicates

Table 2.2: `tcs_vra_scores`

Column	Description
<code>id</code>	PRIMARY KEY
<code>transient_object_id</code>	ATLAS.ID
<code>preal</code>	"real" score. 1 = Real, 0 = Bogus.
<code>pgal</code>	"Galactic" score. 1 = Galactic, 0 = Extra-galactic.
<code>pfast</code>	"Fast" score. 1 = immediate action required, 0 = can wait.
<code>rank</code>	VRA rank
<code>rank_alt1</code>	Alternative rank, usually used by the old models
<code>galcand</code>	Is Galactic candidate (1, 0 or NULL)
<code>timestamp</code>	yyyy-mm-ddThh:mm:ssZ
<code>apiusername</code>	Username responsible for the entry if they used the API to interact with the table. If they did not use the API, None.
<code>username</code>	Username responsible for the entry if their action was recorded through the web interface (they clicked a button). If API was used, None.
<code>debug</code>	False if the entry is genuine recording. True if the entry is a row inserted during code testing or debugging. Default is False.

2.2 VRA DATA STRUCTURES

[Add a cool schema graph from Ken's fancy tool?]

The `tcs_vra_scores` table (see Table 2.2) is the main record keeper for the VRA

The To-Do list of the VRA is a table that keeps track of which objects the VRA should be looking to update the scores of if more data comes to light (see Table 2.3). New objects (ATLAS.ID) are inserted when they first go through the ingest process, and ATLAS.ID are removed when humans make a decision on the webserver.

The `tcs_vra_rank` table (Table 2.4) keeps track of the ranks given by the VRA to each ATLAS.ID. Unlike the `tcs_vra_scores` where a given ATLAS.ID will have several associated rows, here only one rank value is recorded for each ATLAS.ID. When new scores are calculated (e.g. when new data is available), the rank will be calculated again and replaces.

The point of this table is to be read by the Django code to display the VRA rank on the webserver, and provenance/history of the rank does not need to be recorded here as it is easily calculated for each row of the `tcs_vra_scores`.

The ranks displayed on the website are updated using a *trigger* whenever the `tcs_vra_rank` table is updated.



Table 2.3: tcs_vra_todo

Column	Description
<code>transient_object_id</code>	PRIMARY KEY
<code>timestamp</code>	The timestamp and inserted - mostly for book keeping

Table 2.4: tcs_vra_rank

Column	Description
<code>id</code>	PRIMARY KEY
<code>rank</code>	The calculated rank

2.3 VRA PACKAGES AND ENVIRONMENTS

There are two main packages used by the VRA: `atlaspaclient` and `atlasvras`. On the server we also have a `vra` conda environment, but `atlaspaclient` and `atlasvras` are usually exported in the bash files that go on to running our python codes.

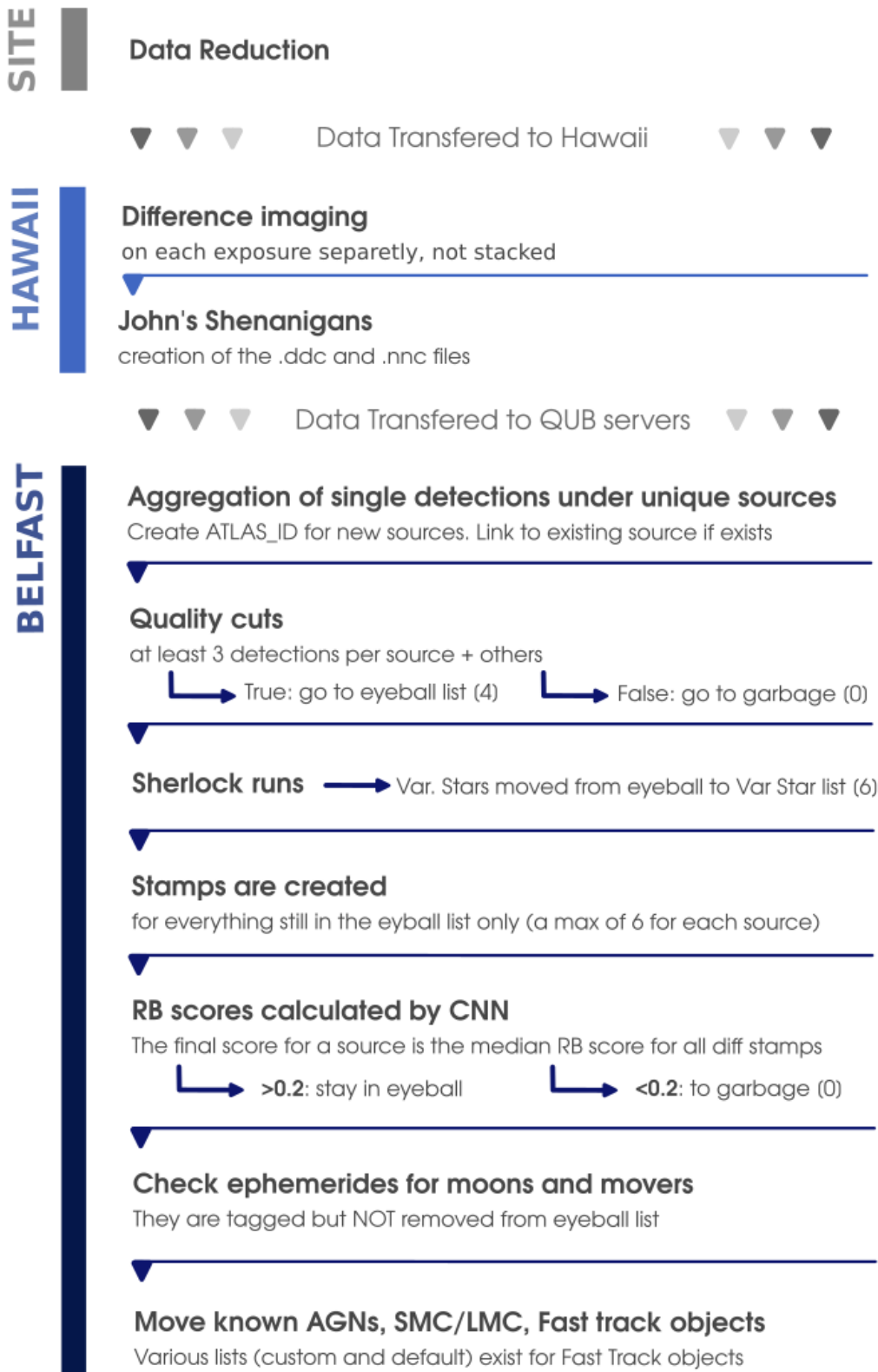
2.4 RECORDING PASSIVE FEEDBACK

An important part of the VRA design is to include `passive feedback` mechanisms. These are simply ways to systematically record user decisions without asking the users to take any new action or change the way they interact with the data. Currently, passive feedback is recorded when users move an object to a new list, according to the logic in table 2.5.

Table 2.5: **Passive feedback logic table**. This is used to populate the `tcs_vra_scores` table when users make a decision on an object. Until Tuesday 2024-03-05 HPM `preal` was set to 1.0 - this was fixed in version 0.3.4 of `psats-server-web`.

Destination	<code>preal</code>	<code>pgal</code>	<code>pfast</code>	<code>rank</code>
Attic/HPM*	1.0	1.0	NULL	NULL
HPM	0.0	1.0	NULL	NULL
Possible	0.5	NULL	NULL	NULL
Good	1.0	0.0	NULL	NULL
Garbage	0.0	NULL	NULL	NULL
Auto-Garbage	NULL	NULL	NULL	-1
Eyeball	NULL	NULL	NULL	NULL
Follow-up	1.0	0.0	0.5	NULL

Figure 2.1: Summary of the data processing for the transient server data (pre-VRA)



Part II

First data and code release: Crabby and Duck

3

Duck - Dealing with auto-garbage

Date cutoffs: 2024-03-27 to 2025-01-22

3.1 DUCK 1.1 - FIXED

Policies:

- Extra-gal eyeball list everything with VRA rank ≥ 7
- Galactic candidates if within 0.9 of coordinate (1,1) (fudge factor = 0.9).
- On day1 Garbage anything with VRA rank < 1
- On the second visit garbage anything with maximum VRA rank < 2
- On subsequent visits garbage anything with mean VRA rank < 3 .

The Data

The raw data is the same as for Duck 1.0 (see Section 3.2.2). The auto-garbage labels were not half-guessed, and the training and validation set were randomly sampled again.

Lessons Learned: Garbage is NOT "Not Galactic"

As we can see in Figure 3.1, the Garbage and Auto garbage categories *are* strongly correlated with the galactic plane. This is not exactly new information, but its only when looking at how to separate the "galactic" from the "not galactic" garbage in the auto-garbage category that I realised how flawed this was.

Galactic classifier NO LONGER USES THE GARBAGE (or autogarbage) categories. Only the PM, Good and Galactic are used.

Lessons learned: Overly represented bad tiles

Another thing we can see in Figure 3.1, particularly in the Garbage (red) map, we can see some clear tiles (Squares) - those are associated with a wave of bad alerts, likely associated with one specific date and event (e.g. trailing). I removed **two of those tiles**:

- The one between RA 315 and 325 degrees, with MJD 60,624 (around 900 alerts)
- The one between RA 348 ad 356 degrees, with MJD 60,605(around 400 alerts)

I re-ran all the policies and didn't find it made that much of a difference. **Still, in future need to be aware that these types of samples can pop up and remove them or resample them.**

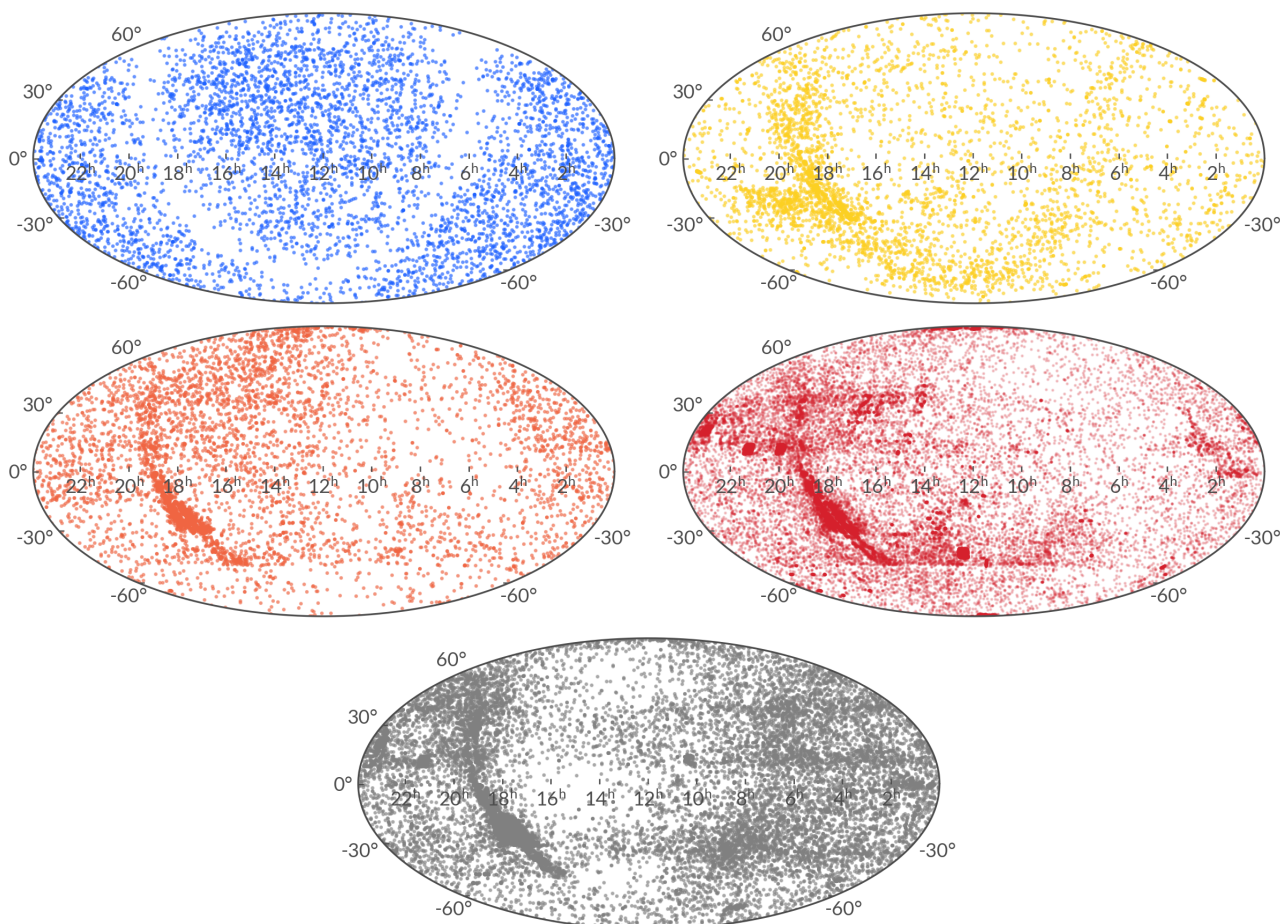


Figure 3.1: Ra and Dec distribution of our labels.

3.1.1 In Production Performance

see paper

Crabby Vs Duck

I didn't do the Crabby comparison to duck 1.0 partially because I didn't have the time and then because I realised the model was sub-par because of the way I had treated the auto-garbage. Also I later realised there were a few bugs with Day N models not running and auto-garbage being too zealous (see version history yellow boxes). So here we start fresh and just look at decisions made after 2nd April. In Figure 3.2 we can see our Crabby Vs Duck scores. The dashed line shows the line of equal scores.

The first thing to note is how the auto-garbage distribution has a really steep rise above the dashed line. Duck is much more sure about the auto-garbage than Crabby was. Note the distribution of the points not extending much past Duck ranks = 3 is a direct result of our auto-garbage policies.

The other interesting feature is that the good objects are showing a distribution that's mostly below the dashed line. So Duck generally ranks good alerts with higher scores.

For the galactic and PM stars the scatter is broader, which is not unsurprising we've seen before that in the middle of this plot the scatter tends to be greater. It's good to see that the galactic alerts and PM alerts are not mixing much - the real galactic alerts have higher scores than the bogus ones. It also looks like galactic alerts are ranked lower in Duck than Crabby. This is likely a direct result of the sharper p_{gal} score distribution (see paper).

- 1225647201451438100: 1 point - discovered by GOTO
- 1223006940264624600: 1 point - discovered by GOTO
- 1222920920352812400: 1 point - discovered by GOTO

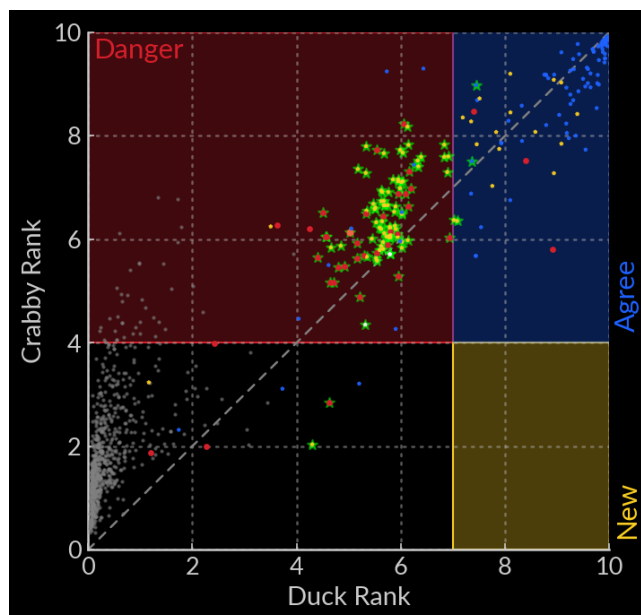


Figure 3.2: Duck Scores VS Crabby scores. The green stars show the alerts that were given the galactic flag by duck (and therefore eyeballed too even if didn't make the threshold cut-off).

- [1204246570015825100](#): noisy only 2 points - ZTF
- [1164932601390234500](#): real faint and noisy
- [1160027150080731800](#): faint goto discovery
- [1130831401843752200](#): faint af
- [1130828681843749400](#): duplicate
- [1123853041070659000](#): duplicate
- [1123852721070658800](#): shit and faint - found by ZTF

3.2 DUCK 1.0 - HOW NOT TO DEAL WITH THE AUTO-GARBAGE

3.2.1 Summary

In production: 2025-02-03 to 2025-03-06

Policies:

- Extra-gal eyeball list everything with VRA rank ≥ 7.5
- Galactic candidates if within 0.45 of coordinate (1,1) (fudge factor = 1).
- On day1 Garbage anything with VRA rank < 1
- On the second visit garbage anything with maximum VRA rank < 2
- On subsequent visits garbage anything with mean VRA rank < 3 .

3.2.2 The Data

Additional data gathered between 18th August 2024 and 22nd January 2025 is added to our existing training and validation sets from Crabby (see Chapter 4). The start date of this data set corresponds to the beginning of eyeballing operations involving the VRA. **Note that there is a gap of 6 days** between the end of the previous data set and this one because there was a one-week development period during which the first working prototype of the VRA was but in production during which the I took over eyeballing operations and focused their efforts on getting the VRA prototype operational. Since a one week hiatus is shorter than some weather patterns we do not anticipate this gap would have negative effect on the overall training of the current VRA. The end-date of this time window was defined by the fact that I had a bit of time between the Oxford Lasair meeting in Jan 2025 and my trip to Belfast at the end of Jan 2025.

This additional data is the first set affected by the interaction between the VRA and the humans. As we can see in Figure 3.3 and Table 3.1, a significant fraction (over 23k) of the alerts were auto-garbage, so not seen by humans. To deal with this "rapidly" I tried to leverage the existing VRA, and guess the label of some of the auto-garbage (see next section). The other change brought by Duck and the VRA 1.0 release is the pruning of some of the features, particularly the sherlock flags (see Table 3.2). These features were known since Crabby to not be useful but now this pruning was finally implemented.

Figure 3.3: Label of the additional data from August 2024 to January 2025

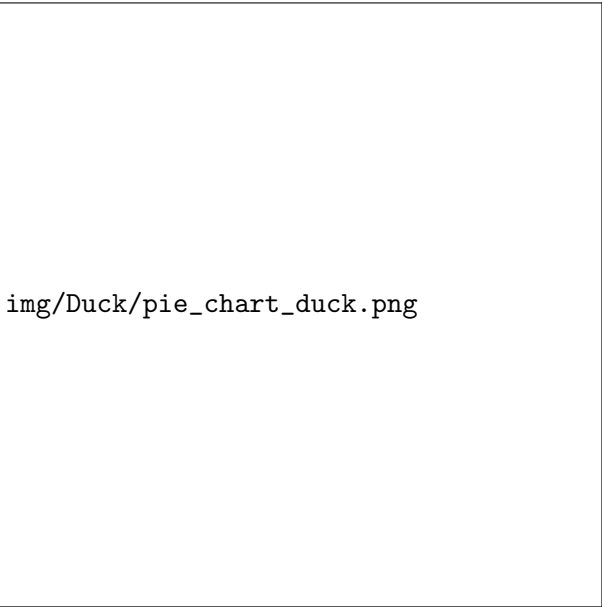


Table 3.1: **Number of samples** for each classes in the Training and Validation sets for the day 1 models.

Label	Training	Validation	Total
Auto-Garbage	[re-labeled]	3,590	23,752
Garbage	2,200	880	5,969
Proper Motion	1,133	66	440
Galactic	1,193	199	1,707
Good	2,000	409	2,493
TOTAL	6,526	5,144	34,361

Table 3.2: **Removed Features** in Duck 1.0

Name	type	Description
DET_mag_median	float	median magnitude of the detections at day N
SN	bool	(sherlock classification) if SUPERNOVA
NT	bool	(sherlock classification) if NUCLEAR TRANSIENT
ORPHAN	bool	(sherlock classification) if ORPHAN
CV	bool	(sherlock classification) if CATAclysmic variable
UNCLEAR	bool	(sherlock classification) if UNCLEAR

Re-eyeballing

I selected data to be re-eyeballed in a similar way to done with Crabby previously. Re-eyeball the **galactic labelled** alerts that are missed by the Galactic flag; re-eyeball the **labelled Good** objects that are missed by an eyeball threshold of 7.

These scores are calculated using a **preliminary model that uses the Crabby training data and the new data we have here that was labeled either by humans or by our "guesses" mentioned above**. These preliminary models can be found in the jupyter notebook `DuckReeyeball_and_train_val_split_day1.ipynb`.

