# Classification and Data Wrangling on the Amateur Astronomy Frontier

By

Zachary Jacobson

A Capstone Project Paper Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

In

Data Science

University of Wisconsin – Eau Claire, WI

Eau Claire, Wisconsin

December 2023

# ABSTRACT


CLASSIFICTION AND DATA WRANGLING ON THE AMATUER ASTRONOMY FRONTIER

Zachary Jacobson

The website, AstroBin, provides a unique image hosting platform to those involved in the hobby of astrophotography. AstroBin is unprecedented in its focus on collecting and organizing the meta data surrounding the images being uploaded; from the acquisition details, the equipment used to take the image, the post-processing steps performed, even the average phase of the moon which the photos were taken. Having these details organized and structured in a queryable form is an invaluable resource to any astrophotographer at all skill levels.

In this client-based capstone, it was sought to improve upon these meta data collection efforts of the AstroBin website. This was done through two deliverables; an Elasticsearch synonym filter of space objects and their alias names and IDs, as well as an image title text classification model which could be used to provide an automated layer to image classification during a user's image upload process. These deliverables were achieved through four objectives: 1) A comprehensive exploration of text classification modeling techniques and training data augmentations, 2) The thorough review and grading of each modeling technique and training data augmentation explored for a final selection of the deliverable classification model, 3) A process in which to compile space object aliases into the solr formatted synonym filter required by the Elasticsearch search engine, which depended upon 4) A novel data structure to map all space objects to their respective aliases (a Space Object Alias Map).

This project was successfully completed, and all deliverables were provided to the client, in the month of December 2023 as part of Zachary Jacobson's capstone requirement to graduate from the Master of Science in Data Science program through the University of Wisconsin - Eau Claire in Eau Claire, Wisconsin.

## Table of Contents

## List of Tables and Figures

# Chapter 1 – Introduction

## A Client Based Capstone Project with AstroBin

AstroBin is a high-quality image hosting site used by amateur astrophotographers all over the world. It is an open-source project started in 2010 and is currently maintained full-time by its creator, Salvatore Iovene. Salvator's site is unique in its unprecedented efforts made in capturing, organizing, and wrangling the meta data around these "amateur" images. These efforts are shared by both Salvator as the site developer, as well as the community of users uploading their images and entering the information. The images uploaded to the site are often very involved and technical in both acquisition as well as in post-processing. Users of AstroBin are typically quite knowledgeable and consistent in how these technical details are communicated on the site, and the upload process designed by Salvator is helping to standardize the capture of these details as well. All this for the benefit of one of the world's largest, best organized, collections of amateur astro-photographs. This collection also offers advance queries of images by object types, equipment used, and even by moon-phases during which images were taken. All of which are greatly beneficial to any astrophotographer looking to explore and improve their own work. In this capstone two main deliverables were provided to the client website, AstroBin:

1) A text classification model which can automate the categorization of an uploaded image based on the title provided by the user uploading the image.
2) A celestial / space object synonym filter for the website's Elasticsearch search engine.

These deliverables were accomplished by completing the following four objectives:

- Several types of text classification models were developed.
- Each text classification model was compared to each other in extensive detail with visuals to provide the optimal model selection for the client.

- The synonym filter was created using the appropriate file format for the current version of Elasticsearch being used on the website and focused on the “popular” space objects that are imaged from Earth.
- A unique data structure was developed for quick space object name and ID alias referencing, a useful tool that could be used in cleaning and manipulating AstroBin text data, as well as a required resource for the building of the Elasticsearch synonym filter.

**Background of Astrophotography**

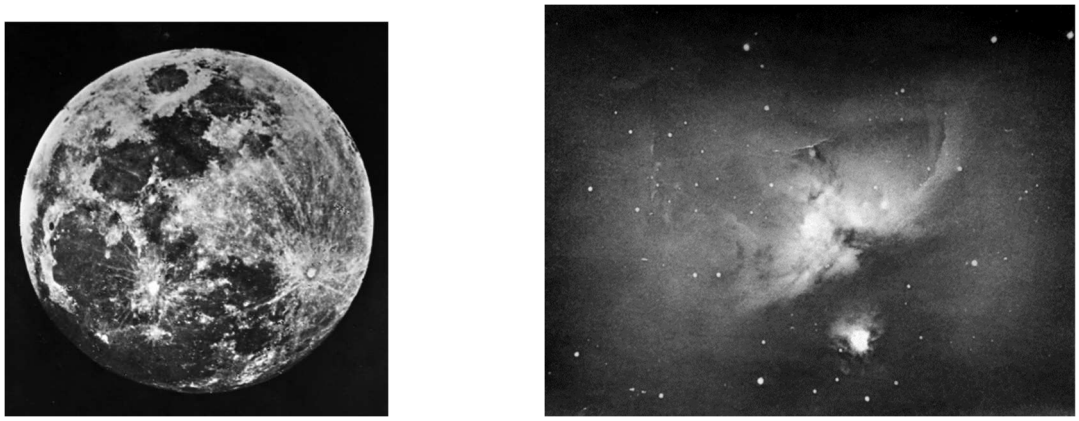
Astrophotography is a specialized type of photography that focuses on capturing images of astronomical objects and celestial events found in the night sky. It involves using a wide variety of equipment to record and create photographic images of stars, planets, galaxies, nebulae, meteor showers, and other celestial phenomena. The primary goal of astrophotography is to produce visually stunning and often scientifically valuable images of objects beyond Earth's atmosphere. The professional and amateur astronomer alike share this activity of astrophotography, and both communities have led the way in its innovation.

The activity of astrophotography is nothing new. The year 1839 is credited as having the first astro-photograph of the moon, 1840 for what is considered the first successfully developed one. Notably, neither were taken by a professional astronomer, but rather first by the French artist and inventor Louis Daguerre and then later by John Draper, an American chemist and medical doctor (M. Lucibella, 2013). So began the era of humans pointing cameras at the sky to record the universe. From then on, improvements continued to be made, again, often by the amateur enthusiasts. Andrew Common, a sanitation engineer, improved the mechanical tracking of his camera on the sky (to counteract the rotation of the Earth and increase the resolution of his

images) to the point where he took one of the earliest high-resolution photos of the Orion Nebula which earned him the Royal Astronomical Society's Gold Medal in 1884.

**Figure 1**

*Early Astrophotography, Late 1800's*



*Note.* From left to right. The Full Moon as taken by J. W. Draper (1840). The Orion Nebula as taken by A. A. Common (1883).

Of course, the professional astronomer has also provided their fair share of innovation. One major example can be found in 1976 where Jim Janesick, a Jet Propulsion Lab engineer, teamed up with the planetary scientist, Brad Smith, to take one of the earliest astronomical images using a charge-coupled device (CCD) imaging chip. Their success quickly led the way to replacing chemical-based photo plates. The amateur astronomy community followed years later but would often opt for a complementary metal-oxide-semiconductor (CMOS) sensor, a more affordable sensor, ubiquitous in all manners of cameras even those in cell phones. Today, both CCD and CMOS sensors are in easy reach of the amateur in regards to both cost and availability.

This mutually benefiting relationship between professional and amateur astronomers advancing the field is not limited to just equipment. A recent notable example of the amateur astrophotographer's role in contributing to the science of astronomy would be the 2022 discovery of the M31 [OIII] emission arc, also known as the Strottner-Dreshler-Sainty Object 1 (shown

below with permission from Macel Drechsler in figure 2). An astounding discovery made by three predominant AstroBin members which surprised professionals and amateurs alike. It spanned a relatively large part of the night sky, nearly four relative lengths of a full moon, and is directly next to M31, the Andromeda Galaxy, arguably one of the most photographed deep space objects out there (over 14,000 images of M31 have been posted on AstroBin alone). So why was this emission nebula not discovered earlier by professional organizations like NASA? A main reason was that this is a large faint nebula in the OIII emission wavelength of light, this meant that the only real way to even know it existed would have been to point a camera with a decent wide field view lens at that region for close to 50 hours using a special OIII light filter, an unlikely thing to do in that region of space for anyone, not to mention the amount of specialized (and intentional) post-processing of the image needed to pull out such faint OIII signal. But leave it to those in the amateur realm, looking to push limits and eke out every bit of signal (the registering of photons from space onto a camera sensor) from of their imaging sessions.