The data flagged for re-eyeballing was flagged before I realised that there were some duplicate IDs in the `X_duck` dataframe and that changed the results of the preliminary models somewhat, with 90 more galactic events and 30 more good events flagged for re-eyeballing on re-running the code. Since I'd already re-eyeballed 800 objects and was further along the steps of retraining the VRA, I frankly just ignored that difference as I didn't think it'd change the results significantly.

In the end the preliminary models are **flawed by the smae mishandling of the auto-garbage describe below**.

If I had time I really should do another round of re-eyeballing based on Duck v1.1 models

3.2.3 The auto-garbage mishandled

Due to lack of human resources and time constrains, re-eyeballing the 23,752 items is not achievable. Instead I attempt to use the Crabby production models of the VRA to help retrieve good examples of the "Garbage" and "Proper Motion" category. I predicted the real and galactic scores of all the data in this new set and assigned the following labels:

- "Garbage" if real score < 0.2 and galactic score < 0.1
- "Proper Motion" if real score < 0.2 and galactic score > 0.9

This is based on the distribution of the data I can see in Figure 4.3.

Why this feels like a good idea:

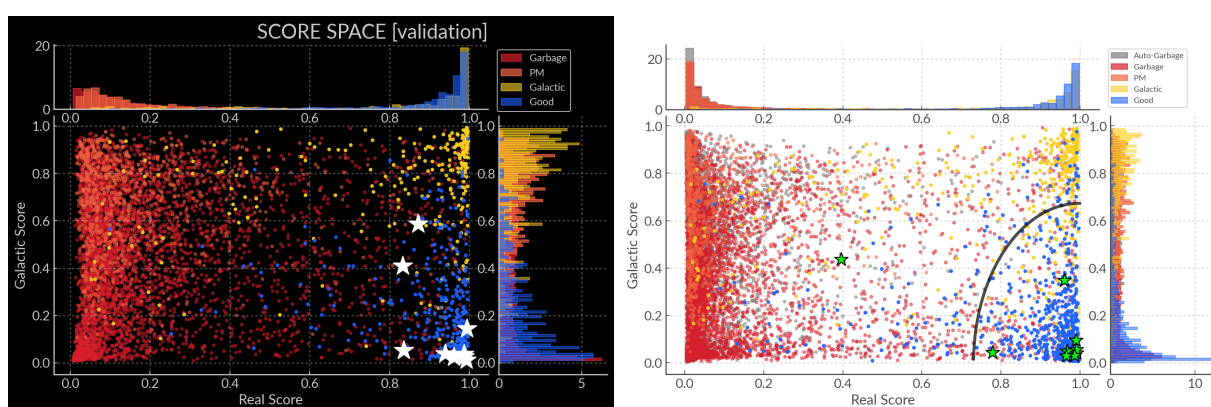
- The previous version of the VRA works quite nicely, and those corners of the plot are where the VRA is most certain so it's very likely to be right
- I can't re-eyeball 23k objects. I *could* re-eyeball randomly 2k objects but since there are about 7 times fewer proper motion stars as garbage, if I want at least another 1k PM stars I need to eyeball on average 8k objects... overall not great numbers.

Why this is actually a terrible idea:

- The corners of the plots is where the VRA is most certain. According to information theory, these are the samples that teach us *the least*. **More data does not necessarily equal more information**, or at least the increase is not linear.
- The worst case scenario is that the samples I select in the corner of the plot all share some specific characteristics or features that make the VRA particularly certain. As a result of this selection process I then **over represent these characteristics** in my training data and **lead the classifiers generalise poorly**.

In the end this latter point is what I think happened. The results are not awful but AT2024aju in the `Key_transients.ipynb` notebook has a really low real score compared to Crabby (see Figure 3.4).

Figure 3.4: Key transients Crabby Vs Duck 1.0. All key transients have real score > 0.8 in Crabby (left panel) but AT2024aju has real score < 0.5 in this version of the VRA (right panel).



4

Crabby - Finer Sherlock features

Date cutoffs: 2024-03-27 to 2024-08-13

4.1 SUMMARY

In production: 2024-12-06 to 2025-02-03

This version of the VRA still used only one eyeball list, and it changed the scale factor (or fudge factor) on the galactic axis using in scoring from 0.5 (in BMO) to 0.4. It is also the first set of models where we tried to **use sherlock features directly** such as the separation from the cross-matched source, in addition to the Sherlock flags we had been using.

Policies:

- Eyeball everything with VRA rank ≥ 4
- On day1 Garbage anything with VRA rank < 1.5
- On the second visit garbage anything with maximum VRA rank < 3
- On subsequent visits garbage anything with mean VRA rank < 3 .

Note: The threshold is much lower than in future iterations because I messed up when plotting the VRA ranks on the plots I was using to eyeball the best threshold (see below).

4.2 THE DATA

Although the time window includes mostly the same alerts as Arin and BMO, **the data are not exactly the same. The lightcurves were re-downloaded** as I realised that since I was downloading the data when decisions were being made on the web server some objects only had a few days of light curves on my machine and a lot more on the server!

A key difference is in the training/validation split - it is now RANDOM and no chronological. I found that the issue with chronological split is that, on such a short baseline, weather events can lead to the training and validation data set to have different enough properties that we are removing valuable training sets from the training data AND comparing apples and potatoes when doing our tests, yielding poorer metrics (I think my val set had a major trailing event in it those time).

Crabby is the first version of the VRA data and models which is released publicly, which can be found here on the [ATLAS VRA Zenodo community](#). There is a lot of information in the jupyter notebooks so these sections are less detailed than those of the early prototypes.

You can see the properties of our complete pre-VRA data set in Figure 4.1 and Tables 4.1, 4.2.

Figure 4.1: Label split in the Crabby data set.

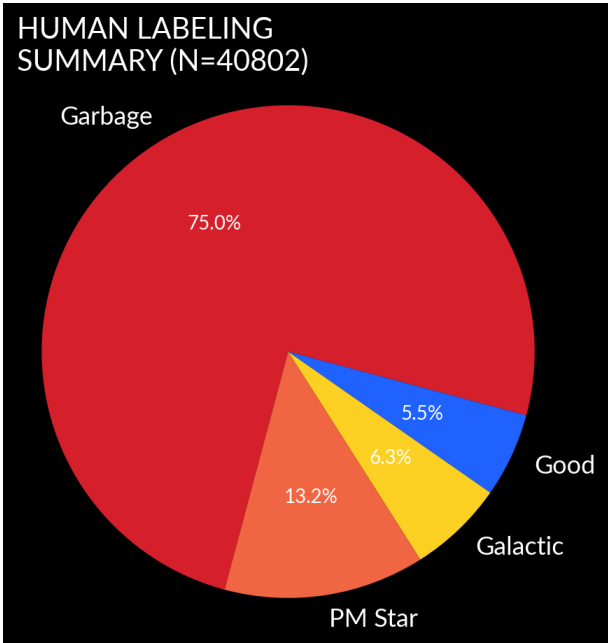


Table 4.1: Day-1 Models. Number of samples for each classes in the Training and Validation sets

Label	Training	Validation	Total
Garbage	2,200	4,619	30,607
Proper Motion	2,200	797	5,372
Galactic	2,210	357	2,567
Good	1,908	348	2,256
TOTAL	8,518	6,121	40,802

Table 4.2: Day-N Models. Number of samples for each classes in the Training and Validation sets.

Label	Training	Validation
Garbage	14,082	30,224
Proper Motion	14,752	5,365
Galactic	14,131	2,366
Good	12,523	2,233
TOTAL	55,488	40,188

We added more contextual information such as the redshift, extinction and separation from the cross-matched source, as well as the Ra and Dec scatter. Also all the features passed on day 1 are also passed on day N with the addition of the day N specific features. You can see the full list of features in Tables 4.3 and 4.4.

Table 4.3: [Features](#) used in the Day-1 Scoring models.

Name	type	Description
Nnondet_std	float	Standard dev. of the number of non detections between detections
Nnondet_mean	float	Mean of the number of non detections between detections
magdet_std	float	Standard deviation of the magnitude of the historical detections
DET_Nsince_min5d	float	[NEW] Number of detections
NON_Nsince_min5d	float	[NEW] Number of non detections
log10_std_ra_min5d	float	[NEW] Log10 of the standard deviation of the RA
log10_std_dec_min5d	float	[NEW] Log10 of the standard deviation of the Dec
ra	float	Right Ascension
dec	float	Declination
rb_pix	float	Real bogus Score from Josh's CNN
z	float	[NEW] Spectroscopic redshift
photoz	float	[NEW] Photometric redshift
ebv_sfd	float	[NEW] E(B-V)
log10_sep_arcsec	float	[NEW] Log10 of the separation in arcsec from a nearby source
SN	bool	(sherlock classification) if SUPERNOVA
NT	bool	(sherlock classification) if NUCLEAR TRANSIENT
ORPHAN	bool	(sherlock classification) if ORPHAN
CV	bool	(sherlock classification) if CATAclysmic VARIABLE
UNCLEAR	bool	(sherlock classification) if UNCLEAR

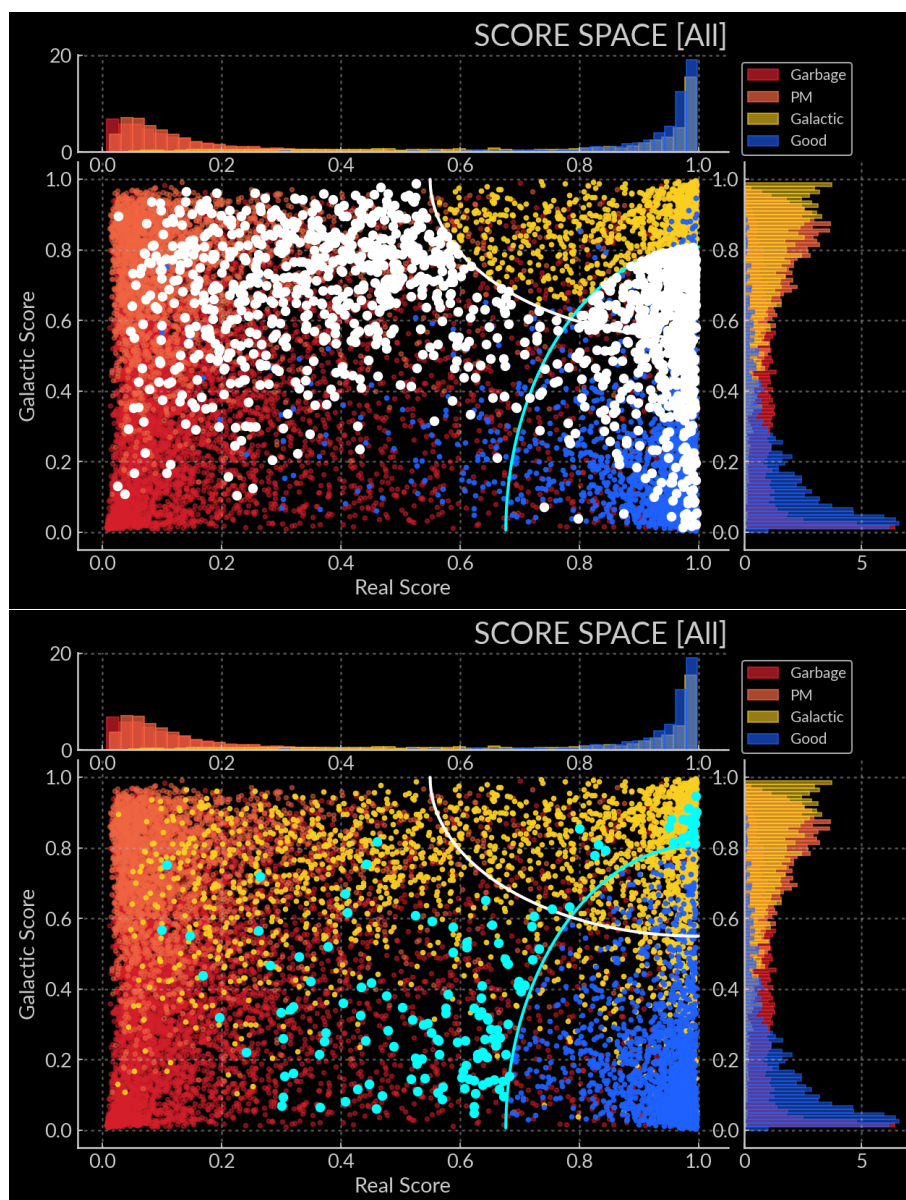
4.2.1 Re-eyeballing

After the initial modeling and ranking was put in production, I decided to re-eyeball some of the data (see spreadsheets in the release). We can see in Figure 4.2 how I selected the objects. In total 148 Good objects and 1370 "Galactic" (i.e. Attic) objects were re-eyeballed a posteriori because they were outliers which did not meet our eyeballing criteria.

Table 4.4: [Features](#) used in the Update Scoring models. Max mag and Mag mag day were inspired by the BTSBot paper. **We only use data from dayN -5 to + 15.**

Name	type	Description
dayN	int	phase w.r.t alert. Phase 0 is (usually, unless there are delays) the first bundle of data to hit the eyeball list. It has <code>phase_init</code> values between -1 and 0 (data to first hit the eyeball list was observed just <i>before</i>).
DET_mag_median	float	median magnitude of the detections at day N
DET_N_today	float/int	Number of detections at day N
DET_N_total	float/int	Number of detections since day -5
NON_mag_median	float	median magnitude of the NON detections at day N
NON_N_today	float/int	Number of NON detections at day N
NON_N_total	float/int	Number of NON detections since day -5
max_mag	float	Maximum magnitude value up to this day N
max_mag_day	float/int	day N when maximum magnitude was reached.
Nnondet_std	float	[NEW for day N] Standard dev. of the number of non detections between detections
Nnondet_mean	float	[NEW for day N] Mean of the number of non detections between detections
magdet_std	float	[NEW for day N] Standard deviation of the magnitude of the historical detections
DET_Nsince_min5d	float	[NEW] Number of detections
NON_Nsince_min5d	float	[NEW] Number of non detections
log10_std_ra_min5d	float	[NEW] Log10 of the standard deviation of the RA
log10_std_dec_min5d	float	[NEW] Log10 of the standard deviation of the Dec
ra	float	Right Ascension
dec	float	Declination
rb_pix	float	Real bogus Score from Josh's CNN
z	float	[NEW] Spectroscopic redshift
photoz	float	[NEW] Photometric redshift
ebv_sfd	float	[NEW] E(B-V)
log10_sep_arcsec	float	[NEW] Log10 of the separation in arcsec from a nearby source
SN	bool	(sherlock classification) if SUPERNOVA
NT	bool	(sherlock classification) if NUCLEAR TRANSIENT
ORPHAN	bool	(sherlock classification) if ORPHAN
CV	bool	(sherlock classification) if CATAclysmic VARIABLE
UNCLEAR	bool	(sherlock classification) if UNCLEAR

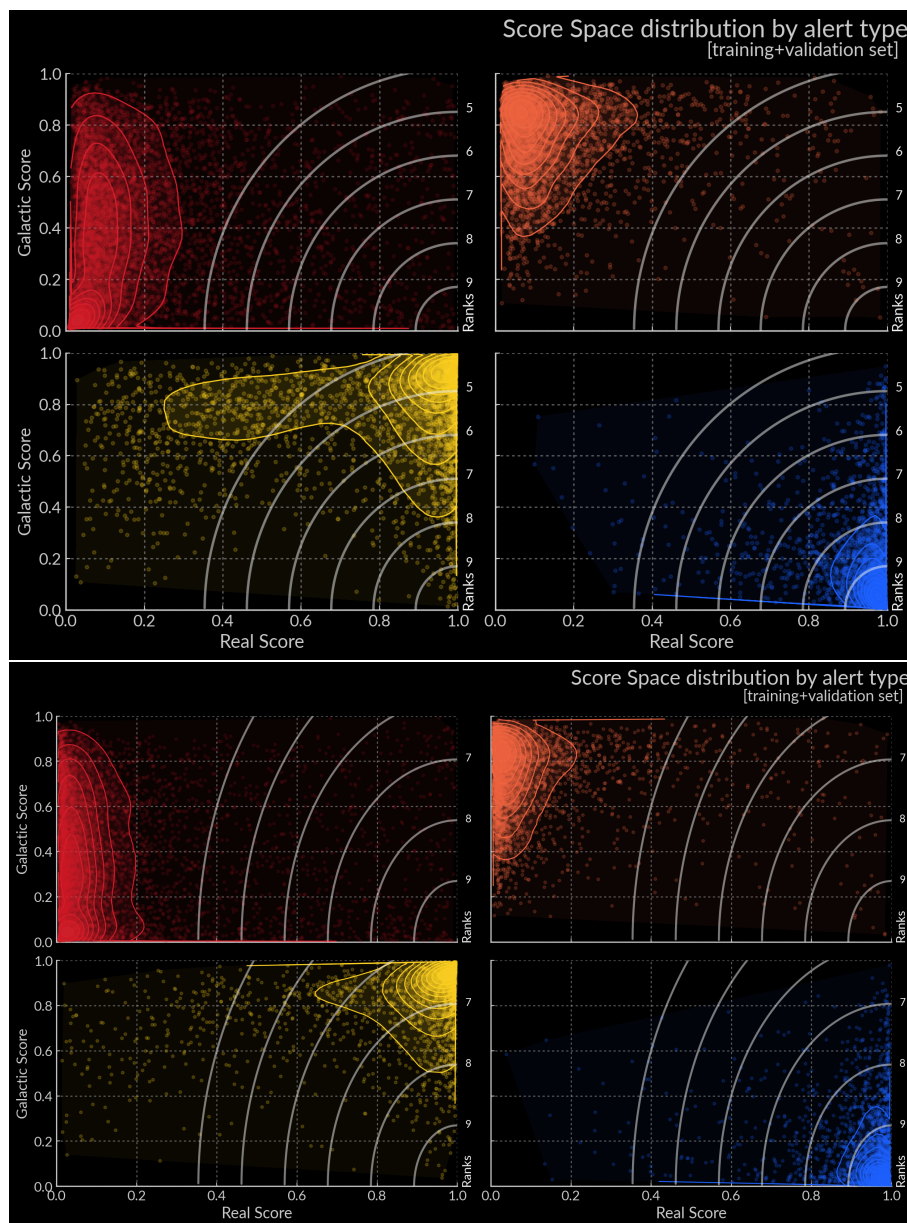
Figure 4.2: Visualisation of the re-eyeballing in Crabby



4.3 MODELS AND RANKINGS

In Figure 4.3 we can see the alerts split by types plotted in score space. The top plot shows the incorrect visualisation with which I selected the eyeball threshold of 4. By comparing the top and bottom plot we can also see the effect of re-eyeballing, mostly the **effect of mislabelings of the Galactic objects** particularly the Garbage and PM stars. About a third of the Galactic sample I re-eyeballed was garbage.

Figure 4.3: Visualisation of the ranks on the alerts distribution by alerts. TOP: **WRONG! I had forgotten brackets in my plotting equation of the ranks**, also the alerts plotted are using the labels before re-eyeballing. BOTTOM: Fixed the ranks display and re-eyeballing had taken place, primarily changing the Galactic (yellow) distribution.



4.3.1 Hyperparamter tuning

Some more systematic hyperparmater tuning was tried with Crabby. We did a grid search of

- learning rates: (0.1, 0.2, 0.3, 0.4, 0.5)
- l2: (1, 10)

Refer to the data release and notebooks under the `train_scoring_models/hyperparameter_tuning` directory. In the end we find $l2 = 10$ works consistently and the only really decent learning rates are 0.1 and 0.2 with marginal differences. We don't find that doing a grid search yields a different conclusion than eyeballing the best learning rate on the validation set predictions. In the future we'll stick to the best values we found here.

4.3.2 The start of the new eyeballing strategy

Whilst the Crabby models were in production comments from SJS led to the review of the eyeballing strategy. Using the Crabby data set we started testing the new strategy , which would consist in separating the eyeball list

by having the most likely galactic candidates put into a galactic candidate list.

The new strategy tests can be found in `_Gal_candidate_policy.ipynb` and `Policy_evaluation_new_strat.ipynb` notebooks.

Part III

Early prototypes: Arin and BMO

5

BMO - The first production VRA

Date cutoffs: 2024-03-27 to 2024-07-08

In production: 2024-07-13 to 2024-12-06

5.1 RAW DATA, CLEANING AND MAKING TRAINING DATA

The raw data was gathered the same as for the Arin models and everything was done on Heloise's machine.

5.1.1 Jupyter notebooks locations

All these notebooks are located in `home/stevance/Science/VRAwork/data/bmo`

`Clean_data.ipynb`: Creates the following output files. It runs in about 20 minutes.

- `detections_data_set.csv`
- `non_detections.csv`
- `non_detections_100days.csv`
- `vra_last_entry_withmjd.csv`
- `contextual_info_data_set.csv`

`Training_data_make.ipynb`: Takes in the csv files above (except `non_detections.csv`), creates the features to be put in the training data, balances the sets and and outputs the following files:

- `X_train_unbalanced`
- `X_test_unbalanced`
- `y_train_unbalanced`
- `y_test_unbalanced`
- `X_train`
- `y_train`

`Summary_plots.ipynb`: Makes plots summarising the data set such as the label composition (Figure 5.1), and plots similar to Figure 6.2 which are not shown here because they look essentially the same.

`update_scorer/UpdateScores_TrainingSets.ipynb`: Takes in the same csv files as the Day-1 Scorer and creates the features to be put in the training data, balances the sets and and outputs the following files. Yes they have the same name as the ones for the day-1 scorer, they're differentiated by their hoem directory only.

- X_train_unbalanced
- X_test_unbalanced
- y_train_unbalanced
- y_test_unbalanced
- X_train
- y_train

update_scorer/UpdateScores_Training_data_Tests1.ipynb: First attempts at making features and an absolute train wreck. Keeping it for posterity and just in case.

5.1.2 Data Overview

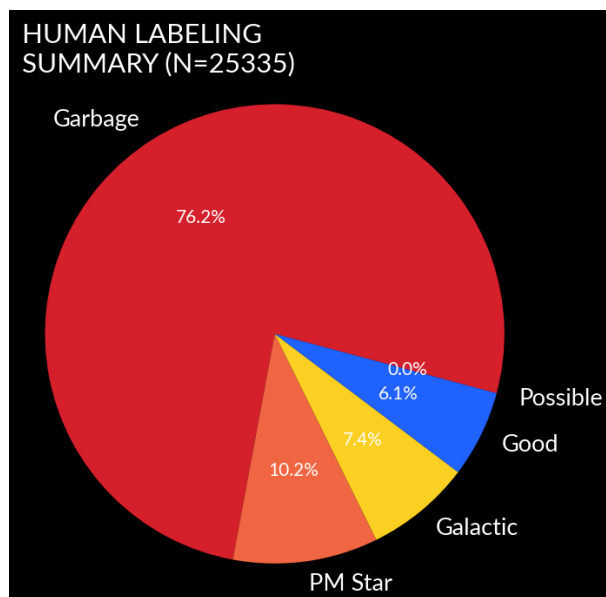
Size and split of the data set

The BMO data set is comprised of 25,335 alerts. Across the whole data set and up to the date cut-off we record 255,113 detections and 4,229,427 nondetections within the last 100 days. This is mostly worth noting to consider how these data sets will scale over time as more data is labelled within the VRA datastructures. **it may be worth considering lowering the history window for our feature creation from 100 days to, say 50 days.** In a few months when more data has been gathered it would be worth checking how different light curve history windows affect classification.

2025 Hindsight: 100 days is completely manageable for data cleaning. The reason for the highlighted comment above is that I had very inefficient for-loop that concatenated new data into the main lightcurve data frame with `pd.concat`. A terrible idea. As the dataframe grew the operation slowed to a crawl.

As can be seen in Figure 5.1, the label split is very similar to what was since in the original Arin Data set and the fraction of good objects stands firm around the 6% mark.

Figure 5.1: Label split in the BMO data.



Note on the 'Possible' labels

I have just realised that with these data I have 0.0% objects labelled 'Possible'. That is because I take the label from the 'last row' in the VRA scores table which is the last row where the decision was made by a human.

If object are first labelled as Possible and then later classified as good or garbage, then the label recorded here is the **final label, not the label given on day-1**. The implications are as follows: the label that the models are going

to be using for the 'Day-1' decision making are labels that were given by humans who had more data. It sorts of sets the model up for failure and potentially could lead to over-fitting if there are too many such objects. Given that the proportion of 'possible' objects was not large to begin with and that this label will disappear eventually, I don't think this is a major problem and **not actions are taken to mitigate at this stage**.

5.1.3 Balancing the data sets

There is a disproportionate fraction of Garbage in the data set (see Table 5.5). To balance the data set we **randomly sample 2000 garbage alerts**.

Table 5.1: Number of alerts in each category for balanced and unbalanced training sets

Type	Unbalanced	Balanced
garbage	15,669	2,000
galactic	1,406	1,406
good	1,192	1,192
pm	2,000	2,000
TOTAL		6598

Note **I did not create a balanced validation set**. Since in production we will perform the scoring on unbalanced data sets I want to check how well we perform on unbalanced tests.

5.1.4 Split into train and test sets

Instead of a random split we use a **chronological split** where the training data is the first 80% of the data set and the test set is the last 20%, as recommended by Steve.

This is because we expect some data drift (although maybe not over the course of 3 months, but these methods will be repeated in the future), so having the chronological split will make the metrics measured on the test set potentially more applicable to what we will see in production.

5.1.5 Features

I have kept the features from the Arin data set and 'exploded' the Sherlock Classifications into their own columns with a boolean value. This is so that we can learn more about the information carried by each class when doing permutation importance analysis. (Note to Steve, I haven't tried the other option you suggested where the sherlock classification remains as a single column but where they are given numerical values from 1 to 7 - I don't like that it would give the classifications some order relation that doesn't really exist)

Table 5.2: **Features** used in the Day-1 Scoring models.

Name	type	Description
Nnondet_std	float	Standard dev. of the number of non detections between detections
Nnondet_mean	float	Mean of the number of non detections between detections
magdet_std	float	Standard deviation of the magnitude of the historical detections
ra	float	Right Ascension
dec	float	Declination
rb_pix	float	Real bogus Score from Josh's CNN
SN	bool	(sherlock classification) if SUPERNOVA
NT	bool	(sherlock classification) if NUCLEAR TRANSIENT
ORPHAN	bool	(sherlock classification) if ORPHAN
CV	bool	(sherlock classification) if CATAclysmic VARIABLE
UNCLEAR	bool	(sherlock classification) if UNCLEAR

5.2 DAY-1 SCORING MODELS

5.2.1 Model description

Like with Arin we use the sklearn implementation of the Histogram Based Gradient Boosted decision trees (`HistGradientBoostingClassifier`). The only parameters set for the 'default' model are the learning rate (0.1) and the random seed (42).

5.2.2 Out of the box performance

True model performance is only meaningful in the context of ranking which we will look at later. However with this new data set we want to explore **hyper-parameter tuning** and **which features are missing/unimportant**. We can visualise the performance of the scoring models with the distribution of the predicted labels in the Score Space (Figures 5.2) and the ROC (Figure 5.3).

Figure 5.2: Label distribution in the Score Space (Galactic Vs Real). The 2D histograms are the marginalised distributions of the Real (top) and Galactic (right) scores.

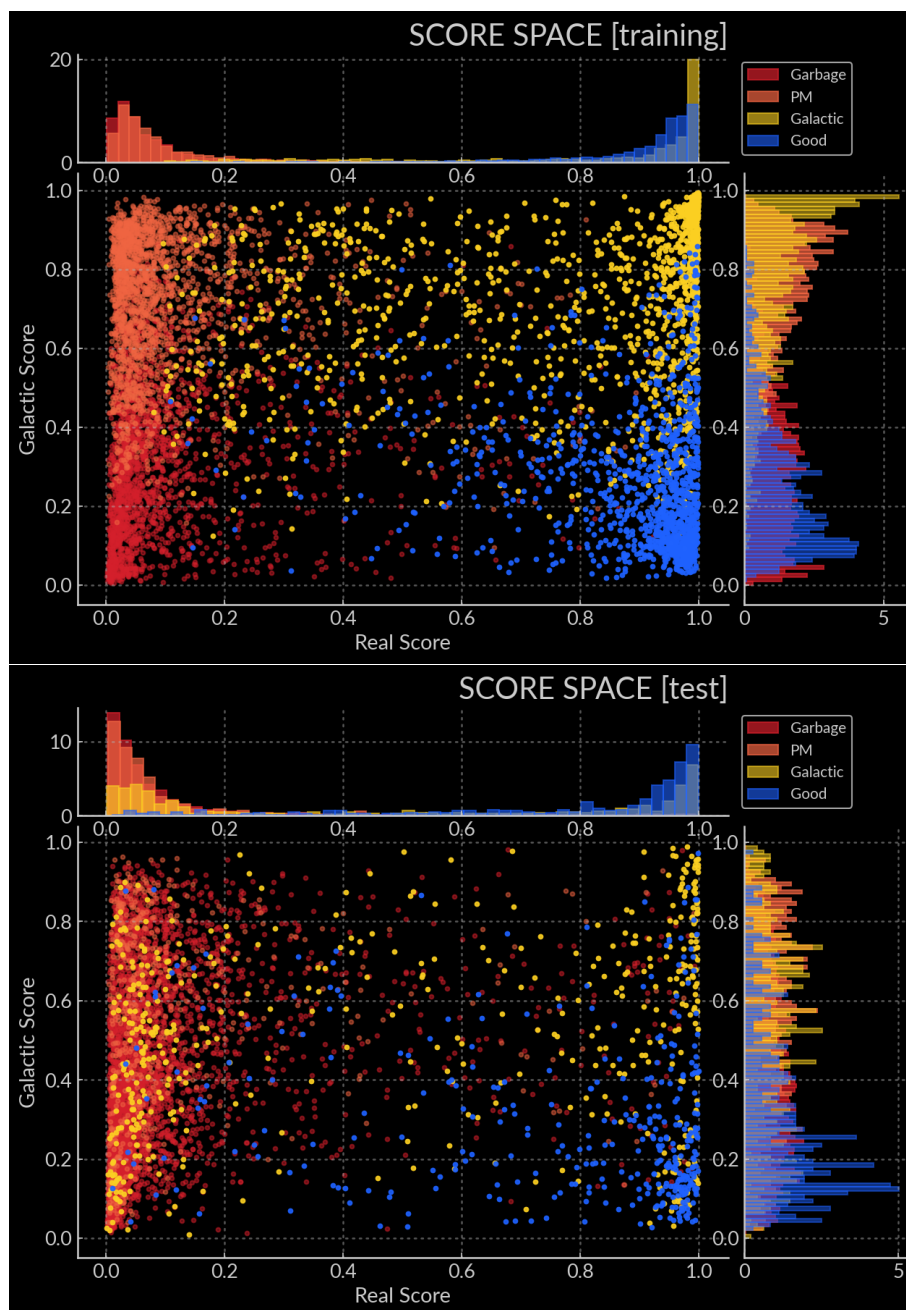
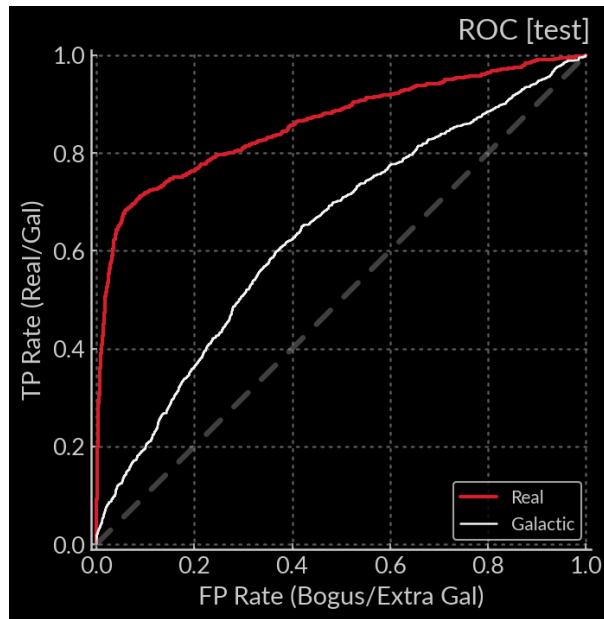


Figure 5.3



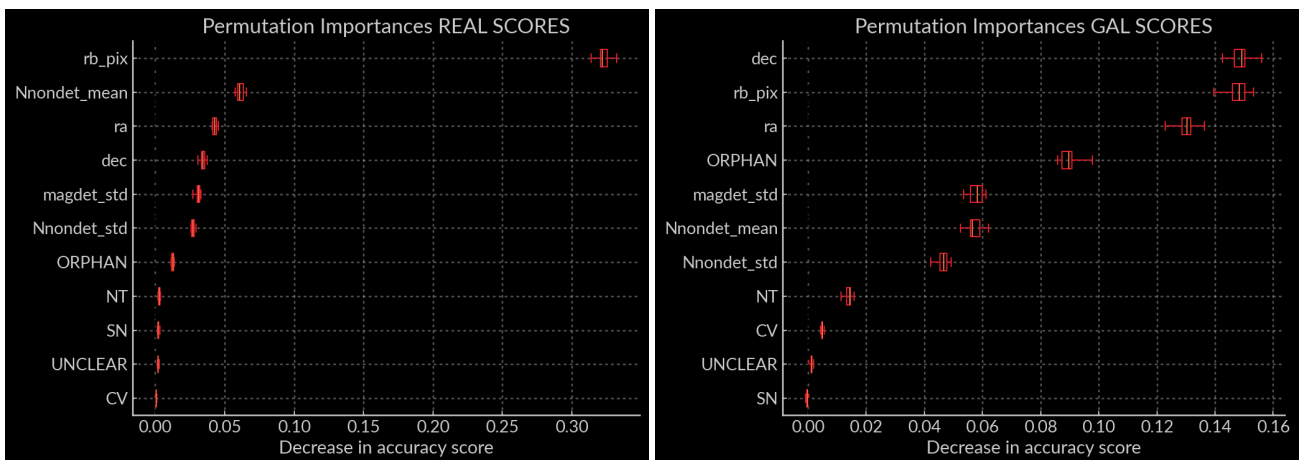
As we can see the Real model performs much better than the Galactic model when using default parameters. Before we start tuning the hyper-parameters of the models we take a quick look at the 'usefulness' of the features.

Permutation Importance

There are a number of interesting behaviours in the permutation importance plots in Figure 5.4. Firstly, the 'SN' label is not nearly as informative as I thought it would be, both for the real and the galactic model. I wonder if it could be due to the fact that 'SN' can be mislabeled when stars are reported as extended in e.g. the Pan-STARRS catalogue, leading Sherlock and the models to make mistakes [Note after mentioning this to Dave: the SN label in Sherlock is very generously attributed to avoid missing SNe, which is way it may be uninformative - **instead of Sherlock classes should I use features from the sherlock crossmatches to train on instead?**].

It is good to see that the features associated with the light curve history are doing *something*.

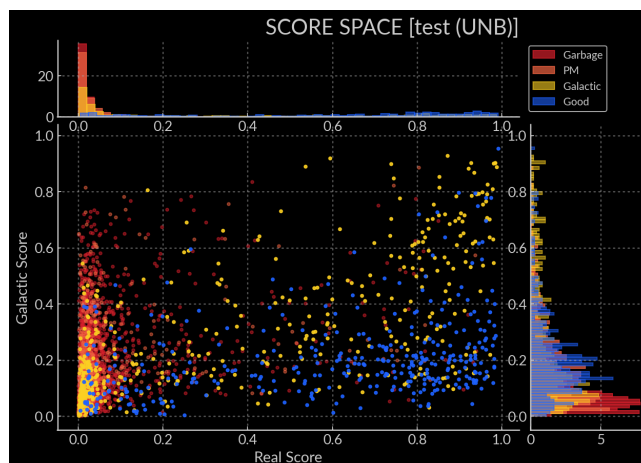
Figure 5.4: Permutation importance of the features in the default Real and Galactic models



Balanced Vs Unbalanced data sets

Just to check our balancing the training set helped our model we can try training on the unbalanced data set. As we can see in Figure 5.5 it basically biases the models to say everything is not real and not galactic (because for a high proportion of Garbage in the training set that is the most effective labeling strategy).

Figure 5.5: Score space when the models are trained on the Unbalanced dataset (with the same default settings which include no class weighting).



5.2.3 Hyper-parameter Tuning

Class weights

The sklearn `HistGradientBoostingClassifier` implementation includes a `class_weight` parameter which applies weights to the classes present in your data set. These weights can be manually set using a dictionary, one can ignore a whole class by setting its weight to 0, or we can selected the `balanced` option.

Here we explore the effect of the `balanced` option. In spirit it would be the same as balancing a data set before training (as we did above), except for the subtlety that we balance our data set according to **four distinct classes**, whereas the `balanced` option in our binary classifiers (Real and Galactic scorers) will balance according to the **two classes they care about**.

In order to explore the interaction of the `class_weights` setting and the use a balanced or unbalanced data set we try all four permutations for both the Real and Galactic scorers. In `Scoring_hyparm_opt_class_weight.ipynb` we plot the Score Spaces resulting from changing the parameters and data used. To avoid having to plot too many permutations we plot the scores resulting from the default Real (Gal) model when changing the training of the Gal (Real) model.

In the jupyter notebooks and saved plots the following codes are used to identify the permutations:

- W/D: Weights or Data
- u/b: Unbalanced or balanced

So for example the plot `score_space_galDbWu.png` is the score space where we used a balanced Data set and unbalanced class weights (default) to train the Galactic model, and the real scores plotted against are unchanged from first basics model in Figure 5.2.

To avoid drowning in plots we only include the Score Space for the best results which was the Balanced Data set with the class weights set to `balanced` – see Figure 5.6.

We can also look at the ROC curves for the Real and Galactic scorers in Figure 5.7. Particularly for the Galactic ROC it may seem like the unbalanced data set with the weighted class weights performs better. However if we inspect the score space for that model, we can see that the 'Good' Objects (the ones we care about), are now all over the place. **This highlights the importance of not relying too much on general performance plots and have visualisations that specifically highlight the question of interest** – in this case 'Can we classify the Good objects sufficiently well to help order the eyeball list' (although note that without the ranking algo and metrics this analysis is still incomplete).

For this reason, automated searches for best model parameters (that rely on the 'classic' metrics) may not be suitable!

Figure 5.6: Balanced data set and `class_weights` set to 'balanced'

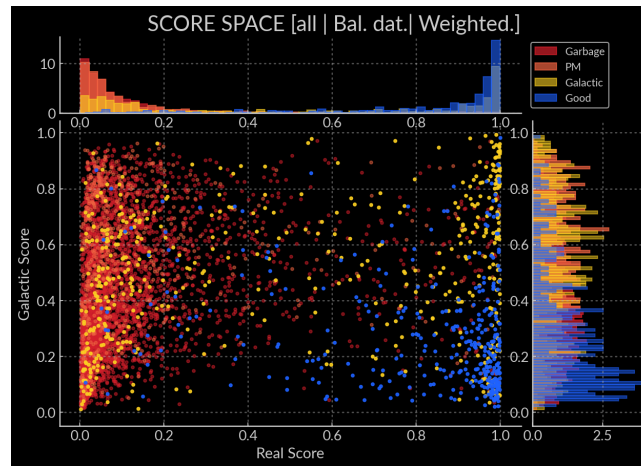
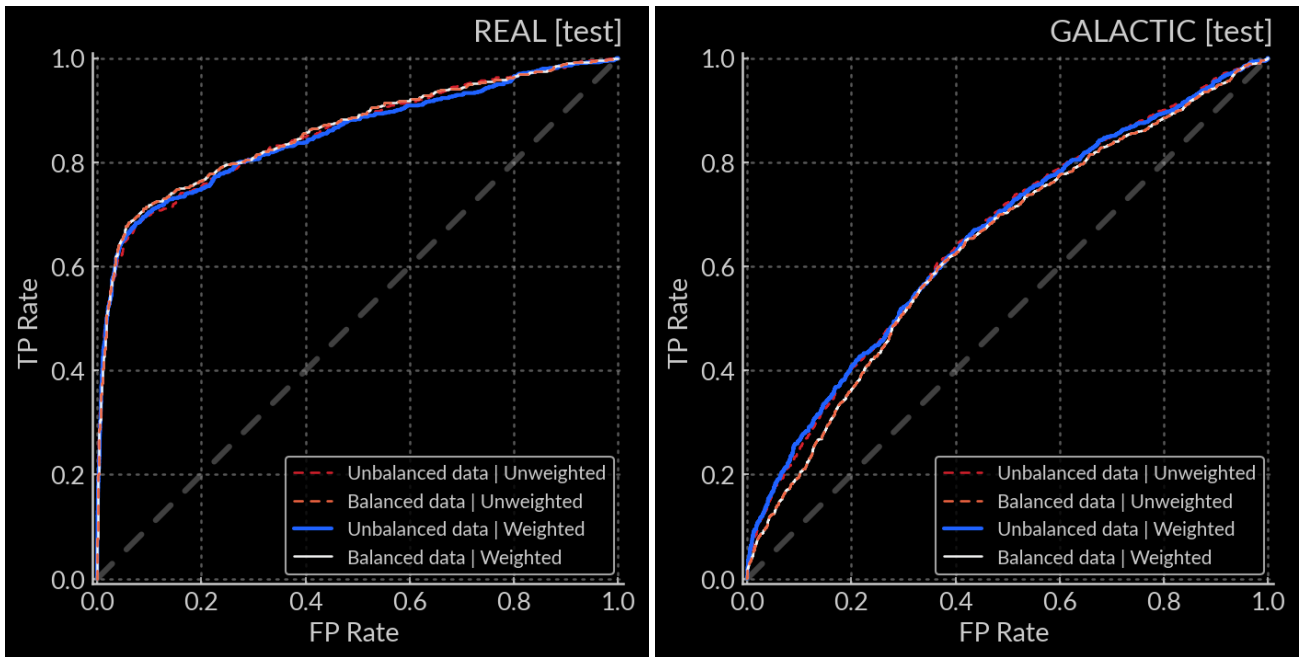


Figure 5.7: ROC for the Real and Galactic scorer models when using a mixture of balanced and unbalanced training data and class weight settings.