**Figure 2**

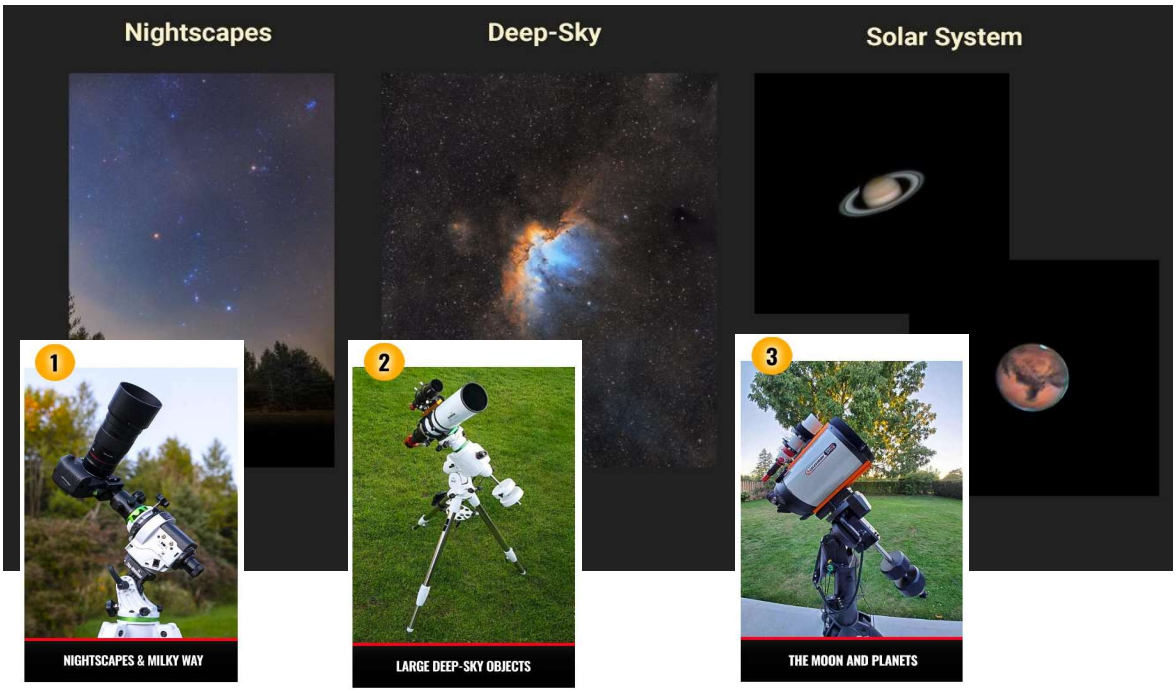
*The M31 OIII Emission Arc - STROTTNER-DRECHSLER-SAINTY OBJECT 1*

*Note:* The 2022 discovery of a large, ionized oxygen nebula (OIII, shown as blue up on the top left in image) in proximity of the Andromeda Galaxy (M31). AstroBin's Image of the Day (IOTD) 01/16/2023, as taken by the collaborative efforts of AstroBin subscribers Marcel Drechsler, Xavier Strottner, Yann Sainty, and others.

Today, a modern astrophotography rig can be as simple as a camera on a tripod pointed at the sky (a very common way to take nightscapes and Milky Way photographs). But often, the average amateur astrophotographer will incorporate many different layers of equipment, all for the benefit of high-quality collections of photons from outer space. The set up and combination of this equipment typically is optimized for a specific type of imaging session. The figure below outlines a few common set ups; images and figures courtesy of AstroBackyard's Trevor Jones and Ashley Northcotte. Each unique set up (of which there are countless variations) can be capable of producing high quality images as long as the astrophotographer understands and works within the limits and strengths of each piece of equipment.

**Figure 3**

*Some Astrophotography Set Up Categories Available to the Amateur Astronomer*



*Note:* This figure shows three examples of various astrophotography set up options. In general, all set ups include some manner of lens / telescope, a camera of some type, and a motorized mount to rotate the lens and camera counter to the rotation of the Earth. There are many variations to those shown above, each with their own purpose and reasons for use. Images and figures used with permission from AstroBackyard's Trevor Jones and Ashley Northcotte.

In addition to equipment choices, the astrophotographer has a variety of acquisition techniques available to them (how long should they set their exposure lengths, what camera settings should they use, should they take the time to take calibration image frames), as well as post-processing techniques (which stacking / pixel averaging methods should be used across the incorporated frames acquired, and how far should the non-linear gamma and color stretches be taken). The complexities come into this hobby from all angles, and it is these complexities which provide both the reward and the headaches to the astrophotographer. With this in mind, it should become apparent to those attempting the hobby, the value in a platform that focuses on the organization and communication of these technical details in each astro-photo image taken and posted online.

**Statement of the Problem**

Outside of AstroBin, the amateur astrophotographer is left with limited resources to share, learn, and improve on their hobby. Obviously, the internet has many options available to socially post and share images in apps like Instagram, but those posting formats are often limited to just the image, who posted it, and whatever other details the user chooses to add into a loose text field, in no standard format. This type of image post does not foster the type of technical information that would be useful to other hobbyist; even if the astro-photo is posted in a dedicated astrophotography group, plus there is still no way to consistently organize image

details like acquisition techniques and what equipment was used across all posted astrophotography images.

If technical help is needed, there are of course online forums; however, these sites are not conducive to sharing images, and the information shared is confined to the limits of the classic "threaded discussion" format. As anyone who has explored forums like this knows, compiling knowledge from these posts is a clumsy way to learn. Posted topics are usually either too pointed or too vague to be useful to those outside of the original conversation and are often rather removed from the actual image result being discussed (if an image was shared at all). AstroBin aims to have both a robust image hosting platform, as well as maintain strong meta data around each image, so as to keep the technical details close to the image and useful to any querying astrophotographer.

**Purpose of the Project**

This capstone project focused on improving the data wrangling efforts within AstroBin. As the hobby of astrophotography continues to grow along with our global interest in space, Salvator's AstroBin will undoubtedly continue to be a valuable platform for us all to learn, share, and gain experience in the hobby. Additionally, there is a plethora of scientific value nested within and across all of the images getting uploaded over the years from all over Earth. Building up the data infrastructures and automated pipelines within the AstroBin site to maintain high quality image meta data will be critical to not only keep pace with increasing site usage, but also to provide paths and opportunities within the scientific community (professional and amateur alike).

## Objectives

The following four objectives were accomplished within this capstone. First, multiple classification models were trained and developed to take in a submitted image title text from a user and return ranked category selections for the image being uploaded. Second, detailed and easily visualized measures and comparisons of each model variation were generated to assist in the selection of a final, optimal classification model for the AstroBin implementation. Thirdly, on a separate track, a synonym filter was created for AstroBin's Elasticsearch search engine. This synonym filter pertains to popular imaged objects and their alias names and IDs across common astronomical catalogs as recognized in image titles and other image associated text on the AstroBin website. Fourthly, a novel data structure was developed to quickly and repeatedly navigate the variety of names, aliases, and IDs applied to a space object, a data structure hence forth dubbed the "Space Object Alias Map" (SOAM). As described throughout the rest of this paper, the SOAM was a required component for the successfully built synonym filter, as well as a potentially useful tool for future text analysis on the AstroBin website.