Regularization

There are two main regularisation parameters `L2_regularization` and `max_features` [and the learning rate but I put that in its own subsection]. The latter defines the '*proportion of randomly chosen features in each and every node split. Smaller values make the trees weaker learners and might prevent overfitting*'. The default is already set to 1 so it doesn't seem like much use to try and change this.

L2 regularization is worth trying and so we explore the use of the following values: 0.01, 0.1, 1.0, 10, 100. The resulting ROC curves for the Real and Galactic Scorers can be seen in Figure 5.9 and it looks like it's doing very little. The Score Spaces (see `Scoring_hyparm_opt_regularisation.ipynb`) do differ and the good objects distributions are affected somewhat. The high values causes the good labels to be more dispersed in score space and are probably not preferable. For the Real Scorer the best L2 (visually) from the Score Space looks to be $L2 = 1$, for the Galactic scorer $L2 = 10$.

When we put together all the 'best' parameters and look at ranking metrics this could be explored further to see if it changes.

Figure 5.8: Unbalanced data set and `class_weights` set to 'balanced' when training the Galactic scorer. The Real scores are the same as those in the first trained model in Figure 5.2.

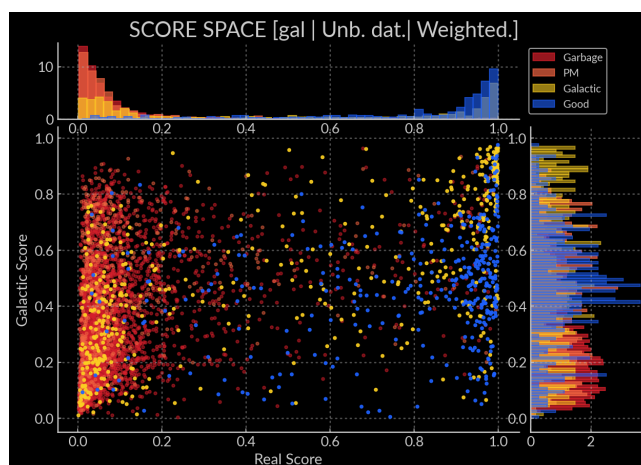
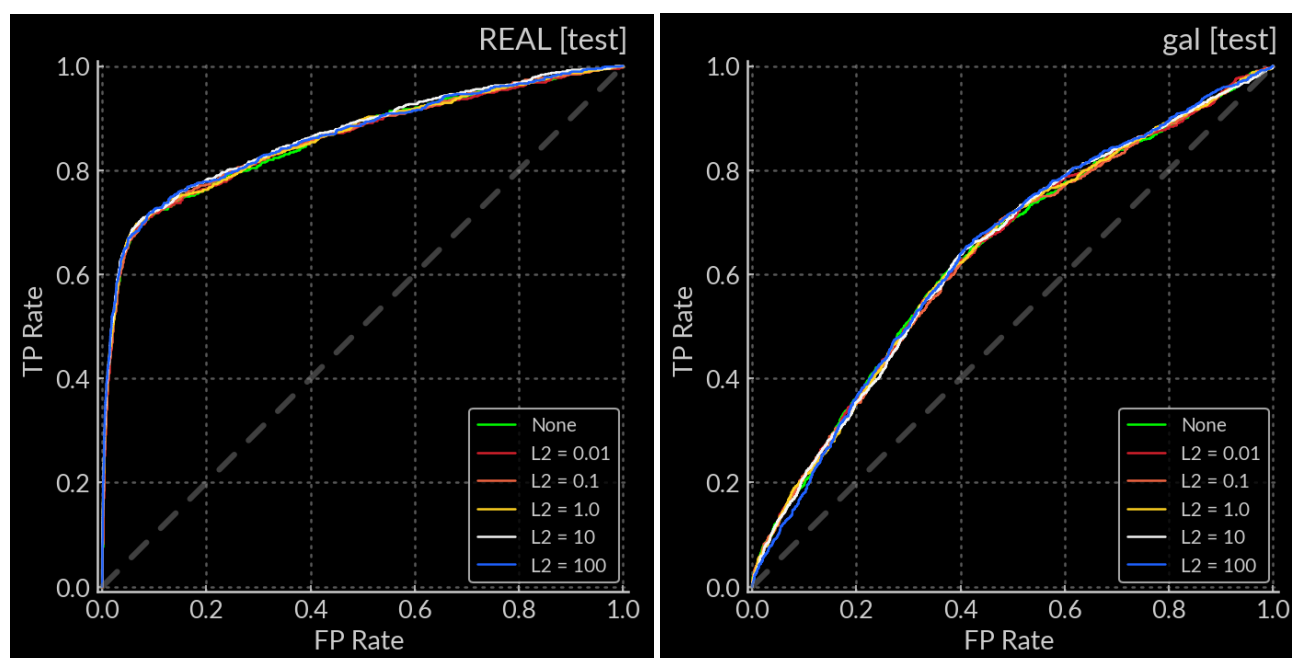


Figure 5.9: ROC for the Real and Galactic scorer models when using a range of L2 values



Tree Size

We explore the Min number of samples per leaf and the maximum depth parameters. The Minimum samples per leaf were explored for 5, 10, 15 (the default is 20) - it doesn't make much of a difference and the default (unconstrained) is working just as well as the best test.

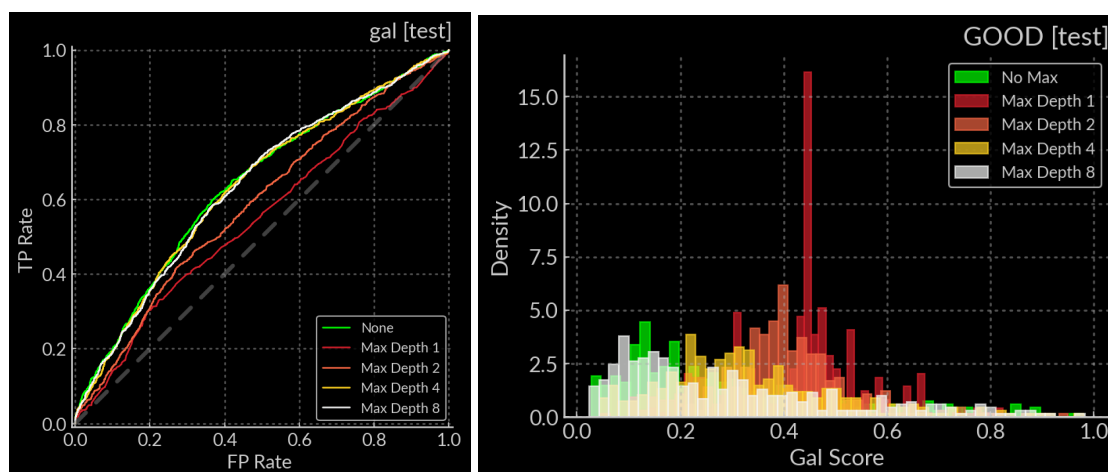
The maximum depth parameter was set to 1, 2, 4, and 8. It did make a difference **specifically for the galaxy scorer but not the real scorer**. You can see in Figure 5.10 how max depth affects the model. Reducing it really stunts the Galaxy scorer.

On the whole it's maybe best not to touch the tree size parameters.

Learning Rate

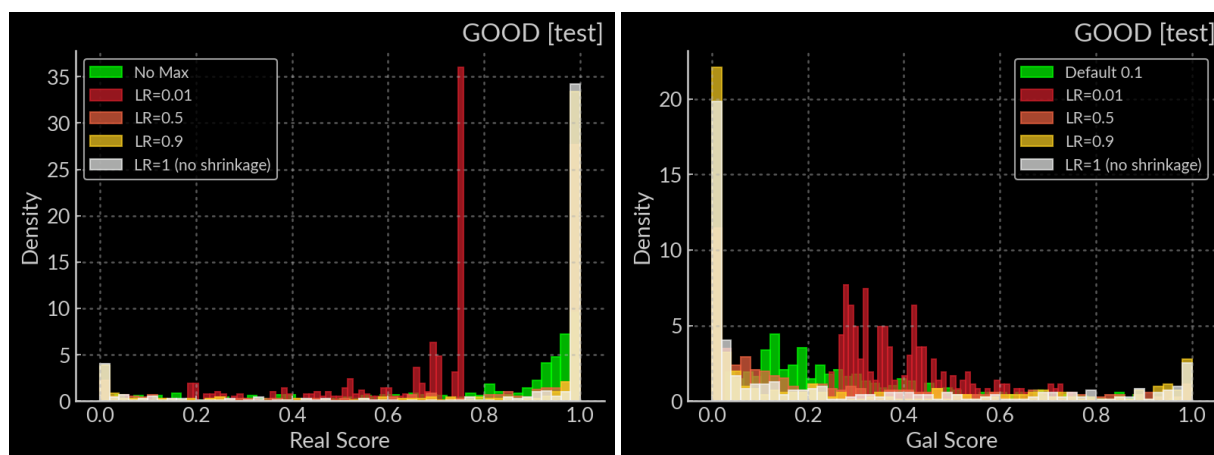
Finally I tried learning rates: 0.01, 0.5, 0.9 and 1 (meaning no shrinkage) because we already have the default 0.1. It is interesting to see the effect of shrinkage on the scores (even when you know what it does in theory it's cool to see it in action). The higher learning rates, or total absence of shrinkage lead to more concentrated distributions

Figure 5.10: ROC and score distribution of the good test samples for various max depth parameters



but also an excess of wrongly labeled Good samples as Real = 0 or Gal = 1. The default learning rate has a shallower rise to the correct scores but it doesn't have as many confidently wrong labels.

Figure 5.11: Score distribution of the good test samples for various Learning Rates



The lower learning rate (high shrinkage) leads to learning less (or less) quickly which is why it helps get better results, but that is up to a point. As we can see for LR 0.01 in Figure 5.11 the Real scores look capped below 0.8 and the Galactic Scores looked capped just above 0.2 (should be 1 and 0 for these good samples). We can see how this 'cap' moves towards or near 0.5 as we lower the learning rate further with LR=0.001 (in the notebook we try 0.0001 as well which follow this trend but ends up dominating the plot so it is not shown) – see Figure 5.12.

5.2.4 Tuned Scoring Models

For the tuned models we use the following parameters:

- **Learning Rate:** 0.1
- **class_weight:** balanced
- **L2:** 1 (real), 10 (galactic)

Even though the ROC curves (see Figure 5.13) look virtually identical, we can see that the key differences between the two sets of models is in how the "good" alerts are scored: in the Score Space in Figure 5.14 they are slightly more concentrated in the right hand corner and the distributions in Figure 5.15 have marginally tighter profile.

Figure 5.12: Score distribution of the good test samples for lower orders of magnitudes of Learning Rates

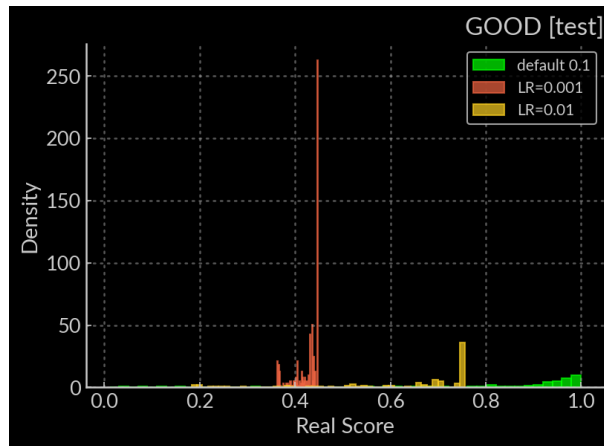


Figure 5.13: ROC of the galactic and real scorers. Comparison of the basic and the tuned models

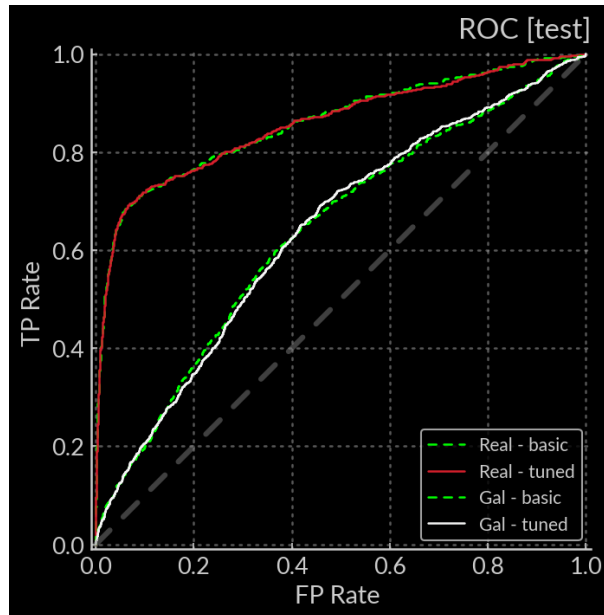
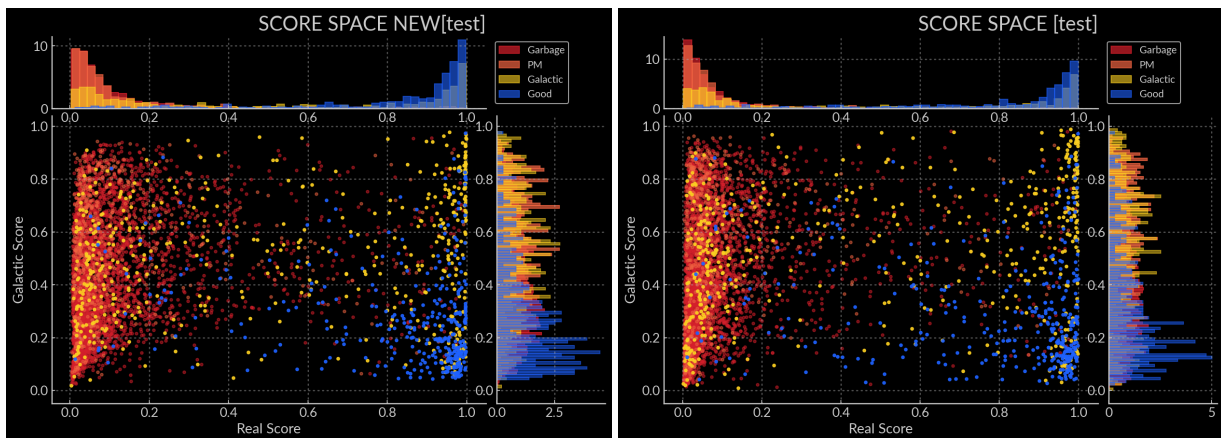


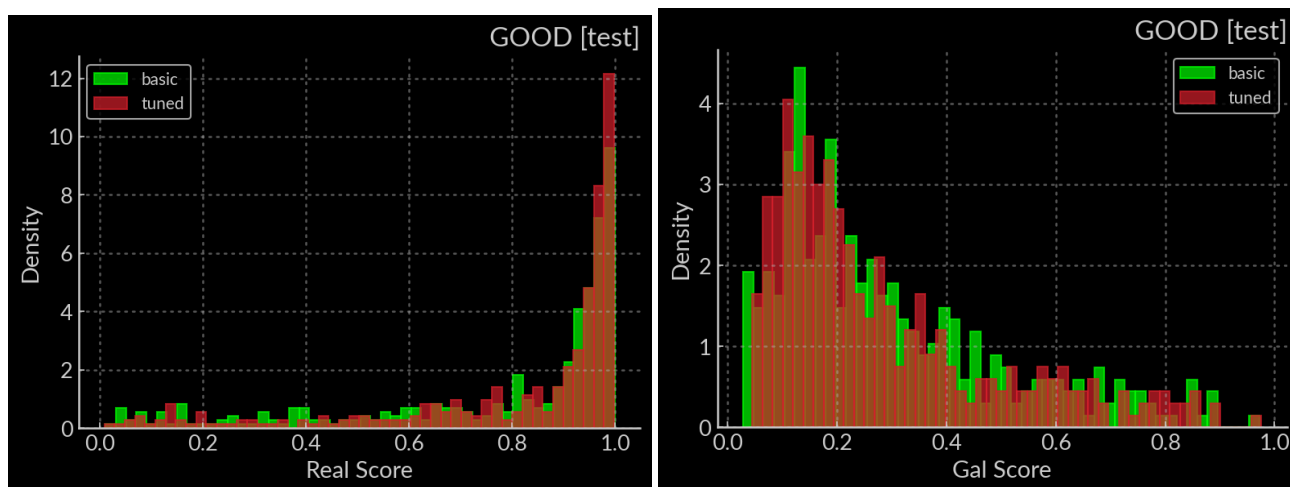
Figure 5.14: Score Space for the test set with the tuned Real and Galactic Scoring Models



5.2.5 Conclusions

1. The Galactic Scorer is struggling the most. It could be that we lack enough variety of data, but it will also be that the 'extra-galactic' category (galactic = 0) actually contains a diverse group of behaviour: a supernova

Figure 5.15: Comparison of the Real and Galactic scores distributions for the Good samples in the basic and the tuned models. (I should plot the run UNDER the green so you can see better)



is not galactic. A crappy image is also not galactic.

2. Tuning the hyper-parameters did not seem to make much difference. Probably because these methods are well established and the default parameters provide good results out of the box for most uses.
3. In the Score Space (Figure 5.14) the garbage looks a bit more smeared towards the Real end of the plot **I am not convinced the marginal improvements obtained from tuning will end up being improvements at all when ranking.**
4. It is likely that we would only see major improvements with either: **additional features** or **additional data**. It could be worth **trying to over-sample the smaller classes** instead of the under-sampling of the garbage class which we have currently implemented.

2025 hindsight: The galactic scorer was probably struggling because the garbage data had a strong correlation with the galactic plane and with less contextual and LC data BMO would rely more on Ra and Dec for decisions.

5.3 RANKING

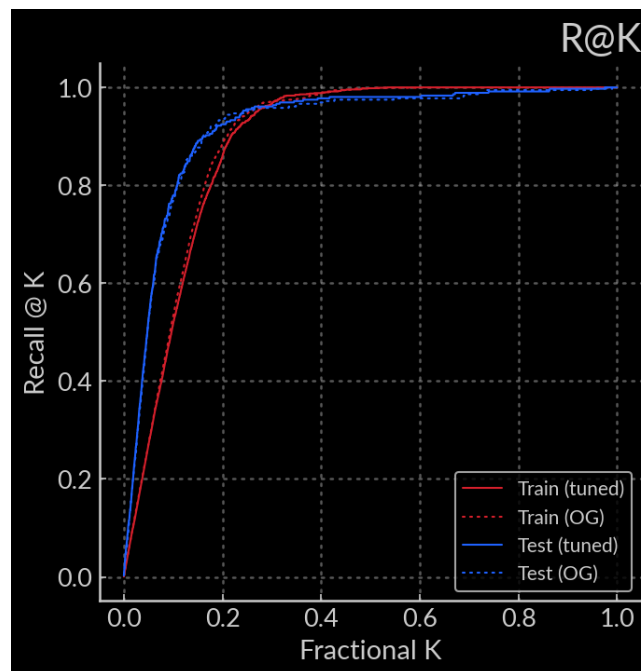
In order to truly assess how the models perform we have to use them for ranking and see how they would help (or hinder) the eyeball list.