*Image Title Classification Modeling Objective – Associating Image Titles to an Image Category*

Although the user community uploading images to AstroBin has consistently demonstrated their ability to go the extra mile and manually connect various tags and categories to their image posts, AstroBin has now taken steps to shoulder this responsibility in categorizing and classifying the images being posted into one of seven pre-existing categories; "DEEP_SKY", "SOLAR_SYSTEM", "WIDE_FIELD", "GEAR", "STAR_TRAILS", "NORTHERN_LIGHTS", "NOCTILUCENT_CLOUDS", and "OTHER". Given the user's image title as input, the developed classification model now provides an automatic selection of category that the image is to post under, a selection which can still be altered by the user or the image post approval process

if needed. Although multiple models and training data manipulations / augmentations were explored, the core, bulk data across all iterations of this text classification model utilized the same source data (~600K image titles from AstroBin alongside their respective categories).

*The Image Title Classification Model Selection Objective – Automating Image Categorization*

There are many things to consider when building a text classification model. Firstly, the representation of each category within a given fold of the training data needs to be understood, and if over-fitting is a concern, must absolutely be addressed. The selection of which model algorithm to employ is another critical decision; one that must be made among choices such as the probabilistic Naive Bayes Classifier, a versatile Support Vector Machine (SVM), the bag-of-words technique of a Random Forest, or even the deep learning model option of Convolution Neural Networks (CNNs). This capstone project accomplished an optimal model selection for the client based on the measures and comparisons of each explored model variant developed in the previously explained objective.

*The Search Engine Synonym Filter Objective –Natural Language in Computerized Queries*

Search Engines are ubiquitous tools found in almost all corners of the internet and are critical to a good user experience. They come in various flavors and types, but the basic concept of a search engine is as follows:

1) Crawling - The engine must first “Crawl” through vast volumes of content and identify information.
2) Indexing - The engine will then “Index” the identified information in a searchable format, organizing all manners of key words, headings, links, and other pertinent data.
3) Ranking - At this point the engine is ready for a query. Given a query the engine must then work out a “Rank” of indexes as they pertain to the query.

4) Displaying - Once the engine has the rank sorted out, it must then “Display” the results of the returned query.

AstroBin relies on a search engine known as Elasticsearch along with the Python abstraction layer known as Haystack. Elasticsearch is a distributed, RESTful search engine which is fast, flexible, and scalable. It also allows for a synonym token filter to be applied. In this capstone project, equivalent synonyms (names and ids) were identified and explicitly mapped for popular and commonly photographed astronomical objects and a filter was built for these object synonyms using the Apache Solr indexing format.

*The Space Object Alias Mapping Objective – The Language of Astronomy: Nouns*

There are millions of celestial objects listed across 14,000+ astronomical catalogs. This had presented a unique challenge to the formatting of the Elasticsearch synonym filter. For example, the Orion Nebula (depicted in the left-most image of figure 1) can be referred to by the following names and IDs (aliases) in the astronomy community:

- “Orion Nebula” (a common name)
- “NGC 1976” (the New General Catalogue reference)
- “M 42” (the Messier catalog reference)
- “LBN 974” (the Lynds Catalog of Bright Nebulae reference)
- “Sh2 281” (the Sharpless catalog reference)

… among many other aliases. An added layer of complexity here is that the conventions used to refer to the various catalog IDs may vary between texts. For example, “M 42” could just as easily be written as “M-42”, “m42”, or even “Messier 42”, again, all referring to the Orion Nebula. To address this challenge, a Space Object Alias Map (SOAM) data structure was created to provide quick reference to a space object’s set of alias names and ids.

**Significance of the Project**

Improving the AstroBin website's ability to collect, store, and use the meta data of these images will pave the way for even more powerful features, beneficial to both the user experience as well as the astronomy community as a whole. These features could include, but would not be limited to:

- Recommendation systems that could provide a user a suggested celestial imaging target suited to their equipment, time of year, and their general latitudinal position on the globe.
- Automated detection of proper motions of fast-moving stars.
- Automated detection of variable stars or novae
- Automated detection of movement in nebulae (see https://www.astrobin.com/ija7jc/B/. for an example of this)

All these advanced site functions would rely on more consistent and automated meta data generation and curation. The previously listed service objectives performed under this capstone project are intended to advance these meta data efforts.

**Assumptions, Limitations, and Delimitations**

The primary deliverable in this capstone to the client, AstroBin, is the automated image categorization given a user's image title as input via a select text classification model. This classification model is strictly limited to the title text input and does not rely on any other data or context (the actual image file is ignored during the automated selection of its category). The assumption here being that the title text provides enough context of the associated image file being uploaded to correctly categorize the image into one of the several existing categories. The provided classification model will not be fail-proof, and it will be assumed that the implementation of automated image category selection will have a manual aspect as well. As this

is the primary deliverable, taking up the majority of this capstone, the subsequent literary review section solely focuses on the academic details of text classification; mentions of search engines and Elasticsearch have been made through this paper with supporting references as needed.

Speaking of; the Elasticsearch synonym filter deliverable has been limited to the "popular" celestial objects which are commonly imaged by astrophotographers, as well as limiting the synonyms of those objects to the most common catalogs and common names. These popular objects were assumed and discerned from the provided title text data, where the SOAM was then used to clean and scrape out recognizable celestial objects and names, while also organizing those identified names into their associated alias sets. Those sets were then formatted into the synonym file for use in the synonym filter. While Elasticsearch offers a wide variety of configurations and applications of synonym token filters, this capstone focused solely on the equivalent and explicit formatting of these synonymous alias name sets. Additionally, this synonym filter was fitted for Elasticsearch version 2.4.1. Layers and formats such as WordNet were not pursued at this time.

## Chapter 2 – Literature Review

### Text Classification Modeling

At its most basic level text is any written form of language. In a computerized system, text is most commonly represented as alphabet characters strung together to form words within a language, where each character is associated to a unique sound. Examples of alphabet text can be found in languages such as English and German. It is also important to understand that text can also include written syllabaries (where each single character can represent an entire syllable, like those found in the Japanese language), or text can even include written logograms (where each character can represent an entire word or idea, like those found in the Chinese language, or

emojis in a text message), points which are summarized in the first chapter of Barry DeVille's and Gurpreet Singh Bawa's book, *Text as Data* (2022). The AstroBin platform supports the following alphabet text languages: English, German, French, Italian, and Portuguese, in addition to simplified Chinese. This capstone focused on alphabet character text within the international AstroBin website, ignoring and often cleaning out all other non-alphabet text (including emojis, syllabaries, and logograms, such as the simplified Chinese) as these were relatively low in occurrences within the provided image title data. This allowed for more immediate access to common methods such lemmatization, tokenization, and n-grams, rather than needing to attempt more complex methodology like sub-character tokenization, a method extensively covered by Chenglei Si, et al (2023) in their paper on *Sub-Character Tokenization for Chinese Pretrained Language Models*.

As alluded to above, there are certain data preparation methodologies which should be considered in relation to handling and cleaning the text prior to training any classification models. Tomas Pranckevicius and Virginijus Marcinkevicius (2017) provide a straightforward approach to preparing a corpus of product review text from Amazon in their paper, *Comparison of Naïve Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification*. The four main steps in their general workflow were as follows:

1) Data extraction and addressing skew by taking an equal number of review text from each review rating category.
2) Preparing each review text by first tokenizing each word in the text, followed by the removal of all stop words, standardizing the text into lowercase letters, and stemming the

words with Porter stemmer to their base/root forms (more on stemming and the Porter stemmer later).

3) Bagging of all *n*-grams (unigrams, bigrams, or trigrams, etc.) from the continuous text flow (without any meaningful sentence or paragraph structure, possibly even without any meaningful word structure depending on the technique) of each review text as prepared in the previous steps. A hashing term-frequency vectorizer is then established, further transforming the text reviews into numerical representations of word frequencies.
4) Selecting a choice classification method and using 10-fold cross-validation, train and test the model.

All steps described above are generalized and can always be adapted to better fit the classification task at hand. HaCohen-Kerner, Miller, and Yigal (2020) show in their paper, *The influence of preprocessing on text classification using a bag-of-words representation,* that the removal of all stop words (the high frequency words that carry little meaning on their own) is one of the most effective single pre-processing steps that can be done to improve a model's accuracy; that is unless the text data is sparse, in which case they found that removing stop words actually harmed the accuracy of the model. The AstroBin image titles are rather small, on average four to five words in length.

Depending on the characteristics of the corpus of text available to the problem space, the choice of employing lemmatization, stemming, or some other word standardization technique is important. For instance, as Polus and Abbas elaborate on in their paper, *Development for performance of Porter stemmer algorithm* (2021), the Porter stemmer is very efficient and well suited for very large documents; however, there are drawbacks where the removal of prefix or suffix may happen irregularly (depending on the commonality of the word within the language)

and will often create awkward word-cores that have lost meaning and no longer get picked up properly in word frequencies or sentiment analysis (like “universe” getting stemmed to “univers”, or “the adventurous cat” getting stemmed to “the adventur cat”). Regardless of these issues and not being a great option for sparce text data, stemming is still considered to be a valid option when the corpus is large and computational resources are limited, frequently providing better performance and results in those situations (Balakrishnan and Ethel, 2014; Polus and Abbas, 2021). Likewise, lemmatization will also standardize words from the text data, but instead a computationally efficient trim of the word, lemmatization will convert the word into its recognized root. This requires additional computational resources as well as a strong understanding of the language(s) being processed. This trade off with lemmatization can often produce better results where ambiguous word forms exist in the context of certain word sets (Vatri and McGillivray 2020; Vatri and McGillivray 2020). Depending on the text data being processed it is also very possible that stemming or lemmatization could actually harm results, as both methods are prone to remove key word traits important to the analysis. Again, as HaCohen-Kerner, et al. (2020) allude to in their research, avoiding stemming and lemmatization should especially be considered when working with sparce and limited text data.

In addition to the pre-processing method choices one can make, the model algorithm chosen will also impact the success of accurate classifications. Referring back to Pranckevicius and Marcinkevicius (2017), the primary outcome of their study revealed a surprising improvement in multi-class classification accuracy when using a logistic regression classification method as opposed to other classification methods such as Naïve Bayes and Support Vector Machine; however, it should be noted that despite their claim, logistic regression also showed the highest level of variability among all tested methods across all tested data sets, where accuracy

trended down as the size of the data set increased. For the purposes of this capstone, four types of classification modeling methods were explored: Multinomial Naive Bayes, Random Forest Classifier, Linear Support Vector Classifier, and One-vs-All Logistic Regression.

*Multinomial Naive Bayes Classifiers*

Regarding Multinomial Naive Bayes classification, C.D. Manning, P. Raghavan and H. Schuetze walk through a clear description of this method in their book, *Introduction to Information Retrieval* (2008). The generalized equation they provide is as follows …

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c) \tag{1}$$

… the intent being to determine $P(c|d)$; the probability of document, *d,* being part of the category, *c*, where $P(t_k|c)$ is the conditional probability of term $t_k$ occurring in the specific document category, *c*, and $P(c)$ is the prior probability of *d* occurring in *c.* This generalized description does not necessarily distinguish the Multinomial Naive Bayes classification method from others, so Manning, et al. (2008) go on to further describe the maximization equation responsible for the actual classification selection, the *maximum a posteriori* (MAP) class, where the highest $c_{map}$ classification is the selected category, *c* …

$$c_{map} = \arg\max_{c \in \mathbb{C}} [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]. \tag{2}$$

To elaborate, the log of the prior predicted $log\hat{P}(c)$ from the training data is the weighted value reflecting the relative frequency of category, *c*, as it occurs across the entire corpus of training data (infrequent *c* categories are less likely to be the correct, max, category to select). Additionally, all conditional parameters $log\hat{P}(t_k|c)$ act as weighted indicators of how well the term, $t_k$, fits the category, *c*. Although this makes strong (and often unrealistic) assumption that

all occurrences of terms (words, in the case of this capstone) are independent of all other terms, this simple probabilistic approach is computationally efficient and well suited for text classification problems which require speed on top of limited computational resources.

*Random Forest Classifiers*

Leo Breiman (2001) defines a random forest as a "classifier consisting of a collection of tree-structured classifiers $\{h(x, \Theta_k), k = 1, \ldots\}$ where the $\{\Theta_k\}$ are independent, identically distributed, random vectors and each tree casts a unit vote for the most popular class at input x". In contrast to the probabilistic Multinomial Naive Bayes classification method, the Random Forest classifier is comprised of multiple, uniquely generated decision trees that are built from bootstrapped samplings of the training data. This ensemble learning method builds and merges all decision trees and their predictions to provide a more robust and accurate result than individual trees ever could. Although this method can often outperform its counter parts with fewer computational resources, Leo Breiman (2001) also recognizes the "black box" issue of Random Forest classifiers, stating, "A forest of trees is impenetrable as far as simple interpretations of its mechanism go". Depending on one's application of classification modeling, this could be a deal breaker. In the case of image title classification on the AstroBin website, interpreting how the model made the decision of classifying an image as, say, a 'NORTHERN_LIGHT' subject category over a 'SOLAR_SYSTEM' subject category is not critical to the intended application of the model, making Random Forest a viable classification method for this capstone.

*One-vs-All Logistic Regression Classifiers*

Another classification method option would be logistic regression; however, this method on its own is only applicable to binary decision making. AstroBin has more than two image

subject types available for category selection, so how can a binary decision-making algorithm be of use here? One recent example of using logistic regression in a multi-class categorization problem can be found from Amjoud's and Amrouch's paper, *Transfer Learning for Automatic Image Orientation Detection Using Deep Learning and Logistic Regression* (2022). They used logistic regression in their deep learning workflow to recognize and categorize whether an image is rotated 0, 90, 180, or 270 degrees using a "one-vs-rest" (OVR) approach. Their binary linear classifier for each degree category was built on the following optimization:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} log(e^{-y_i(X_i{}^T w+c)} + 1) \tag{3}$$

Where training samples are represented by $X_i$, and $y_i$ represent their corresponding class label. The hyperplane $w$ is the optimum of negative and positive linear spacing, and parameter $C$ sets the tradeoff between the penalty of errors, $\epsilon$, and the $w^t w$ regularization term.

$$\epsilon = log(e^{-y_i(X_i{}^T w+c)} + 1) \tag{4}$$

Amjoud and Amrouch then proceeded to use the quasi-Newton method algorithm, Limited-memory Broyden Fletcher Goldfarb Shanno(L-BFGH), to iteratively solve the optimization problem where then an independent model was trained for each class to ensure each OVR classifier was trained separately for all the categories. This approach may seem similar to Random Forest classification; however, it should be noted that OVR Logistic Regression typically will not allow for multiple models to be built in parallel, as each iteration will have one category set to a "positive / on" value and all other categories set to "negative / off" value. Whereas a Random Forest classifier can generate its individual trees in parallel to then merge all trees at an end step into a final result. OVR Logistic Regression also contrasts with Multinomial Naive Bayes classifiers, as OVR Logistic Regression does not assume independence between

features. This makes OVR Logistic Regression a versatile option worth considering in a text classification problem.

*Support Vector Machine Classifiers*

Quite often, though, the method of choice when it comes to text classification is some variation of the Support Vector Machine (SVM). Joachims is frequently credited to having first applied an SVM to a text classification application. As thoroughly described in his 1998 paper, *Text categorization with support vector machines: Learning with many relevant features*, SVMs are ideal for use in the text classification space for several reasons; one being SVMs can circumvent the curse of dimensionality, and a second being SVMs having minimal need for parameter tuning. A major contributing factor for both of these characteristics of the SVM method would be the kernel, which is used to implicitly map relationships between the high dimensional text features.