We can see the recall isn't too shabby and the differences between the tuned and OG models are truly in the noise. It's very interesting that the test set has a higher recall within the top 25% of the list, **I think this is due to the fact that the test set is unbalanced (and there is a larger amount of obvious garbage) and indicates the model generalised alright.**

There are a few things that are good but don't rank very high though, since we don't reach 100% recall until very far down the list (we plateau around 95% recall in the top quarter and then stay quite flat). **This could be because some good objects look crap in ATLAS but are classified as such based on a TNS cross match.** I actually checked by eye all the objects that have Real < 0.2 in Figure 5.2. 15 looked like acceptable errors (because they were a cross match or because they only became solid good labels with more data and therefore they would be caught by the Update Scorer). 10 of them looked fine but some times it was the RB pix that was inexplicably low and I can't do much about that.

Doesn't look like tuning helped a lot...

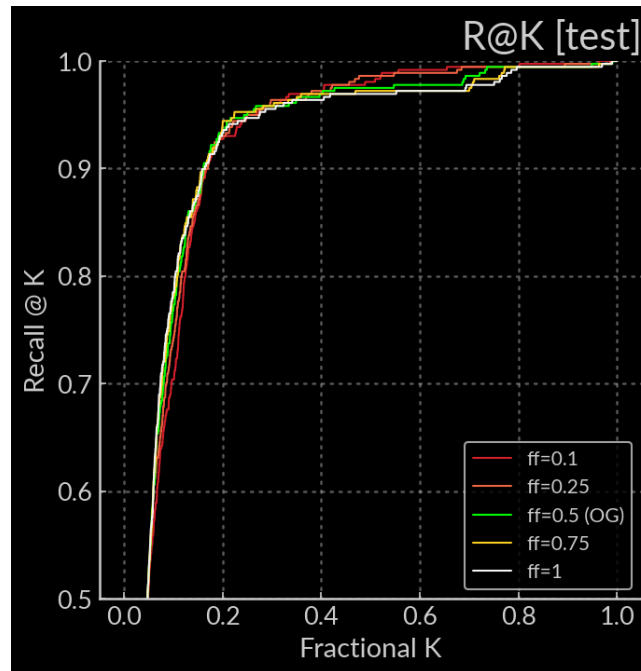
Figure 5.16: Recall at rank K for our training and test data scored with the OG and Tuned models.



5.3.1 Effect of varying the fudge factor

To investigate the effect of my fudge factor (see Section 6.5.1) we can vary it for a given set of models (we use the OGs, because why not). The lower fudge factors perform less well in the top 20% of the list and better in the last 45%. **Since we care most about the top of the list** we can stick to fudge factor = 0.5 for now.

Figure 5.17: Recall at rank K for the test set scored with the OG models and ranked with a range of fudge factors



5.3.2 Conclusions

The fudge factor and current ranking system can stay the same for now

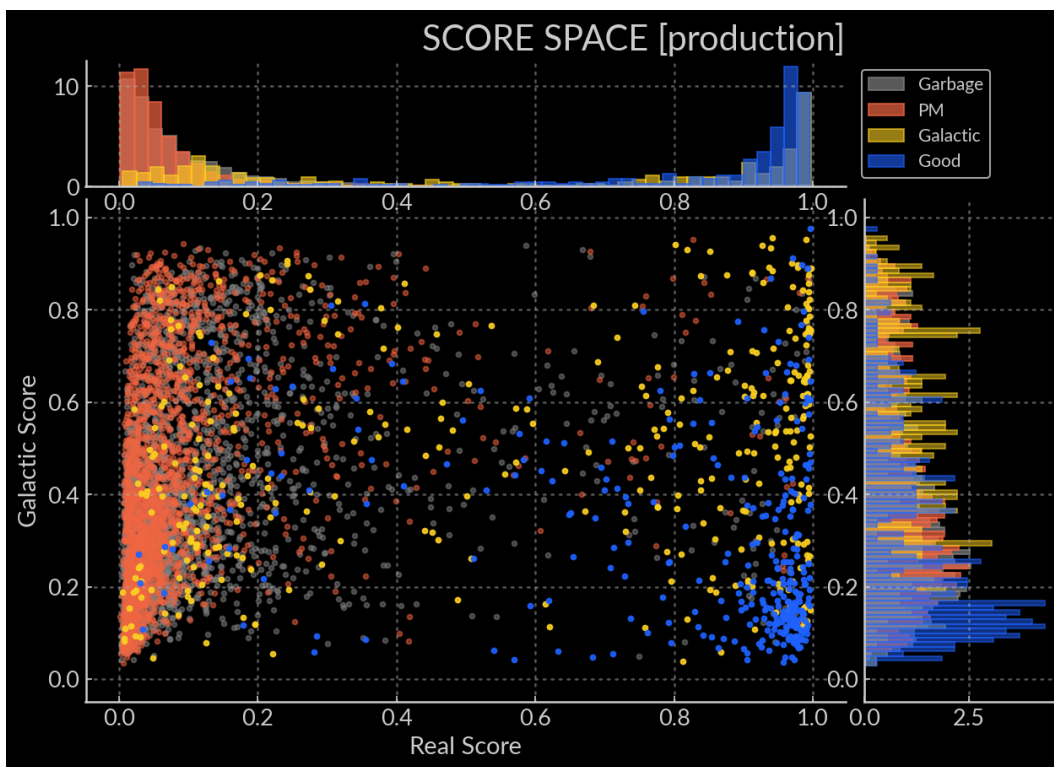
5.4 DAY 1 MODELS IN PRODUCTION PERFORMANCE

Me and Ken implemented the BMO Day-1 scoring models (the ranking remains unchanged since we determined the fudge factor of 0.5 was okay) on 18th July 2024.

Note to self: Need a better way to test the MakeFeatures() code - we were able to check it ran but I haven't been able to check it was making the features properly (e.g. the Sherlock classes could come out as all False all the time without throwing an error if I've done something arcane). This testing will be essential to check the features for the Update Score models are coming out right

We can plot the Score Space and see how well separated our categories are (see Figure 5.18) - not too shabby, nice separation. There are a couple of low scoring good objects which **are worth looking at manually**. The Recall at rank K (see Figure 5.19 is improved from Arin as it plateaus much closer to 100% recall. The rise of the slope seems a little steeper (especially for the first week in red) but it might be too early to tell. As we can see from the Arin plot (right hand side) there seemed to be some variation week on week (at the time there were issues with data download from Hawaii which may have resulted in low number statistics especially for the orange line).

Figure 5.18: Score Space and Recall for the first two weeks after implantation



5.4.1 Are the real scores different from the Real Bogus scores from the CNN?

And do they bring anything to the table. As we've seen earlier `rb_pix` is one of the most useful features and the Real Scorer is essentially doing the job again. To justify its existence we need to look at what the Score Space and the Recall at rank K would look like if we use the CNN's real bogus score as our real score. (Note that even if it makes no difference then the real score update is where the VRA would bring something truly new, but let's check if the Day-1 Real Scorer has a "*raison-d'etre*".

We can see in Figure 5.20 the effect of the Real Scores we calculate on day-1. The Recall at rank K rises faster, meaning that if we sort by VRA rank (rather than `rb_pix`) we get recover more good objects at the top of the list. For the first two weeks the lines seem to cross around 0.2 meaning that beyond the top 20% the ranking is equivalent. If we look at the Score space in Figure 5.20 we can see why this happens: The Real Bogus Score favours larger values and the distribution of the garbage population is smeared across the space. This results in more garbage residing at the top of the list when sorting by `rb_pix` only.

Figure 5.19: Recall at K comparison. Left: BMO, Right: Arin.

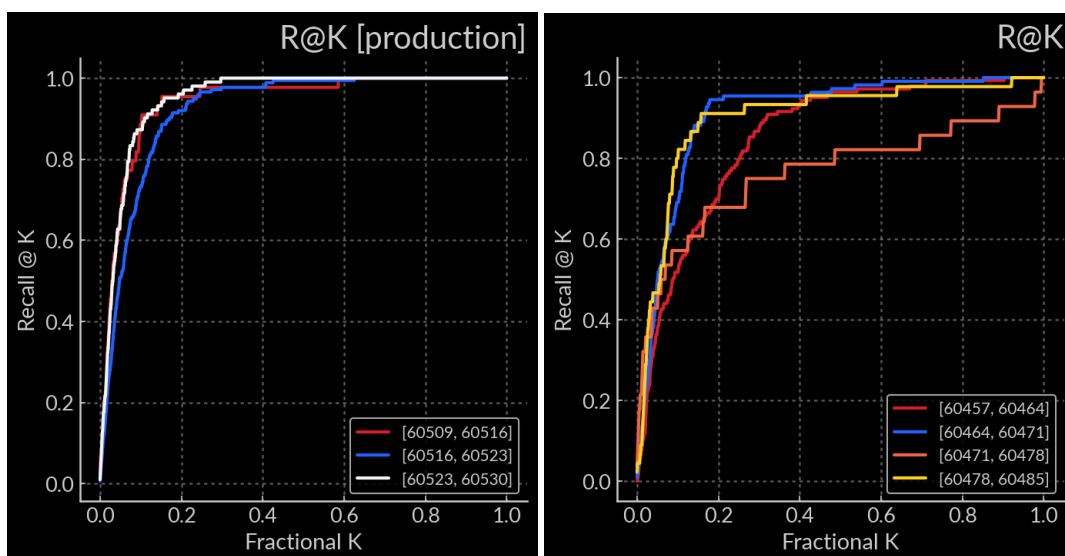
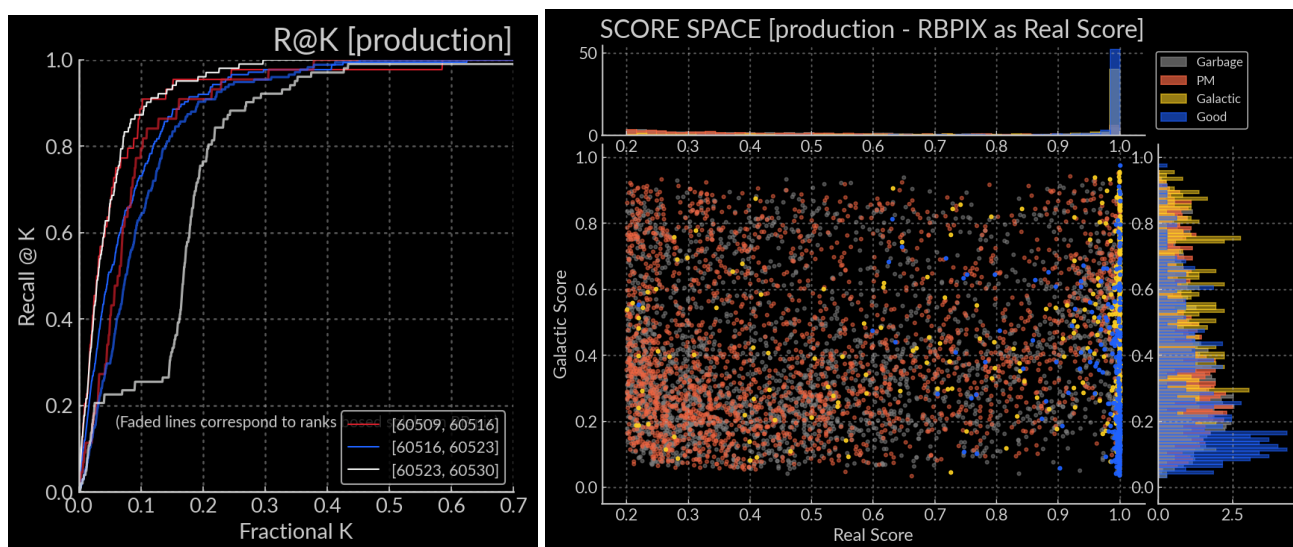


Figure 5.20: Score Space and Recall at rank K in production if we **substitute the Real Score from our day-1 model with the Real Bogus score (rbpix) from the CNN**. Note the rb_pix seems to have had a problem in that last week given how much contamination there is at the top of a list that is rb_pix sorted



It's also important to note that the good objects more consistently have very high rb pix values. This is good but only if we can mitigate the smearing of the garbage distribution, which leads to an impure eyeball list even when sorted.

We need to make sure that the lower ranks for some of the good objects in our Real Scorer is acceptable, that is, we need to ensure we will not forsake or delay the classification of interesting objects.

Some next steps to take are:

1. Visually check some of the objects that ranked low in the VRA
2. Try the lightcurve features of the Update Scorer in the Day-1 Scorer to see if we get better results and better recovery of good objects.
3. Implement some obvious hard coded decisions, such as: "if in TNS, rank = 9.999" (not 10 because sometimes the TNS is wrong)
4. Investigate if the Update Scorer will make the initially low scoring objects rise to the top of the list quickly enough to be looked at by the humans. It is possible (and visual checking will help determine) that the low

scoring good objects in the VRA had a messy light curve and were hardly notable in ATLAS but got high rbpix anyway because there seems to be a bias towards high values (beyond the 0.2 threshold that is).

5.4.2 Looking at some individual objects

Stephen posted links to a few objects that were ranked relatively low but were reported in the TNS. I did the following:

1. Recalculated the Day-1 score because I realised on August 11 that I had a bug in my light curve features I had created for the day-1 models so they weren't trained on good data.
2. Calculated the Update Score to see what would happen to these if the update scorer was actually working in production
3. Took the difference to see if they'd go up or down in ranking.

Table 5.3: VRA Ranks for objects flagged by Stephen

ATLAS_ID	day1	update	diff
1154423781352245400	3.33	1.46	-1.87
1010752701033040000	5.07	3.43	-1.64
1042730840113341600	3.49	2.97	-0.51
1172000660822613300	1.61	1.41	-0.20
1221600130231820700	1.44	2.19	0.75
1180859021435947700	1.60	3.73	2.13
1230610500195014700	1.92	5.64	3.73
1194651100235857000	2.78	7.87	5.09
1184055021381725200	2.12	7.95	5.83

Here is some more details on each objects:

- **1154423781352245400**: v. faint not surprising VRA not doing well here - should be raised to top of eyeball artificially when TNS cross match appears
- **1010752701033040000**: Not v clear in observed LC but obvious in FP. Shows a limitation of the current VRA: it doesn't use the forced phot. We need to consider within our new eyeballing strategy plan a new FP calculation condition which could open new features to the VRA for early decision making AND will be needed for the future parts of the VRA which will do long term monitoring with LC fitting.
- **1042730840113341600**: This one is very clear cut good but I know why the VRA got it wrong: the rbpix is low. 0.78 is down in the weeds when you look at the distribution
- **1172000660822613300**: Super faint and seen only once in the observed LC. like the first one it should be caught by a TNS cross matching clause
- **1221600130231820700**: Idem. Now this is very interesting though because **the VRA rank "rose"**. **This might be interpreted by a user as "the VRA believes it's more likely to be interesting now than it did on day1" but that's wrong**. These two models are calculated independently on a different lightcurve phase window and trained separately. The interpretation here is "the two VRA models are not keen on this one", but it's not an obvious one unless you're really familiar with the modeling. I need to think about this carefully because human-model interaction is crucial to things running smoothly. **[2025 hindsight: This is why I ended up passing all the day 1 features to day N]**.
- **1180859021435947700**: The update score is quite a bit higher but not high enough for my liking. It's the same lack of FP issue. With the sky background fluctuating we have some non dets in between the dets and maybe these "good" objects are not yet very present in the current training set. More data AND adding FP features will solve this.

- **1230610500195014700**: The day1 score is low because of the low RB score. The update is doing it's job though!!
- **1184055021381725200**: Dunno why the initial score is that low. And the update score is higher but not high enough to my liking given how many detections there are now. I'll think about this more
- **1194651100235857000**: Initial rank is low because of the history of weird stuff in the LC. but the update score really does its job here

5.5 UPDATE SCORES MODELS

Note: the "Update" models were later renamed "day N" models

5.5.1 Features

I picked the following features described in Table 5.4. The light-curve related features are different from that of the Day-1 Models and now that I've done these **there may be a case to try training the Day-1 Models on these features too** and see if it works better or worse.

Table 5.4: **Features** used in the Update Scoring models. Max mag and Mag mag day were inspired by the BTSBot paper. **We only use data from dayN -5 to + 15.**

Name	type	Description
dayN	int	phase w.r.t alert. Phase 0 is (usually, unless there are delays) the first bundle of data to hit the eyeball list. It has <code>phase_init</code> values between -1 and 0 (data to first hit the eyeball list was observed just <i>before</i>).
DET_mag_median	float	median magnitude of the detections at day N
DET_N.today	float/int	Number of detections at day N
DET_N.total	float/int	Number of detections since day -5
NON_mag_median	float	median magnitude of the NON detections at day N
NON_N.today	float/int	Number of NON detections at day N
NON_N.total	float/int	Number of NON detections since day -5
max_mag	float	Maximum magnitude value up to this day N
max_mag_day	float/int	day N when maximum magnitude was reached.
ra	float	Right Ascension
dec	float	Declination
rb_pix	float	Real bogus Score from Josh's CNN
SN	bool	(sherlock classification) if SUPERNOVA
NT	bool	(sherlock classification) if NUCLEAR TRANSIENT
ORPHAN	bool	(sherlock classification) if ORPHAN
CV	bool	(sherlock classification) if CATAclysmic VARIABLE
UNCLEAR	bool	(sherlock classification) if UNCLEAR

5.5.2 Some discussions

Why not use the scores calculated on Day 1?

I thought about using the Day-1 scores as a 'starting' point. After all isn't it, oh so Bayesian, to update one's belief. Problem is 1) I don't know that I can say with certainty that what's happening in the models is Bayesian strictly speaking 2) The priors in Bayesian stats carry a lot of weight unless you have a lot of data.... here we'll never have a lot of data (maybe a week's worth at most) so a poor initial guess could screw us over. Instead I opted to include the most relevant Day-1 info in the training set, so we're not ignoring that information.

Q to Steve: Is what I've just said sensible or not?

Table 5.5: **Balanced training set Numbers**. These are different from the previous data set because each ATLAS_ID has several data sets, one for each phase (dayN) that has data (whether it be detection or non detection). Each of these are **seen as independent**. I sub-sampled the Garbage which is why it has a round number

Type	N
garbage	5,000
galactic	5,137
good	4,521
pm	6,715
TOTAL	21,373

Why is each phase (dayN) looked at independently?

The option to look at the time series as a whole is appealing to an astronomer and a human that can process time series easily with their very own pattern recognition machine, but it is problematic in a few ways.

1. Practically speaking it's annoying, because we would be dealing not just with features on a time series but an actual time series. One with varying sparseness and including both detections and non-detections. Easy for a brain, absolute pain for a computer.
2. We want to be able to handle an arbitrary range of **partial** lighthcurves. The features I've created (the ones relating to the number of detections or maximum magnitude *up to this point*) are dependent on the full lightcurve we have **so far**(from phase -5), so **each dayN is 'knowledgeable' about past data**.
3. It gives me extra samples for free, because most ATLAS_ID have information for several phases. Woohoo!!

5.5.3 Results

well... it looks like it's working! We'll have to see how it performs in prod, and frankly there's not a sure way to simulate that. Better to just try it live asap!

Figure 5.21: Score Space for all the samples (note that means the same ATLAS_ID will be shown as many times as it has phases in the data).