## Chapter 3 – Methods

Please note, in addition to the descriptions of methods provided below in this paper, all Jupyter Notebooks, Python scripts, and data sets used can be found at the GitHub link referenced in the Appendix at the end of this paper.

### Classification Modeling on Image Title Text and AstroBin Categories

*Data Acquisition, Cleaning, and Augmentation*

The core training data for the image title classification deliverable contains over 600,000 image titles from the AstroBin website along with the corresponding subject type categories of those titles. An outdated category labeled '600' was transformed into the 'OTHER' category, and all null titles and subject types were removed from the data set prior to processing. Next the title text was cleaned by the following cleaning method:

**Text Cleaning Method**

1) Standardize on lower case.
2) Put a space between each alpha and numeric ("abc123" becomes "abc 123").
3) Replace all punctuation with a single white space (' '), except for the apostrophe " ' ".
4) Trim all leading and trailing white space.
5) Replace all multiple/repeated white spaces with single space.

*Original test string:* "  #][!,@ ^&*NGc224-.99+9ab's.  ...  "

*Cleaned test string:* "ngc 224 99 9 ab's"

It is important to note that this text cleaning method was applied to all incorporated text data mentioned here on in, not just the title text. It should also be noted that special, non-English characters like 'æ', 'ø', and '色' remained in all text data, as AstroBin is an international community and these characters should be considered as signal (not noise) in the model training process, even though these characters and their corresponding words/sentences have a relatively low occurrence in the original title text data.

Another issue needing to be addressed was the severe over representation of subject type 'DEEP_SKY' titles, and the general skew across all categories. See table 1 below:

**Table 1**

*Image Title Counts in Provided AstroBin Data Across the Existing Subject Type Categories*

| Subject Type | Counts (N) | Percentage (%) |
|---|---|---|
| DEEP_SKY | 444255 | 75.13 |
| SOLAR_SYSTEM | 111233 | 18.81 |
| WIDE_FIELD | 20108 | 3.40 |
| OTHER | 9670 | 1.64 |
| GEAR | 3587 | 0.61 |
| STAR_TRAILS | 1628 | 0.28 |

| | | |
|---|---|---|
| NORTHERN_LIGHTS | 674 | 0.11 |
| NOCTILUCENT_CLOUDS | 162 | 0.03 |

To address this skew, a smoothing function was written in python to set new count limits for each subject type. This function would then either sample down or sample up titles within a subject type category to meet the new limits corresponding to the given subject type category. This data augmentation method can be summarized as follows:

**Data Augmentation Method**

1) Collect a table of all subject types and their original title counts (see table 1 above).
2) Given a smoothing factor, provide new target counts of titles for each subject type.

$$smoothing\_factor * (sum(Count) / len(Count)) + (1 - smoothing\_factor) * Count[i] = Target\_Count \quad (5)$$

WHERE:

*smoothing_factor is a number 0 – 1, 0 being no smoothing.*

*Count[i] is the Count number for Subject[i]*

*(sum(Count) / len(Count)) is a central value of all Count numbers*

3) If a given subject type's title count exceeded the new target count, those titles within the given subject type's category were down sampled to the new target count for that subject type through random selection.
4) However, if a given subject type's title count fell short of the new target count, the following options were available to up sample the titles for the given subject type category:
    a. Original subject type title data from AstroBin were always included, in full.
    b. The remaining title data needed to reach the new target count was portioned out equally amongst the following augmentation options if the supplemental data was made available. These augmentations were varying blends and duplications of the original AstroBin title data for the given subject type:

i. Straight up, raw re-sampling (duplications) of the original data

ii. Original data blended with public online wiki text, OPT web shopping catalogue data, and ChatGPT 3.5 generated text.

iii. Original data altered with a synonym replacement function.

… it should be noted that all supplement text data: wiki text, shopping catalogue text, and ChatGPT generated text; were cleaned with the standard text cleaning method described above, as well as had all English stop words removed before being blended with the original title text data. Stop words were removed in an attempt to reduce false signal within heavily augmented title text.

5) The smoothed, augmented title data was then returned for use in the downstream model training method.

**Figure 4**

*Example of smoothing factor's impact on the data distributions: (Sample Size (n) = 10,000 titles)*

Sampled Data --- smoothing factor = 0.0

Table of subject_type Counts:

| | Subject | Count | Percentages |
|---|---|---|---|
| 0 | DEEP_SKY | 7511 | 0.7511 |
| 1 | SOLAR_SYSTEM | 1874 | 0.1874 |
| 2 | WIDE_FIELD | 356 | 0.0356 |
| 3 | OTHER | 155 | 0.0155 |
| 4 | GEAR | 63 | 0.0063 |
| 5 | STAR_TRAILS | 33 | 0.0033 |
| 6 | NORTHERN_LIGHTS | 7 | 0.0007 |
| 7 | NOCTILUCENT_CLOUDS | 1 | 0.0001 |

TOTAL = 10000

Augmented Data --- smoothing factor = 0.3

Table of subject_type Counts:

| | Subject | Count | Percentages |
|---|---|---|---|
| 0 | DEEP_SKY | 5633 | 0.5633 |
| 1 | SOLAR_SYSTEM | 1687 | 0.1687 |
| 2 | WIDE_FIELD | 624 | 0.0624 |
| 3 | OTHER | 483 | 0.0483 |
| 4 | GEAR | 420 | 0.0420 |
| 5 | STAR_TRAILS | 397 | 0.0397 |
| 6 | NORTHERN_LIGHTS | 379 | 0.0379 |
| 7 | NOCTILUCENT_CLOUDS | 377 | 0.0377 |

TOTAL = 10000

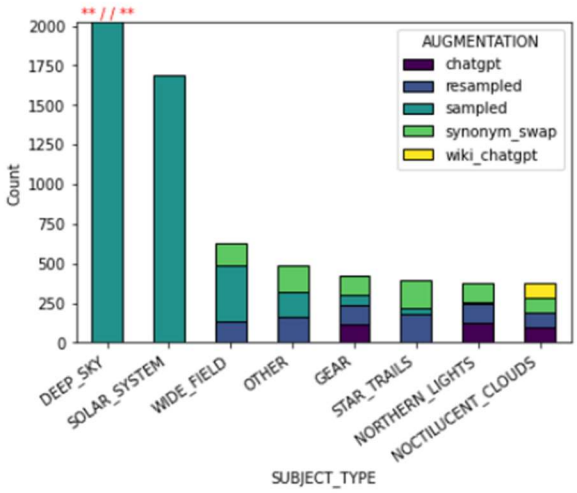
**Figure 5**

*Original Sample Data Percentages:*

*(percent of original data from AstroBin, after smoothing factor 0.3)*

| subject_type | Original Data (%) |
|---|---|
| DEEP_SKY | 100.00 |
| SOLAR_SYSTEM | 100.00 |
| WIDE_FIELD | 57.05 |
| OTHER | 32.09 |
| GEAR | 15.00 |
| STAR_TRAILS | 8.31 |
| NORTHERN_LIGHTS | 1.85 |

Augmentation Visual




*Iterating Through Model Variations and Exploring Optimums*

The following approach was taken to assess the best level of training data augmentation along with the best classification model configurations:

**Training Data Augmentation and Model Configuration Approach for Determining Optimum:**

1) Each of the following steps were performed at three different scales: small (sample size (n) = 5,000 titles), medium (n = 20,000 titles), and large (n = 100,000 titles); where each scaled sampling was pulled from the cleaned title text data described in the Data Acquisition, Cleaning, and Augmentation section.
2) During each of the 3 scaled iterations, sub-iterations of smoothing factors were performed, creating unique data sets from the sampled title text according to the iterated smoothing factor using the Data Augmentation Method described above. Both the small and medium scale iterations sub-iterated through 11 different variations of the sampled data, augmented by scaling

factors 0.0 through 1.0. The large scale iteration, sub-iterated through 6 variations of sample data, augmented by scaling factors 0.1 through 0.6.