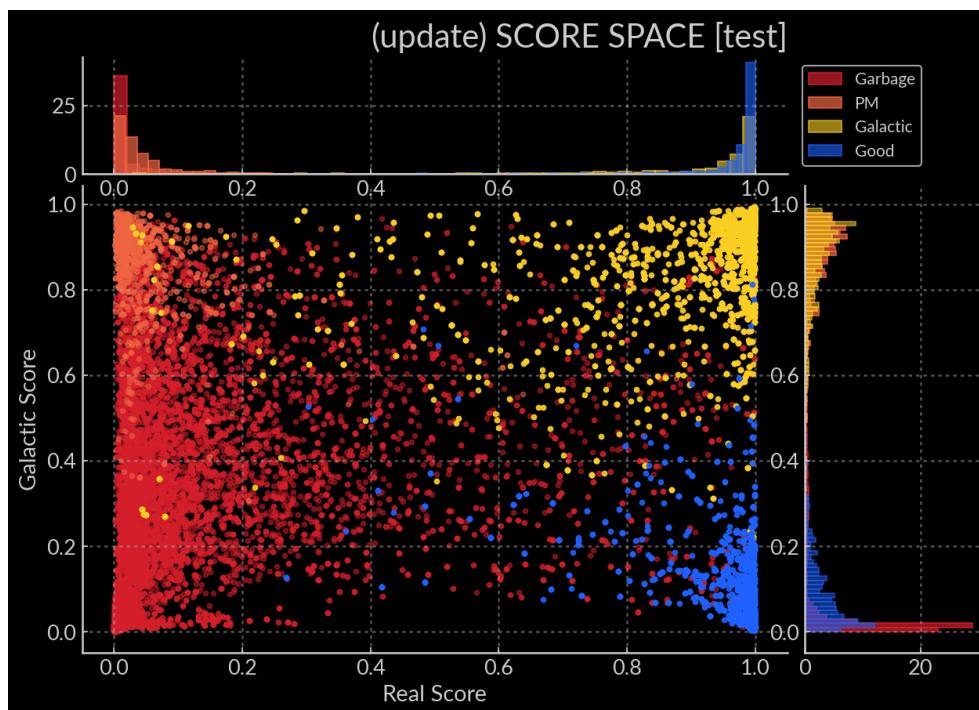


Figure 5.22: ROC for all samples

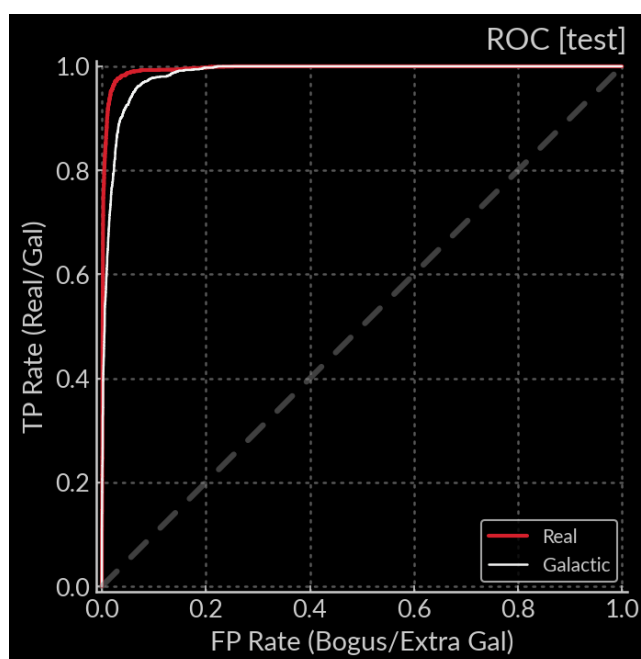
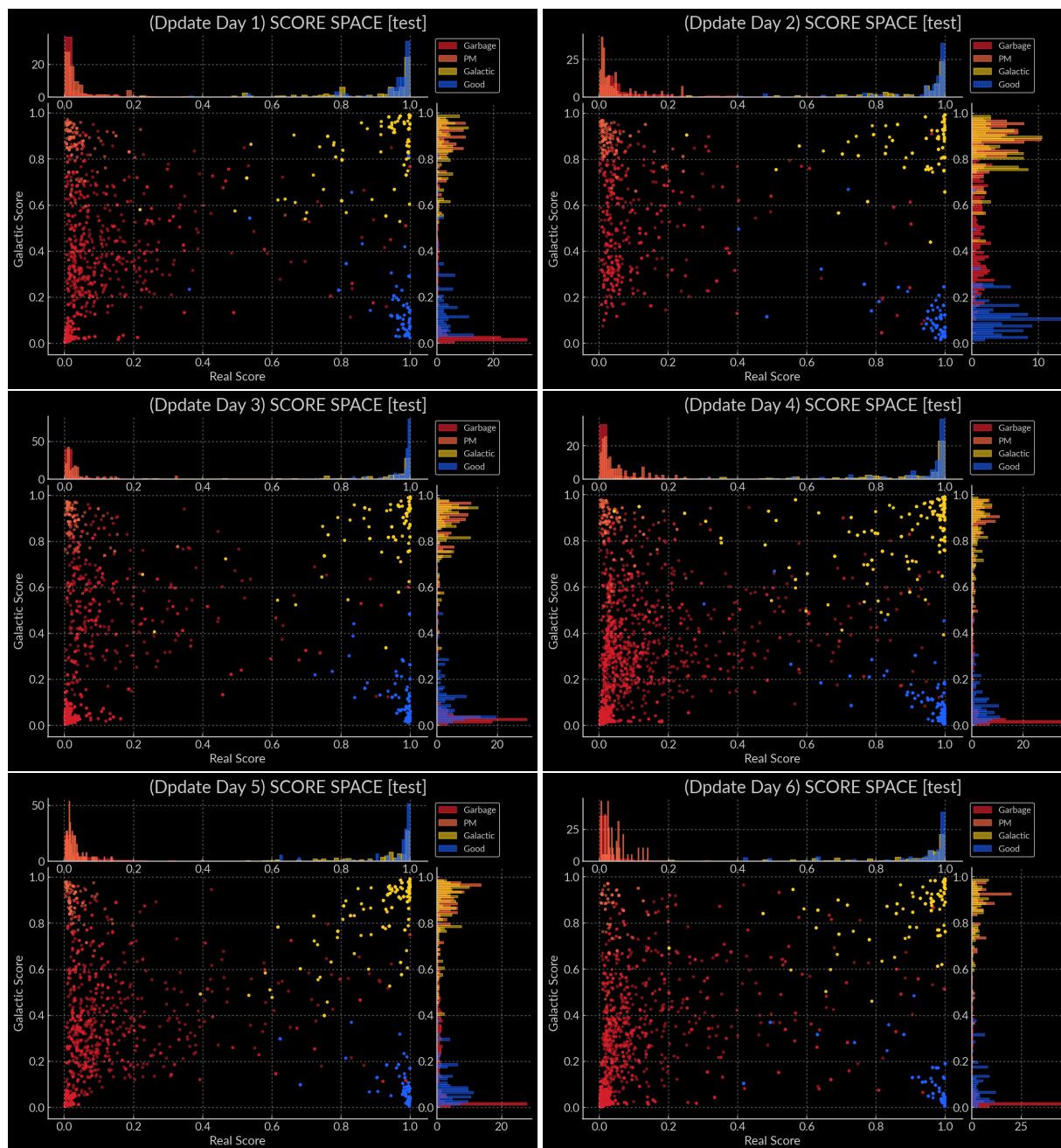


Figure 5.23: Score Space for samples as specific day N (phases). (didn't show phases 6 to 15 because dear god this is enough plots already)



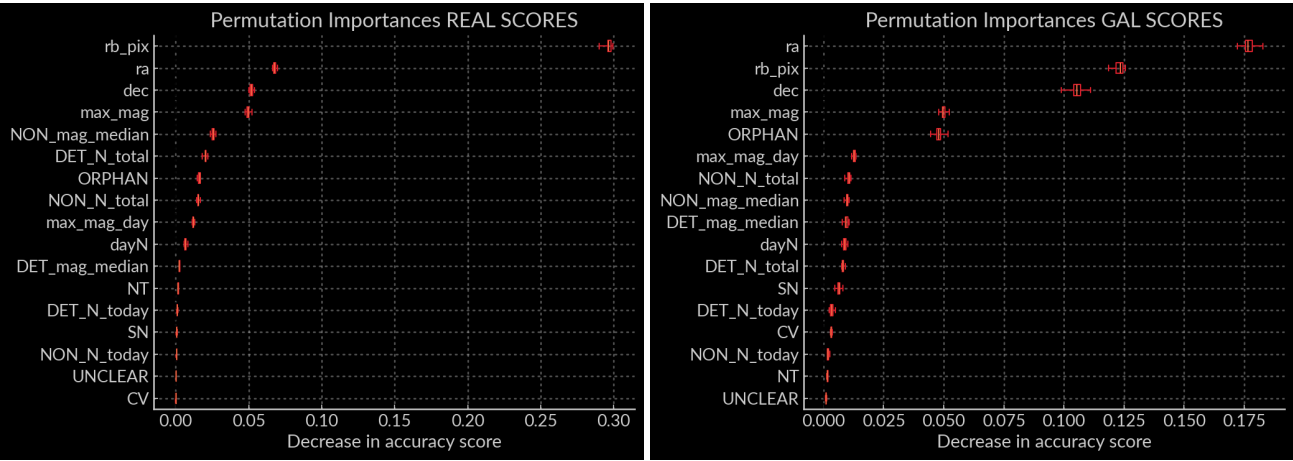
5.5.4 Permutation Importance

See Figure 5.24. Some Sherlock Classes don't pull there weight just the same as the Day-1 Scorer, like CV and UNCLEAR but it could be because there's not many of these in the training. I don't have many thoughts on this right now because I'm writing this at the eleventh hour before I go on holiday¹, but **could be worth training models without some of the lower scoring features and see if we can make 'lighter' alternatives that perform just as well.**

NOTE: Since the RA is important (here and in the Day-1 scoring model) we are going to have to **retrain regularly in the first year**, as our sample currently does not span all possible RAs.

¹note 28-02-2025: This indicates the chapter must have been written on 3rd week of July 2024

Figure 5.24: Permutation Importance for the Real and Galactic Updater



5.5.5 What Next?

First of all I need to implement the code in `st3ph3n` to make the features of the Update Score model on the fly in the ATLAS pipeline.

The next important step when that is implemented and score updates are happening in production is to **think about how we are going to start testing the new eyeballing strategy and how we are going to measure success or failure of the update model**. Because currently the eyeballers empty the eyeball list every day, we can't simulate how specific objects would rise in the list compared to others and how long it would take for them to get the attention of the humans.

5.6 UPDATES TO SCORES AND RANKS - TESTING THE NEW EYEBALLING STRATEGY

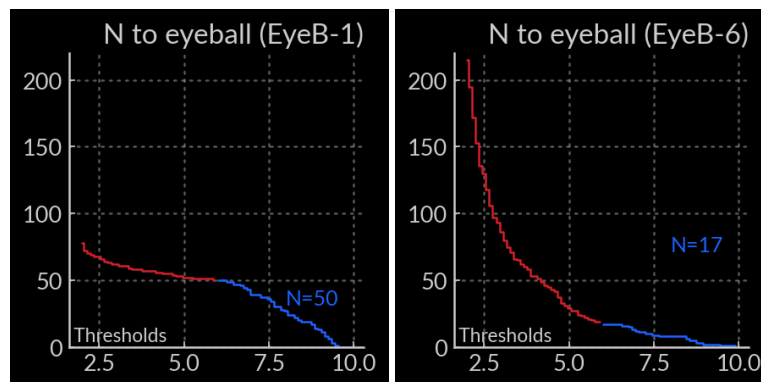
On August 14 we added the Update Score Model (later called day N models) to the ingest pipeline and began testing the new eyeballing strategy where only events with a rank $> X$ are eyeballed and the rest are left in the eyeball list for their rank to be updated at a later date and be removed automatically by the VRA.

5.6.1 Selecting the VRA rank threshold to eyeball

As one can see in Figure 5.19, the ranked top 10% of our data in production (on day 1) recovers 95% of the good objects. The other 5% either need more data to be convincing or are simply faint for ATLAS and would be found through cross-matching with the TNS.

We find that the objects that fall in that top 10% have ranks > 6 . To check that this threshold is workable we can check how many objects we would need to eyeball for a chosen threshold - see Figure 5.25. Overall we find our eyeballing ranges from 6 to 50 (only once) depending on the backlog.

Figure 5.25: N to eyeball for a chosen threshold. The blue portion of the plot indicates the Ns for rank 6 and above. The number of objects to eyeball is marked on the plot. We show the plot for the first eyeball list of the test, and for an eyeball list 3 days into the test. The first list has more to eyeball because there was a backlog, and the latter has a large number of objects at the low thresholds because the eyeball list has not been emptied every day as usual. These will be handled by our automated garbaging conditions.



We must also ensure that this threshold is relevant to the ranks assigned by the Update model. In Figure 5.26 we can see what mixture of alert types eyeballers would be dealing with. We also checked if this would lead to missing a transient completely, i.e. having one transient whose maximum rank never reaches the threshold. Only one out of 144 was in this situation (1075454201535047200) and I don't think a human would be convinced this is real in ATLAS so it is an acceptable loss for now. (It would have been found by the TNS cross match).

5.6.2 Selecting automated garbaging conditions

In order for the eyeball list to not grow forever we must set some auto-garbaging conditions. The first type of logic we test is "If object sky location has been seen more than X times and the maximum rank never rose above Y, put in garbage".

The first test was using $X \geq 3$ and $Y \leq 2$. This only removed roughly 25 to 30 objects at a time (all were eyeballed manually to check for real transients but none were found).

To explore a more stringent deletion strategy we can create a similar plot to Figure 5.26 but for ranks less than Y (in this case 3) - see Figure 5.27. As we can see there are only a few galactic alerts (actually "attic", which is not a pure sample) and one good alert (the one mentioned above).

Testing the Logic "if alert has at least two observations (day 1 and another) and a maximum rank < 3 " we select large portions of the eyeball list (248 on day 3 on the 6th eyeballing - although that number would be lower if we had applied that logic earlier on). In that list one object turned out to be a real CV but it was very close to a star spike (1223123311634958900)

We can monitor if this logic is sufficient to keep the eyeball list number stable.

Figure 5.26: Mixture of types **in the test data** for day N after initial alert

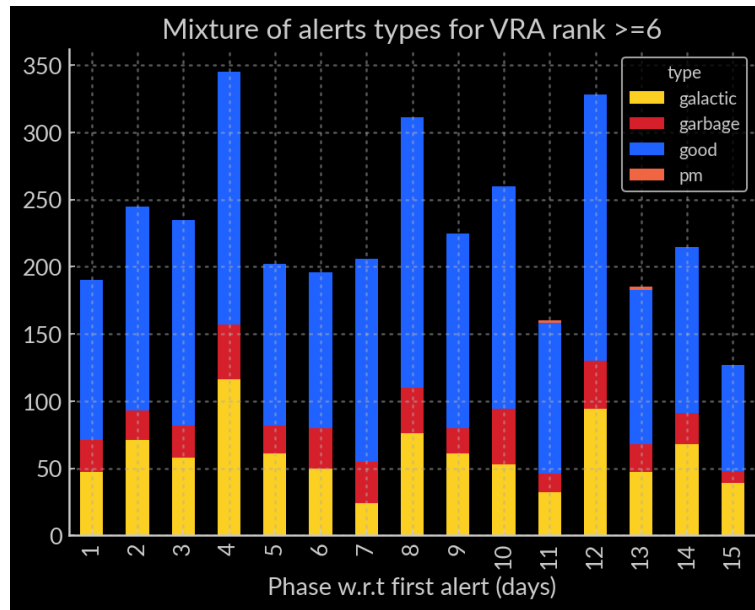
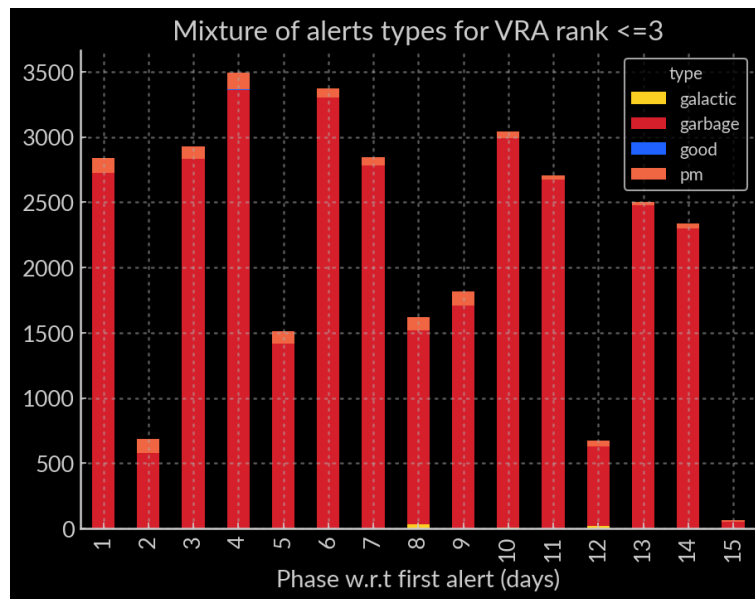


Figure 5.27: Mixture of types **in the test data** for day N after initial alert for rank < 3



There are potentially other low-hanging fruit avenues: the psf measurements are given in the data retrieved from the API and we could **explore discarding alerts based on psf measurement thresholds alone**, this way they don't hang around in the eyeball list even for a couple of days.

Note: the psf measurements major, minor and phi were briefly tried as features and not found informative. Likely because they double with rb_pix

5.6.3 Eyeballing strategy performance

After 3 days, the new eyeballing strategy leads to eyeballing 84 objects out of 786, or 10.7%. This is in line with expectations from our selection of the rank threshold. We found 47 good objects, 14 of which were previously classified by TNS. The latter were checked individually to see why they were "missed" - some had a high VRA score (> 6) but were reported already, some had low rp_pix scores (which should be addressed separately) and others were simply too faint. There were no obvious oversights.

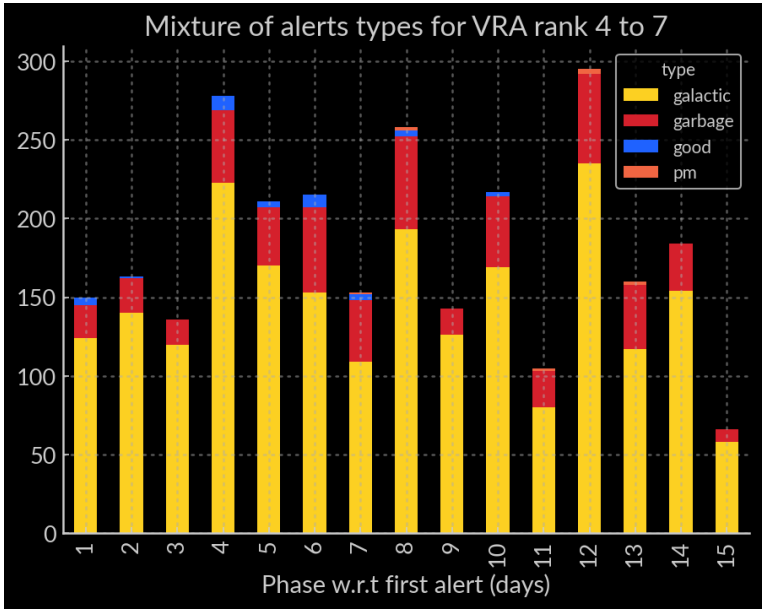
Our retrieval rate is 5.9% which is in line with the rate of "Good" objects in our samples of the last few months (see

Figure 5.1).

5.6.4 Eyeballing Strategy for Galactic Transients

Finally we can ask what could an eyeballing strategy look like for the galactic folks. Although this is not being actively tested right now we can check what an "galactic eyeball" list would look like for different ranges of VRA rank. We find that for ranks between 4 and 7 eyeballers would have a good proportion of galactic alerts. This range would miss 7.5% of the galactic sample in our data but $< 5\%$ would be truly lost as the rest would be found by the Extra galactic eyeballers.

Figure 5.28: Mixture of types in the test data for day N after initial alert for ranks between 4 and 7.



Note that to do a good job of the galactic stuff we will need to clean up the "Attic" sample I have! Some of these may be AGNs or duplicate extra galactic transients

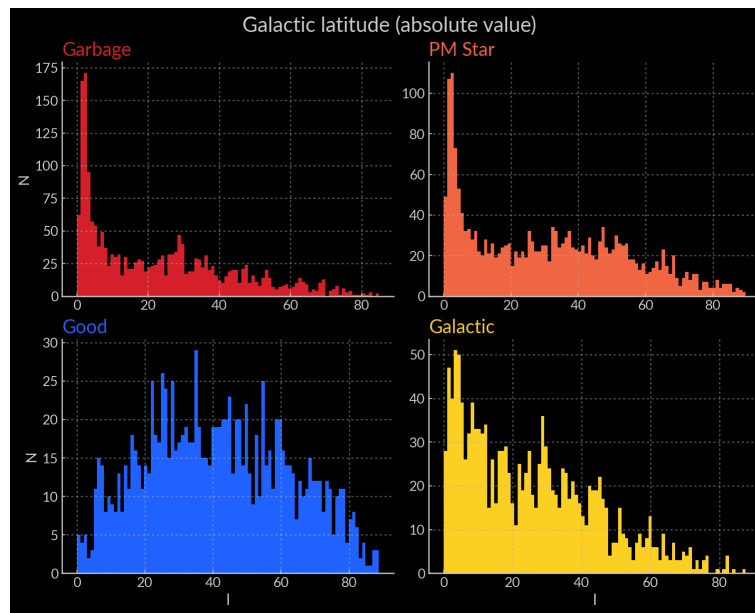
5.7 FURTHER FEATURE ANALYSIS

5.7.1 Using the Galactic latitude as a feature

We thought that including galactic latitude as a feature would be a dead giveaway and strongly help separating the galactic objects from the extra galactic objects.

Instead we found that including the galactic coordinates as features decreased the efficacy of the VRA's scoring leading to much more scattered "good" and "galactic" Pgal distributions. Figure 5.29 shows why that is the case: the overwhelming effect in the galactic latitude distribution is that low values are much more likely for garbage and proper motion stars. The galactic transients also prefer low values but the slope is much shallower.

Figure 5.29: Distribution of the galactic latitude b split by alert type.



We explored using galactic latitude only for the Preal model but it was no better than just ra and dec².

Ra and dec are good enough predictors AND they don't require another on-the-fly calculation so we leave it at that.

5.7.2 Culling features in day 1 model

When adding the galactic latitude to the model it did score highly in the permutation importance but made the model worse non-the less. That is because permutation importance is only a mathematical tool to assess if the values of the feature help predictions over the whole data set, but we specifically care about the good and the galactic objects being more tightly grouped in their corner of the Score Space and being separated from the garbage. If the PM and garbage distributions are more clearly separated we don't care (for example) but the model would could that as a win.

As a result we check manually how removing some individual features from the model affects the scoring and ranking. The details are in the jupyter notebook³ and frankly not super interesting. The OG set of features performs best.

What will be interesting later is to try including a couple other features: **PSF measurements**, turning the Sherlock SN class into a **distance to galaxy**. Given how few real transients there are, I think the latter is more effort than its worth. The PSF measurement could help kill the garbage early.

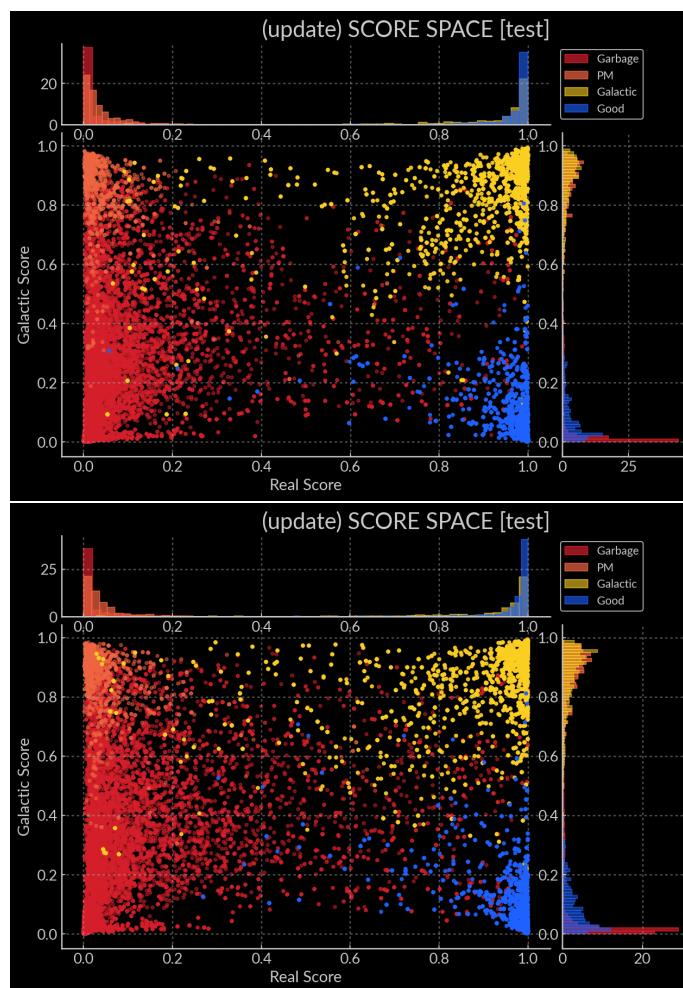
²(Scoring_models.tuned.try_removing_features.ipynb)

³(Scoring_models.tuned.try_removing_features.ipynb)

5.7.3 Retraining Update Scores Models including day 1 features

The Update Score models as trained above had the issue to be **TOO INDEPENDENT** of the day 1 model, some objects looked like their score decreased when they were actually good. The issue was that the Update Score model **did not know about the lightcurve history features!**. Adding these to the model improves scoring. We can see in Figure 5.30 that the distributions are more separated.

Figure 5.30: **Left:** New Update Score Space calculated including the day 1 features. **Right:** Previous Update Score Space



6

Arin - benchmarks and first tests

Model Naming Convention: Instead of numbering each model or prototype, each family of prototype (or release model) will have a name. First starting with A, then B, etc... The names will be gender neutral, the models will be referred to using "it" not "they". A named family of prototypes may have a number of model updates (e.g. hyper parameter tuning, even different model types), but they will have **the same training data set**.

6.1 RAW DATA GATHERING AND CLEANING

Date cutoffs: 2024-03-27 to 2024-05-14.

All of this was done on Heloise's machine and later saved to a hard drive. The locations are relative to the hard drive directory Arin.

Name: `dwlnld_data_obj_with_decisions.py` **Location:** `./raw_data/scripts`

- Get last run date
- Add the the log the current date
- Get VRA scores table rows added since last run date
- If data frame is empty log that no data was found
- If not data frame has rows, find the rows that correspond to human decisions (username column not None)
- Get unique list of ATLAS_ID with decisions
- Request the data of each event that has a decision
- append the VRA scores columns of the data with decisions to the `vra_with_decisions.csv` file
- For each new event with a decision, write their data to a `.json` file in the output directory.

Note that this script is run by a bash script of the same name (apart from extension) which calls the python script with the necessary grep command to find the the last run date from the log file and parse it as an argument python `dwlnld_data_obj_with_decisions.py 'grep "Finished" vra_dwlnld_data_obj_with_decisions.log | tail -n 1 | cut -d',' -f1 | sed 's/ /T/'`

Name: `Dashboard_PDA_eyeballer_and_training_data.ipynb` **Location:** `./training_data`

This jupyter notebook is a cleaner version of the first exploration of the data. The data it creates is summarised in the follow few tables.

FUTURE IMPROVEMENT: When creating future data sets **need to include a non-detection date cutoff**. It took over an hour to gather the non detections for 12 thousand or so objects. **Without a cut off this will be too slow for the live server.**

Table 6.1: Columns for the following table `vra_last_entry.csv`. Only the last VRA rows for each ATLAS.ID (i.e. the row containing the human decision) plus some extra columns. **Shape:** 12,376 x 12. **Index:** ATLAS.ID. The OG columns are directly from the VRA scores table whereas the EXTRA columns were added in the data gathering and cleaning steps.

Columns	Type	Descriptions
id	int	OG
preal	float	OG
pgal	float	OG
pfast	float	OG
timestamp	str	OG
apiusername	str	OG
username	str	OG
debug	bool	OG
mjd_decision	float	EXTRA: Modified Julian Date of when the decision was made
mjd_init	float	EXTRA: Modified Julian Date of when the event first entered the eyeball list.
ndays_to_decision	float	EXTRA: How many days between initial alert entered eyeball list and decision was made (<code>mjd_decision - mjd_init</code>)
type	str	EXTRA: Type of alert, as labeled by the humans (see Table 2.5)

Table 6.2: Columns for the following table `contextual_info_data_set.csv`. **Shape:** 12,376 x 5. **Index:** ATLAS.ID.

Columns	Type	Descriptions
ra	float	Right Ascension
dec	float	Declination
rb_pix	float	RB score from the CNN
sherlockClassification	str	Classification from Sherlock (e.g. SN, OPRHAN etc.. see sherlock manual)
type	str	EXTRA: Type of alert, as labeled by the humans (see Table 2.5)

Table 6.3: Columns for the following 3 tables. `detections_data_set.csv`: All detections since and up to the date cut offs for this data set. (**Shape** 107,514 x 8). `non_detections_data_set.csv` all non detections since the beginning of ATLAS and up to date cut off (**Shape**: 30,443,520 x 8) and `non_detections_data_100days.csv` All non detections from 100 days before the alert entered eyeball list (**Shape**: 2,067,953 x 8) . **Index:** ATLAS.ID

Columns	Type	Descriptions
mjd	float	Modified Julian Date of the recorded detection
mag	float	AB magnitude
magerr	float	error on the magnitude
band	str	ATLAS filter
det	bool	Whether this row is a detection or non detection.
phase_init	float	N days since it first entered the eyeball list. Note there may be valid detections before if they were below the ingest cut thresholds (not 5 sigma).
phase_decision	float	N days since the humans made a decisions
type	str	Type of alert, as labeled by the humans (see Table 2.5)

BENCHMARK 0: Data cleaning and feature extraction for score and rank calculation needs to be fast enough to run in the live ingest.

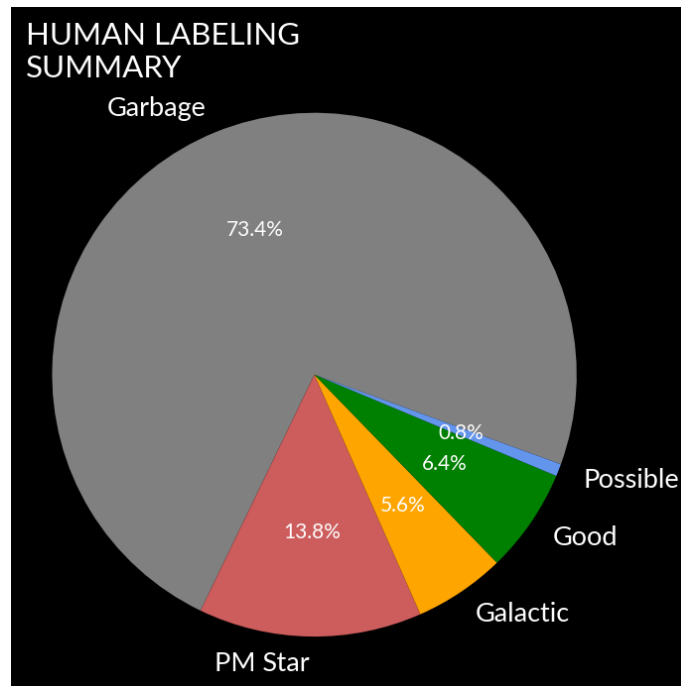
Note: When creating the data, I had to remove some objects with decisions because they were first ingested before the VRA scores table was recording and therefore they don't have the `mjd_init` needed for a complete row.

6.2 EXPLORING DATA AND SETTING BENCHMARKS

6.2.1 Alert type fractions

Plotting the label composition of the data reveals how much garbage still makes it through the eyeball list. Note that both the **Garbage** and **PM Star** labels are **bogus** and so the total fraction of undesirable alerts is over **86%** of the eyeball list.

Figure 6.1: Label composition of the data



BENCHMARK 1: We want the new eyeball list to be filled with **85%** undesirable alerts.

WARNING: Sometimes "good" objects are CVs that are never removed from the good list - should check whether the couple hundred good transients here are genuinely extra galactic.

6.2.2 Human decision timescales

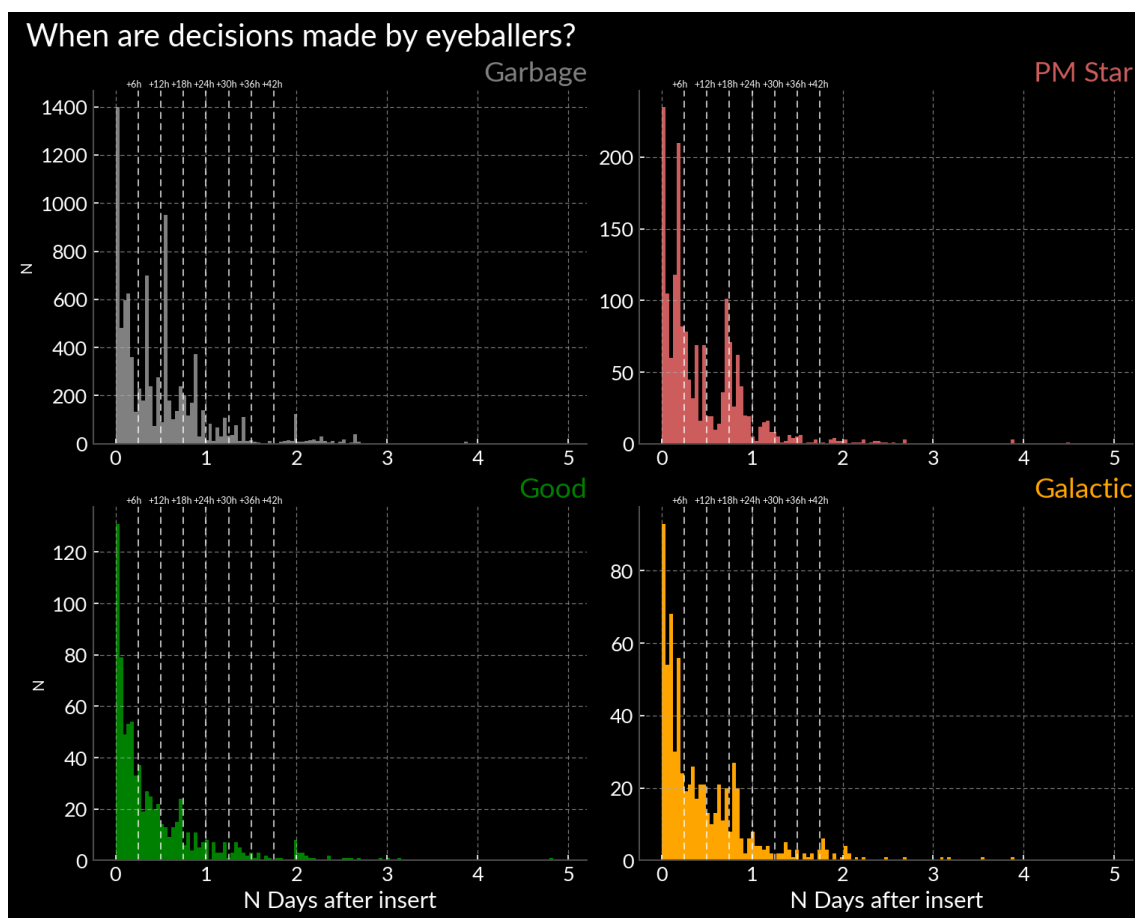
For this we exclude the "possible" alert type as it does not seem very informative at this point. Ultimately the new eyeball list won't make use of the possible list - we will just leave alerts in the eyeball list and as more information comes to light they will rise or fall in the ranking.

In Figure 6.2 we can see that most of the good objects are classified as such **within 1 day** of first entering the eyeball list. Nearly all within 48 hours. The secondary peak at 2 days is interpreted as being due to additional data being made available to humans after a couple of days (as ATLAS visits this part of the sky again), but note that **this hypothesis has not been checked**. *This can be verified by seeing if a new detection has occurred between the first alert and the decision being made for these specific objects.*

BENCHMARK 2: We want **all good objects** to make it to eyeball it on **first day** - and preferably rank high.

This speed requirement (so as not to slow down the humans) means that we are likely to keep quite a bit of trash (at least at first) but if we can have less than 85% and order it in a useful way for humans it'll be a win.

Figure 6.2: How quickly do the humans make decisions split by alert types



6.3 TRAINING SET FEATURES

6.3.1 Creating Light Curve history features

Figure 6.3 shows the distribution of detections for each alert types. Main take aways are:

- "good" don't have history nearly as frequently as the other classes.
- garbage could be separated from the other alert types by looking at post alert behaviour up to 5 days where the other alert types (especially) are very likley to have detections.

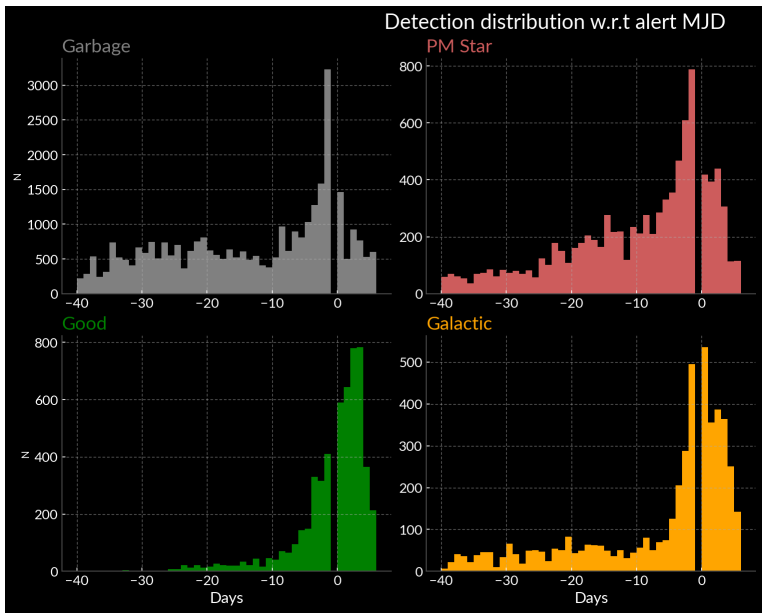
Since we need to make decent decisions very quickly so as not to slow down the humans, **we are going to have to rely more on the light curve history than continued evolution for now**. The continued evolution and the update of the scores over time is something we can look at later when Arin is ranking new alerts live in the new test eyeball list.

The goal of quantifying the history of the light curve is to see if that event location is prone to bogus events or if there may be recurrent flares from a galactic phenomenon that is not reported in catalogues that Sherlock has access to. Looking at the past behaviour of the lightcurve is one of the key steps in human eyeballing and we need to find a way to harness past data for classification.

To quantify the lightcurve behaviour we calculate the following from the aggregated detections + non detections (since day -100) tables (**cropped to only contain data before phase_init == 0**): 1) Count the number of detections 2) Count how many non detections occur before each detection. To do this we find the first detection in the historical lightcurve, remove it and all non detections before, and then count the number of non detections in the intervals between detections (PROBABLY NEED A GRAPH).

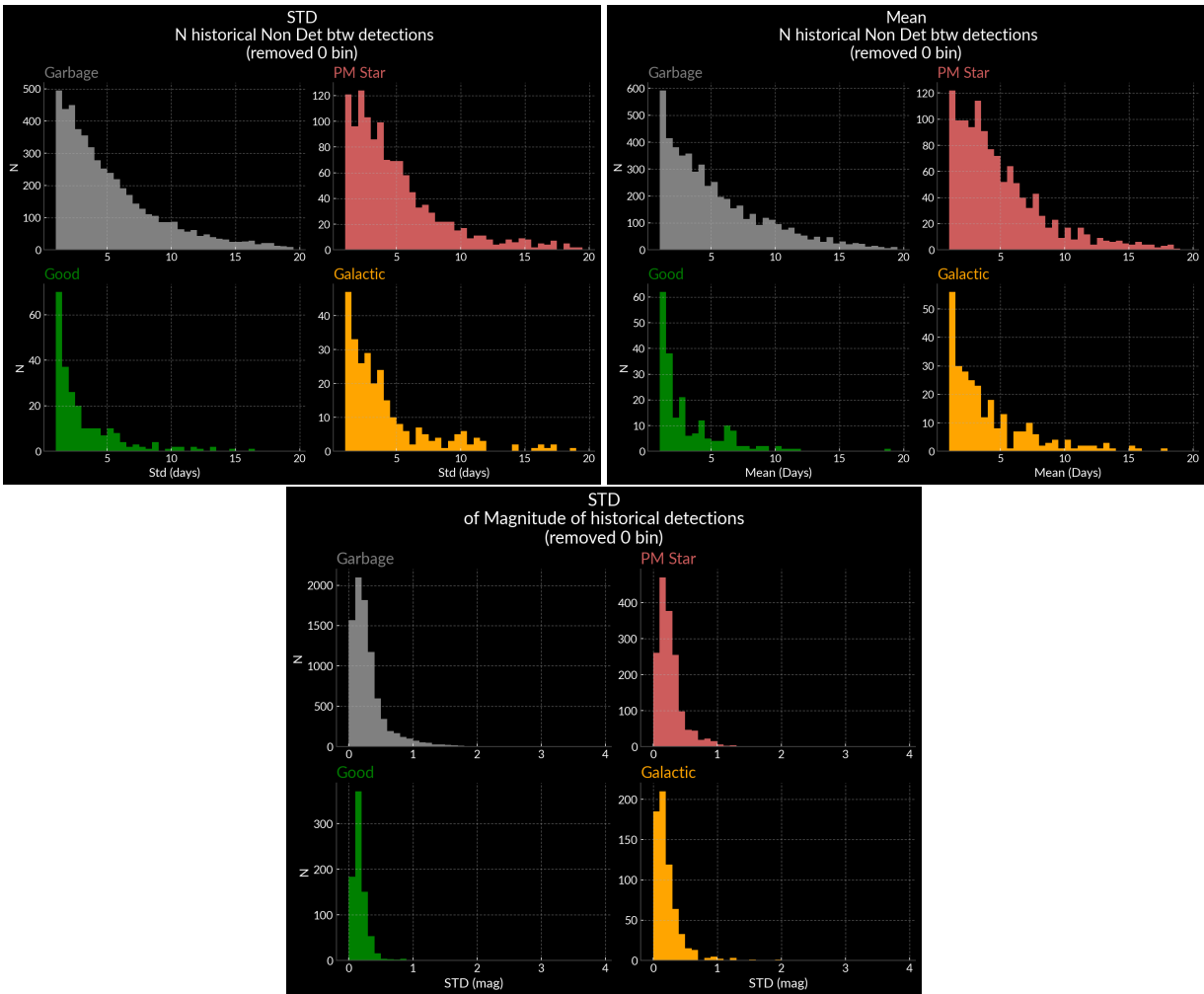
Caveat: The latter doesn't take into account if a non detection is brighter than a previous detection - this may need looking at.