3) At each of the sampled data augmentation iterations, multiple model types were generated, each model type being repeated across, and using several different Term Frequency - Inverse Document Frequency (TFIDF) Vectorizers.
    *a.* At small scale, 4 model types were generated (Multinomial Naive Bayes, Random Forest Classifier, Linear Support Vector Classifier, and One-vs-All Logistic Regression) each repeated across 12 different TFIDF Vectorizers. The small-scale title sampling iteration across each of its 11 smoothing factor augmentation sub-iterations resulted in a total of 528 different models to grade and assess.
    *b.* At medium scale, the same 4 model types were generated, but were instead only repeated across 6 different TFIDF Vectorizers. The medium scale title sampling iteration across each of its 11 smoothing factor augmentation sub-iterations resulted in a total of 264 different models to grade and assess.
    *c.* At large scale, only 2 model types were generated and assessed (Random Forest Classifier and Linear Support Vector Classifier), across only 3 different TFIDF Vectorizers. The large-scale title sampling iteration across each of its 6 smoothing factor augmentation sub-iterations resulted in a total of 36 different models to grade and assess.
4) Within each scaled iteration, each generated model was then graded using the following scoring method:
    a. First, using sklearn's "MinMaxScaler", the following model measurements were standardized into a number 0-1 across all models within the current iteration. These measurements were 'accuracy', 'f1 score', 'precision macro average', and 'original data percentage'.

**Figure 6**

*The sklearn.preprocessing.MinMaxScaler Python Function:*

The transformation is given by:

```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min
```

where min, max = feature_range.

Note: *https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html*

b. Second, the grades for each of the models within the current iteration were determined by using the following partitioned weighted scoring:

$$\textbf{Partitioned Weighted Score} = \frac{1}{3} * \frac{A+F1}{2} + \frac{1}{3} * P_{macro\_avg} + \frac{1}{3} * O \qquad (6)$$

WHERE:

1/3 is the Weight of each term (partition).

(A+F1)/2 is the mid-point between model Accuracy and F1 score.

P_(macro_avg) is the model's macro average Precision.

O is the amount of original (not augmented) data from AstroBin used to train the model.

5) At the final large-scale iteration, out of the 36 generated models, the top 5 models with the highest grades were reviewed and the following considerations were made accounting for the results of each scaled iteration:

a. Word vectorizers tend to slightly outperform character vectorizers. The final model should use a word vectorizer.

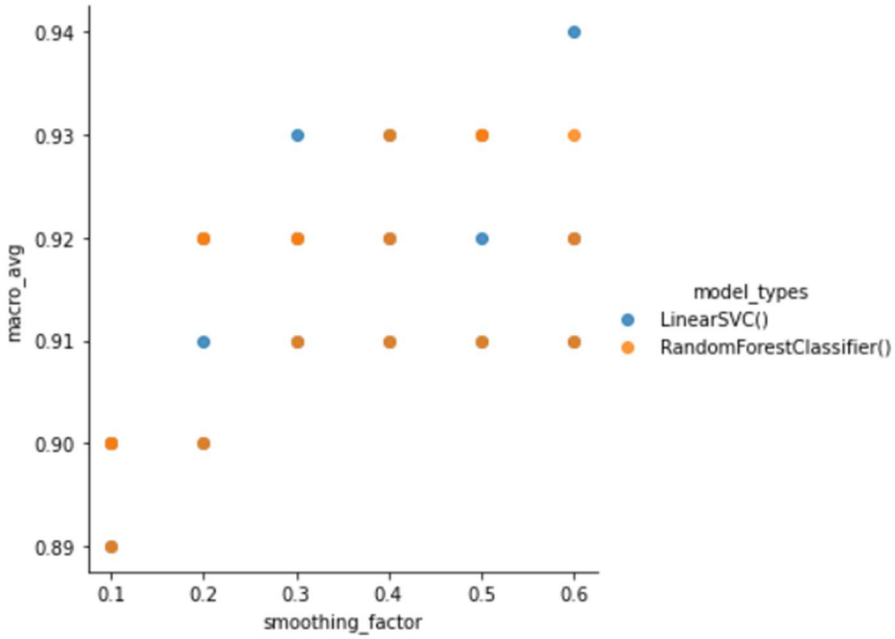
b. Multinomial Naive Bayes models tend to perform worst on average. The final model should not utilize the Multinomial Naive Bayes method.

c. The top two performing models are Linear Support Vector Classifiers and Random Forest Classifiers. Both outperform One-vs-All Logistic Regression. The final model should not utilize the One-vs-All Logistic Regression method.

d. A lower smoothing factor is preferred, so as to maintain a higher percentage of original / un-augmented AstroBin data for training. Increasing the augmentation smoothing factor beyond 0.5 may run the risk of over fitting on synthetic signal. The final model should not augment data with a smoothing factor higher than 0.5.

**Figure 7**

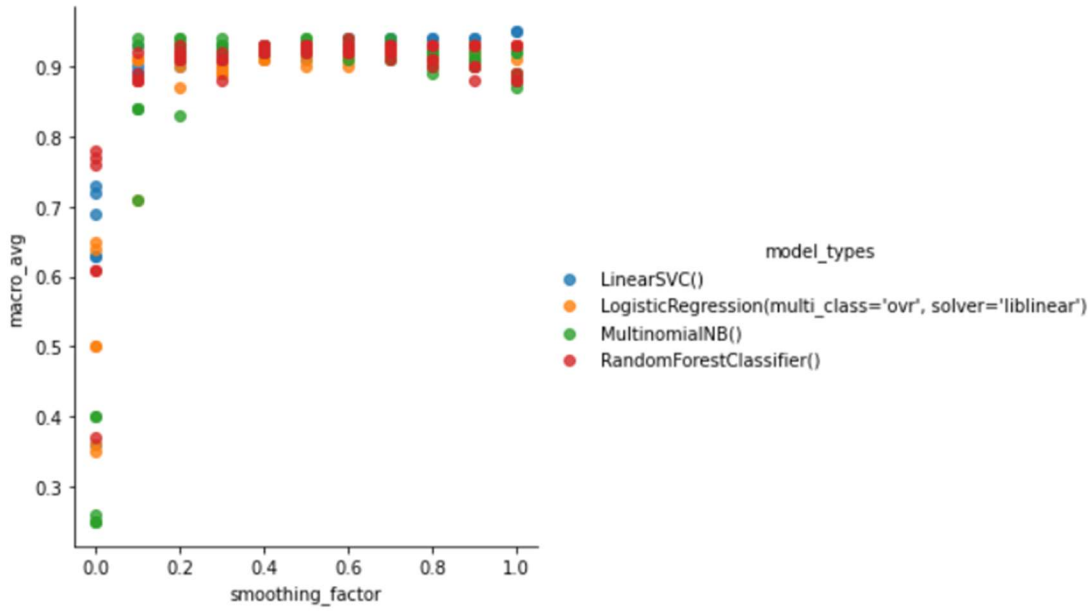
*Concerns of over fitting on synthetic augment data when using a large smoothing factor:*




e. Without some augmented data to smooth over the data skew, precision across the subject type categories would suffer. The smoothing factor should be greater than 0.0 (it does not take much smoothing to get major improvements in the precision macro average).

**Figure 8**

*Major improvements in a model's precision macro average after only minimal smoothing factor increases in data augmentation across all model types:*




f. The Support Vector Machine combined with a TFIDF Vectorization of titles into both 1- and 2-word programing appears to be top performing when using a smoothing factor of 0.3.

**Table 2**

*Top 5 Models with the Highest Grade in the Large-Scale Iteration:*

| model_grade | accuracy | f1_score | macro_avg | smoothing_factor | model_types | vectorizer_types |
|---|---|---|---|---|---|---|
| 0.255597 | 0.927400 | 0.926846 | 0.94 | 0.6 | LinearSVC() | TfidfVectorizer(ngram_range=(1, 2)) |
| 0.240668 | 0.929350 | 0.927481 | 0.93 | 0.3 | LinearSVC() | TfidfVectorizer(ngram_range=(1, 2)) |
| 0.237023 | 0.928154 | 0.926832 | 0.93 | 0.4 | LinearSVC() | TfidfVectorizer(ngram_range=(1, 2)) |
| 0.236963 | 0.929500 | 0.928809 | 0.93 | 0.5 | LinearSVC() | TfidfVectorizer(ngram_range=(1, 2)) |
| 0.227007 | 0.919204 | 0.917280 | 0.93 | 0.4 | RandomForestClassifier() | TfidfVectorizer(ngram_range=(1, 2)) |

*Final Modeling Methods and Tuning*

The optimal model configuration was identified in the previous steps as a Linear Support Vector Classifier with a TFIDF Vectorization word n-gram range of (1, 2) and a data augmentation smoothing factor of 0.3. The final model configuration was then tuned further in a 10-fold cross validation, sklearn pipeline, where each model was scored on balanced accuracy.

$$\textbf{Balanced Accuracy} = \frac{\text{Sensitivity} + \text{Specificity}}{2} \tag{7}$$

WHERE:

Sensitivity = (True Positives) / (True Positives + False Negatives)

Sensitivity = (True Negatives) / (True Negatives + False Positives)

The cross-validation grid search was set up across the regularization term 'C', where values 0.1, 1, 10, and 100 were tried. Note that a small ‘C’ parameter value could lead to underfitting by penalizing large coefficients, while a large ‘C’ value could lead to overfitting due to low regularization / minimal generalization, making the model unable to accommodate new data. Prior to full scale model training, small scale tuning runs pointed to a parameter ‘C’ of 1 or 10 to be best performing. Small scale tuning runs also showed appropriate model performance with the convergence tolerance parameter ‘tol’ set all the way up to 0.1. A larger convergence tolerance value will allow the model to converge more quickly at some expense to accuracy. The final best model returned at full scale had a regularization term, 'C', of 1.0, with a tolerance value, ‘tol’, set to 0.1.

**Space Object Alias Map (SOAM) and Wrangling an Elasticsearch Synonym Filter**

Before moving on to discuss the results of the classification model, a few comments on the SOAM and the Elasticsearch synonym filter creation methods and data wrangling efforts should be made. As noted above, the end result here was the Elasticsearch synonym filter

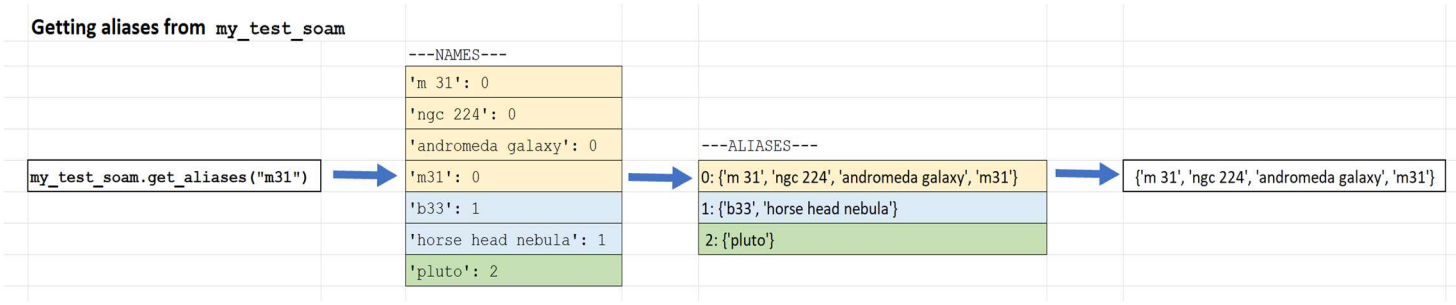
deliverable, where the file created for the client was simply rows of names and IDs of space objects. The challenge in this deliverable was in ensuring each individual row of names and IDs in the file represented a single space object unique from all other rows. To quick review the problem, an example would be the commonly named “Crab Nebula” space object which also goes by the references / aliases of “M 1”, “NGC 1952”, and “Taurus A”. Having multiple names and descriptions available for the same space object can present a unique challenge to a text-based search engine. An additional challenge on the data wrangling side would be that professional catalogues do not typically cross reference other catalogues, and it is rare to consistently have a “common name” associated with a space object catalogue ID. The following subsections describe the data wrangling efforts and the challenges faced in this capstone to deliver this conceptually simply synonym filter.

*The Soam Python Class*

Prior to creating the Elasticsearch synonym filter, names and IDs were organized into a Space Object Alias Map (SOAM) data structure with the help of a self-made Python class called Soam. The Soam Python class relied on two dictionaries; where one dictionary (the “aliases” dictionary) had all associated alias values in their own separate lists and keyed with a simple integer index, and the other dictionary (the “names” dictionary) had each individual name or ID as the dictionary key matched to the integer value of the integer index from the “aliases” dictionary. These two dictionaries together comprise the SOAM data structure.

**Figure 9**

*A Simple SOAM Example:*




The Soam Python class had a rather extensive set of internal helper methods, but the primary method used to manage these two dictionaries would be the ‘add_associations’ method. This method took a list of association sets; in example ‘[{"a", "b", "c"}, {"c", "d"}, {"e", "f"}]’ where 'a' is associated with 'b' and 'c' as well as 'd but not 'f' nor 'e'. With the provided list of associations, each association set (i.e. {“c”, “d”}) would be compared to each existing alias set in the aliases dictionary values that were set prior. Any alias set values which shared a common item in the association set will then get counted as a match. All matches were then merged as a single alias set in the aliases dictionary, and then the association set would be merged into the alias set as well to incorporate any new names or IDs that were brought in with the other shared item(s). Once all associated sets were incorporated into the aliases dictionary, the names dictionary was re-written, so each name / ID key would again be matched to its correct index as it pertained to the new aliases dictionary. A few notes regarding what was just mentioned here; the "a", "b", "c" example was just an example, all items in the provided associations list that were one character or less were filtered out and not added to the SOAM. Additionally, the Soam class requires a cleaning method to get passed in, so as to standardize all name and ID text prior to

storing in the SOAM. The cleaning method used in this part of the capstone was the same cleaning method described above in the classification modeling method subsection.

*Data Wrangling: Acquisition, Cleaning, and Loading*

The first task in creating a SOAM data structure with the Soam Python class was to wrangle sets of associated names and IDs of space objects; these initial sets were called "seed associations" as they were the base foundation of the alias mapping. There were several sources of data used to corral these associations. Wikipedia proved to be a valuable source along with several other public web pages where the following older catalog IDs were scraped in full alongside various other references:

**Table 3**

*Astronomical Name and ID Data Wrangling Sources (Seed Associations)*

| Web Source | Catalog / Collection Description | Number of Space Objects Pulled | Notes |
|---|---|---|---|
| https://en.wikipedia.org/wiki/Messier_object | Messier (M) | 110 | All IDs were associated with NGC number and the occasional common name. |
| https://en.wikipedia.org/wiki/Caldwell_catalogue | Caldwell (C) | 109 | All IDs were associated with NGC number and the occasional common name. |
| https://planetarynames.wr.usgs.gov/Page/Planets | Planets, Moons, and Minor Planets | 194 | Just names |
| https://pacrowther.staff.shef.ac.uk/WRcat/ | Galactic Wolf Rayet (WR) Stars | 669 | All IDs were associated with HG number along with, up to, 4 other aliases |
| https://en.wikipedia.org/wiki/Sharpless_catalog | Sharpless (Sh2) | 38 | All IDs were sporadically associated with other catalog IDs and common names. |
| https://en.wikipedia.org/wiki/Gum_catalog | Gum (Gum) | 13 | All IDs were sporadically associated with other catalog IDs and common names. |