Figure 6.3



The features we derive are : The standard deviation and mean of the number of non detections between each detections, the standard deviation of the magnitude of historical detections. See Figure 6.4.

Figure 6.4





6.3.2 List of features used for training

The list of features used for training the preal and pgal score classifiers is summarised in Table 6.4. In a first instance we chose some very basic feautres: the quantities describing the historical lightcurve as described in the previous section, the location on the sky, and the information passed on by the CNN classifier and the Sherlock algorithm.

Table 6.4: Summary of features used for training the Real score and Galactic Score classifiers

Feature	Type	Descriptions
Nnondet_std	float	Standard dev. of the number of non detections between detec-tions
Nnondet_mean	float	Mean of the number of non detections between detections
magdet_std	float	Standard deviation of the magnitude of the historical detections
ra	float	R.A.
dec	float	Dec.
rb_pix	float	Real Bogus score
sherlockClassification	categorical	Sherlock classifications

[More features should be tested](#) in variations of the Arin Virtual Research Assistant.

6.4 ASSIGNING "REAL" AND "GALACTIC" SCORES

6.4.1 First model

Training-test split and cross validation

The data were split 80/20 and **no cross-validation** is performed for this first pass. The training data contains 9825 events and the training set is **not corrected for class imbalance** - for example we note that 73% of the events in the training set are garbage.

FUTURE IMPROVEMENTS: We will need to address bias in the sample, and we should implement cross-validation to get better results.

Model choice: HistGradientBoostingClassifier

For the first model I went for the `scikit-learn` implementation of the histogram-based Gradient Boosting Decision Trees as it natively **supports categorical data** which is very handy because **we want to use Sherlock classification**.

Two **separate models** are trained to predict **preal** and **pgal** independently. **FUTURE IMPROVEMENTS:** Would it be better to have a single model that predicts both simultaneously? Discuss pros and cons with Steve (and if yes, then what family of models can we use).

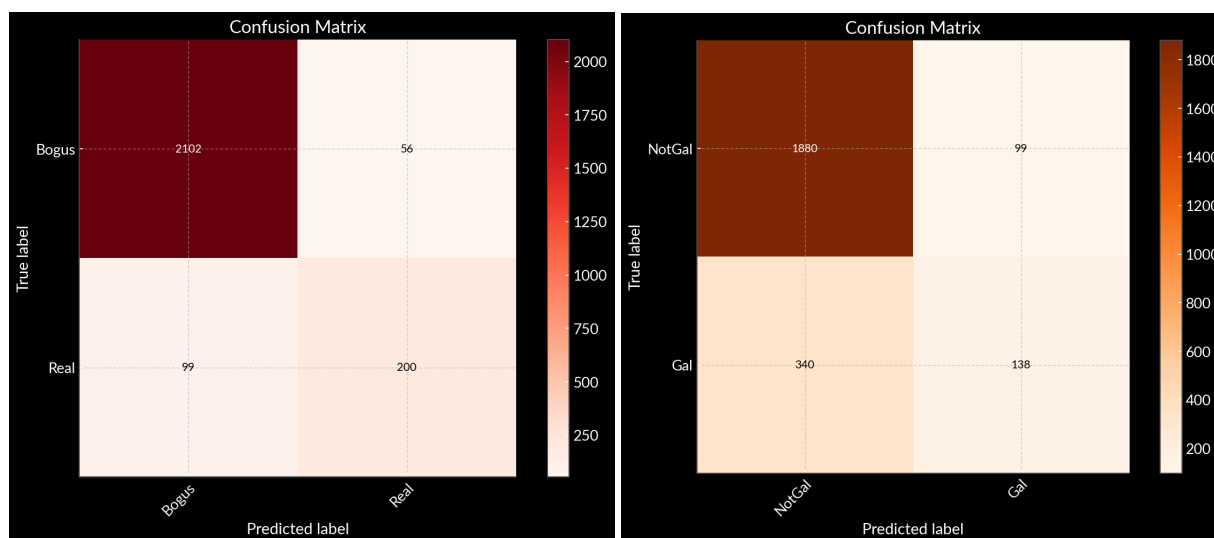
For this we used a **learning rate of 0.1** and **didn't tune any hyperparameters**.

Predictions and performance

Here we only use the **binary predictions**, not the probabilities that the models can assign using `.predict_proba()`.

FUTURE IMPROVEMENTS: How can we add nuance to our performance assessment by using the probabilities instead?

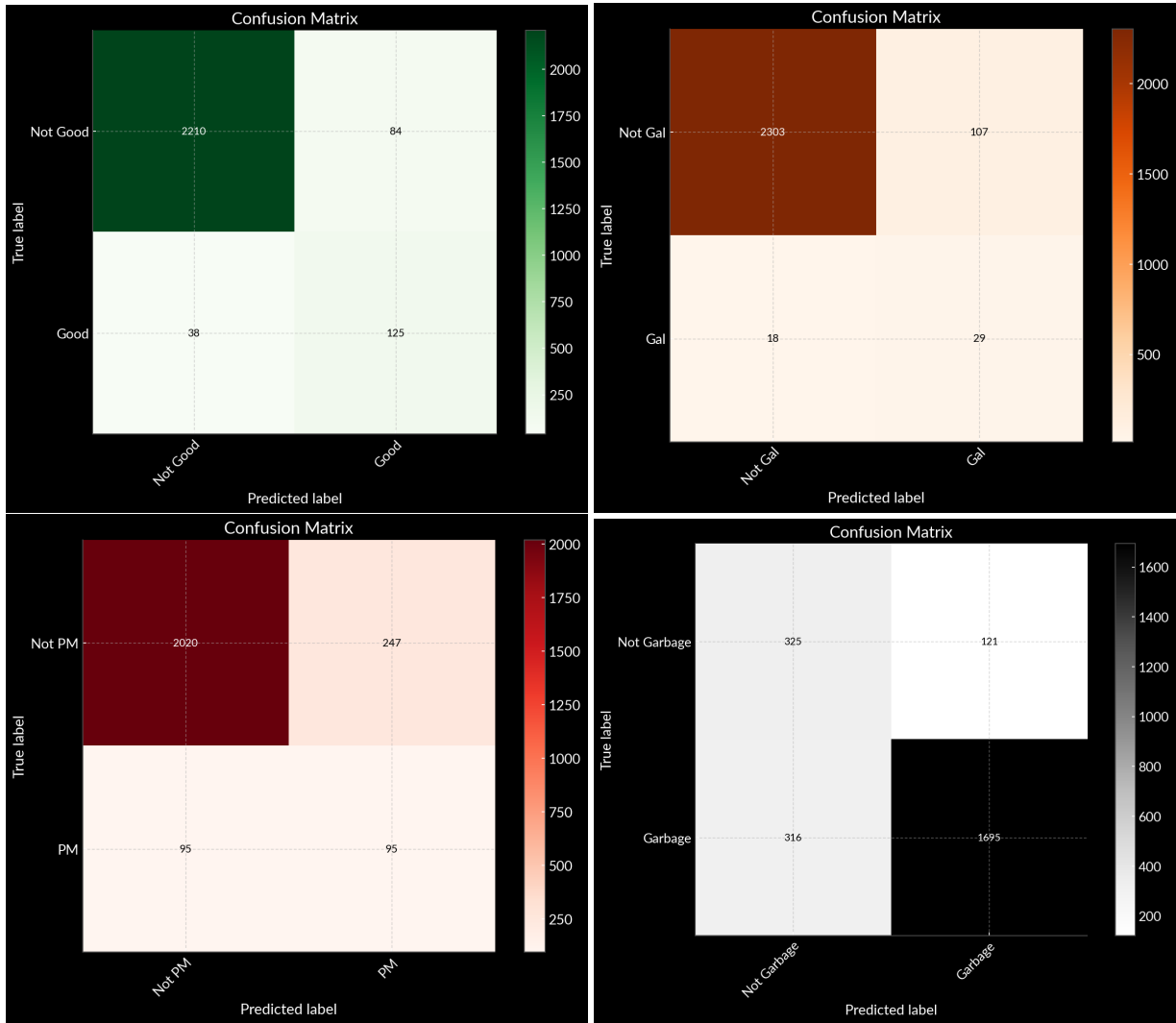
Figure 6.5



In order to see how good a job Arin is doing at "eyeballing" we need to look at the scores together and what they represent for the alert labels. In particular the *good* label is very important as **ideally we don't want to miss any of the good alerts**. Currently the labeling is done with the binary predictions for preal and pgal using the logic table 2.5. **FUTURE IMPROVEMENTS:** How can we use the probabilities assigned to preal and pgal to create more nuanced values of the labels? Do we even need this since we are going to rank according to preal and pgal alone? Is it better to focus on measuring **where the good alerts rank and which ones can be "missed" or ranked low?**

With this model 23% of the good events are misclassified **but that doesn't mean they would be ranked very low - so that metric albeit informative for now, isn't a target to focus on.**

Figure 6.6



6.5 RANKING THE ALERTS

6.5.1 Ranking Algorithm

The ranking of the alerts is currently done through a simple **geometric mean with extra steps**. We rank the alerts in the **Score Space** (x: Real Score, y: Galactic Score) relative to the (0,1) coordinate (Real, Extragalactic) or (R,E). See Figure 6.7. To place each event in that space we use the value of the Real and Galactic scores given by the `predict_proba()` method from our models. **This is why in the scoring models we need to assess the quality of these PROBABILITIES not the absolute CLASSIFICATIONS..**

The real score S_{real} and galactic score S_{galactic} of each event i can be used to calculate their distance D_i to the (R, E) coordinate:

$$D_i = \sqrt{(1 - S_{\text{real}})^2 + (ff \times S_{\text{galactic}})^2} \quad (6.1)$$

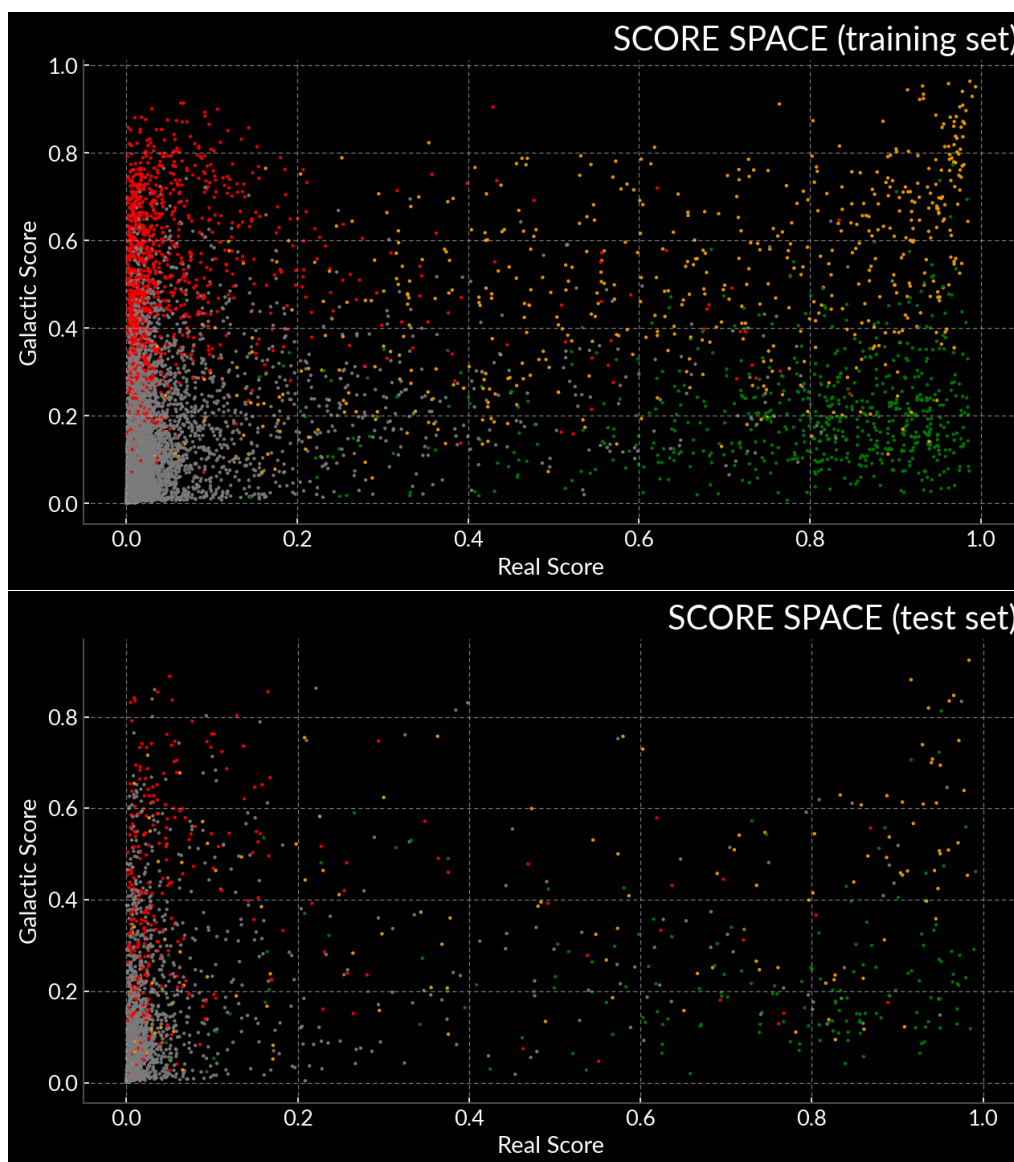
where ff is a fudge factor with value 0.5. This is included because, as can be seen from **Figure add ref**, having a 1:2 ratio of the R:E axes in our Score Space allows for better separation of the events we want to be ranked highest (good and real galactic phenomena). **This is a hyper-parameter that can be tuned later.**

We then calculate the final ranking R_i following:

$$R_i = \frac{(D_{\text{max}} - D_i)}{D_{\text{max}}} \times R_{\text{max}} \quad (6.2)$$

where $D_{\text{max}} = \sqrt{1 + ff^2}$ is the maximum possible distance to the (R,E) coordinate, and R_{max} is the maximum possible score. For now we chose $R_{\text{max}} = 10$. The goal of this step is to give **higher rank** to **more relevant** events,

Figure 6.7



as well as change the range of values that the humans will interact with. We are of the opinion that it will be more intuitive to distinguish between good and poor candidates if the values range from 0 to 10 than if they range from 0 to 1.

6.5.2 Performance in train and test set

In order to evaluate the ranking quality we first focus on the Recall at K ($R@K$) which measures what fraction of the relevant objects is found in the top K elements of the our ordered list.

As can be seen in Figure 6.8, with our test set we recover 100% of our good objects in the top 30% of the ordered list, which suggests eyeballers would only need to consult 30% of the eyeball list.

[WILL HAVE TO SEE HOW THIS CHANGES IN PRODUCTION: can make codes that regularly pull info from the rank table to see what values have been given and how they order (group by similar timestamps of 1 daybins), and once we have labels for all these can calculate the $R@K$ in prod].

6.5.3 Performance in production

Figure 6.8

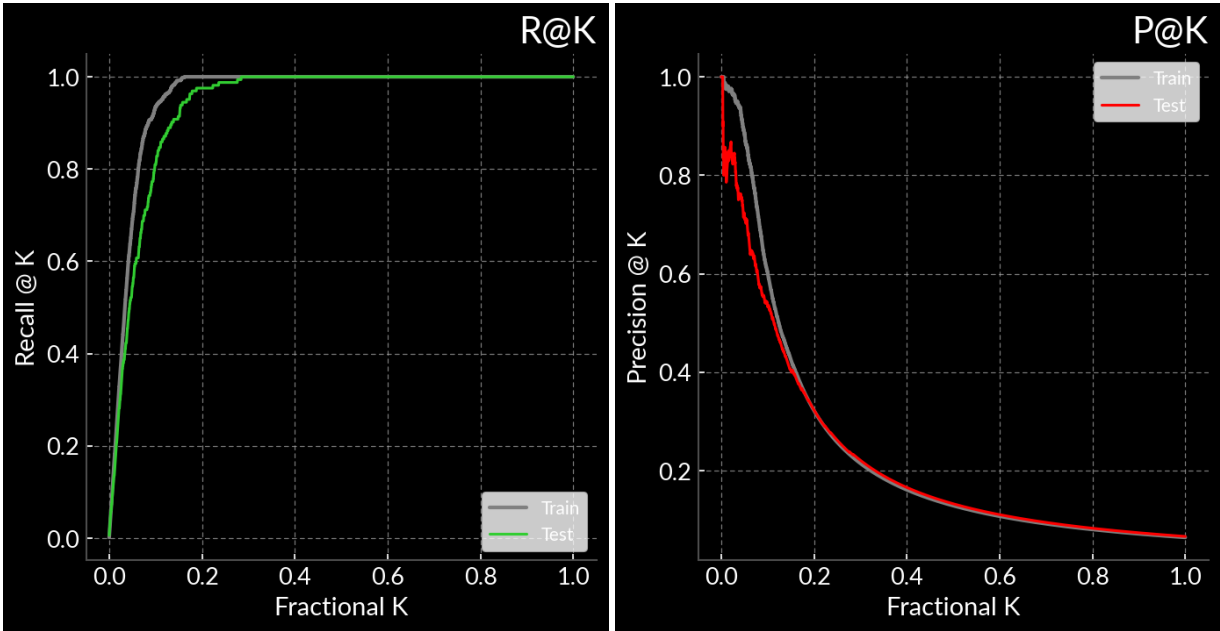
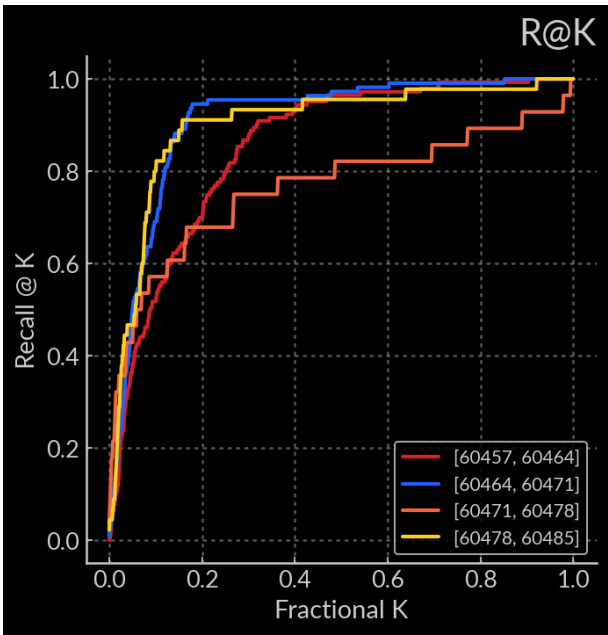


Figure 6.9: Recall at K for the last 4 weeks (1st of July 2024)



Bibliography

- [Smith et al., 2020] Smith, K. W., Smartt, S. J., Young, D. R., Tonry, J. L., Denneau, L., Flewelling, H., Heinze, A. N., Weiland, H. J., Stalder, B., Rest, A., Stubbs, C. W., Anderson, J. P., Chen, T. W., Clark, P., Do, A., Förster, F., Fulton, M., Gillanders, J., McBrien, O. R., O'Neill, D., Srivastav, S., and Wright, D. E. (2020). Design and Operation of the ATLAS Transient Science Server. , 132(1014):085002.
- [Weston et al., 2024] Weston, J. G., Smith, K. W., Smartt, S. J., Tonry, J. L., and Stevance, H. F. (2024). Training a convolutional neural network for real-bogus classification in the ATLAS survey. *RAS Techniques and Instruments*, 3(1):385–399.