Along with these seed associations, the SIMBAD database (the Set of Identifications, Measurements, and Bibliographies for Astronomical Data) was also incorporated into the SOAM. SIMBAD contains information for about 13,000,000 astronomical objects (stars, galaxies, planetary nebulae, clusters, novae and supernovae) but does not contain any information regarding solar system objects like planets or moons. To query SIMBAD, three other catalogs were used: the New General Catalog (NGC), the Index Catalog (IC), and the Henery Draper (HD) catalog. There were two main steps to incorporating SIMBAD into the SOAM data structure, each step with its own unique challenges:

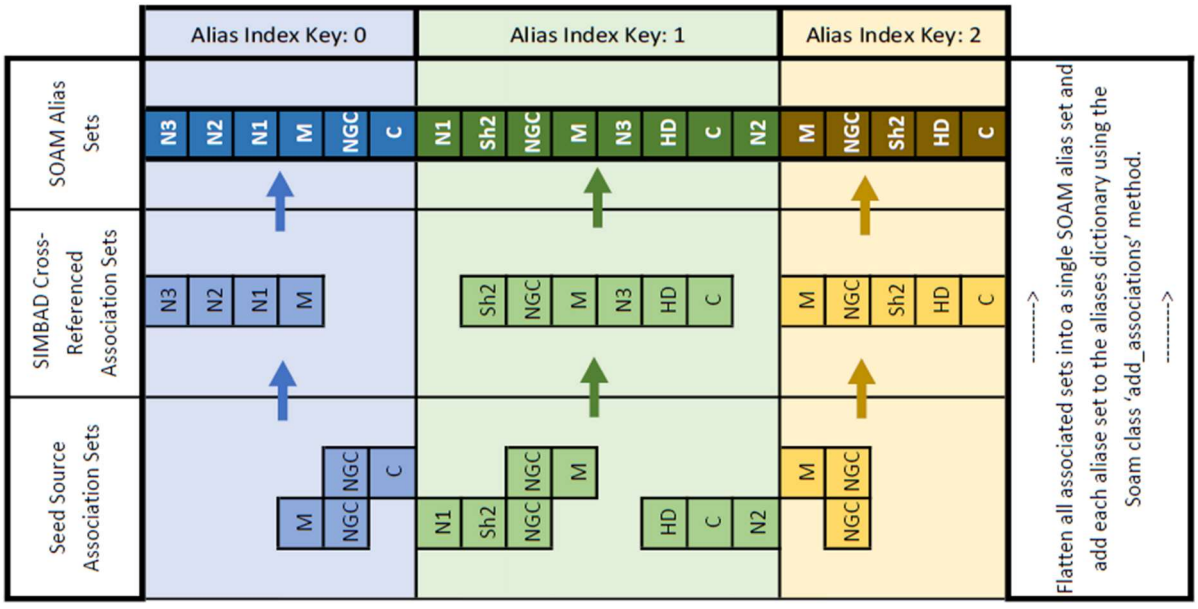
1) The first step was to filter each catalog down to just space objects which pertained to AstroBin. Each of the three catalogs (NGC, IC, and HD) contain thousands of space object IDs, an impractical number to incorporate without greater computational resources, where, additionally, only a subset of each of these objects are likely to be visible in any detail from earth making the rest of the objects from these catalogs impertinent to astrophotographers. To address this challenge of surplus IDs, the csv titles data set from the classification modeling methodology was cleaned with same cleaning method described above, and all NGC, IC, and HD -like references were scraped from each title into a queued list. All duplicates were deleted.
2) Unlike the seed associations, this queued list did not have any cross-referenced names or IDs. The second step here was to compile the cross-references using SIMBAD, which was accessed indirectly via astropy's astroquery 'query_objectids' method. This method simply took a name or ID input and returned a list of all identifiers (aliases) recognized in SIMBAD. This is exactly what we are trying to accomplish with the SOAM; however, there are several reasons this indirect access to SIMBAD was impractical for large

volume text processing. First reason, a single SIMBAD query may take multiple seconds to process, which is impractical for a public facing website and painfully slow when performing text analysis. And second reason, not all aliases returned by SIMBAD are in an immediately usable format (i.e. all common names returned are prefaced with the tag, "NAME", or occasionally some of the returned identifiers are only 1 or 2 meaningless characters in length, all of which needs to be cleaned out prior to any text processing applications). Since the objective was to have fast and repeatable alias search ability for space objects, astropy's indirect access to SIMBAD did not fit the bill; however, it was still an invaluable resource for compiling space object cross-references.

Once these seed associations were wrangled and the SIMBAD cross-references were compiled, both data sets were formatted into lists of associated sets, and the SOAM data structure was built using the 'add_associations' method as described above. The overall process to building the SOAM can be visualized in the following figure.

**Figure 10**

*The SOAM Building Process*



Note, unique colors represent unique space objects, where ‘M’ represents the Messier ID of that space object, ‘NGC’ represents the NGC ID for that space object, ‘Sh2’ represents the Sharpless ID for that space object, ‘HD’ the Henry Draper Number, ‘C’ the Caldwell ID, and ‘N1’…’N3’… simply represent possible common names for the space object.

One subtle challenge to this approach was the “grabby” nature of the ‘add_associations’ method. This method was essentially a blind merging of all common item sets between the existing alias sets and the provided association sets which assumed all common item matches were accurately representing the same single space object. To verify this assumption, the alias sets of longest length (the alias sets with the most names and IDs) in the SOAM were reviewed. One alias set was found to not follow the assumption made in the ‘add_associations’ method, where the issue was traced back to bad data returned from SIMBAD. In SIMBAD, there is an issue where a query for all "NGC 6405" identifiers returned three Messier catalog IDs. This cascaded into an issue where the SIMBAD cross-referenced association set for “NGC 6405” was aligned with two other incorrect space objects from the seed source association sets, breaking our assumption and causing incorrect aliases to be returned for all three of these space objects. To correct this issue, all Messier catalog IDs were removed from the SIMBAD cross-referenced association sets, since all Messier catalog IDs were already accounted for in the seed association sets alongside their respective NGC reference. This approach solved the problem. The final SOAM built in this capstone was comprised of 25,642 names and IDs across 4,264 unique space objects.

*Building the Elasticsearch Synonym Filter*

Once the SOAM was built, some additional processing was needed to package the synonym filter. First, all SOAM names and IDs were filtered down to just the names and IDs found in the csv titles data. This simplified SOAM was then extracted into a list of alias sets

(essentially the values from the filtered aliases dictionary), and then the following items were expanded into the corresponding descriptions:

- All Messier catalog IDs (prefixed ‘m’) were repeated in the sets with the prefix ‘messier’
- All Sharpless catalog IDs (prefix ‘sh 2’) were repeated in the sets with the prefix ‘sharpless’.
- All Caldwell catalog IDs (prefix ‘c’) were repeated in the sets with the prefix ‘caldwell’

The final list of alias sets looked something like this: [{'messier 1', 'sh 2 244', 'crab', 'sharpless 244', 'm 1', 'crab neb', 'crab nebula', 'taurus a', 'ngc 1952'}, {'messier 2', 'ngc 7089', 'm 2', 'gcl 121'}, {'ngc 5272', 'messier 3', 'gcl 25', 'm 3'}, … ]. This list of alias sets was then exported to a text file were each set was written into its own row, and each of the items in the set were delimited by a comma and a space (‘, ’). This is the solr format recognized by the Elasticsearch synonym filter function. For example, when implemented, a search for ‘m 1’ should return features which include ‘m 1’ as well as ‘crab nebula’, ‘taurus a’, etc.

## Chapter 4 – Presentation of Results

### The Image Title Classification Model Results

The final classification model returned performed quite well, with an overall accuracy of 92.39% across all image subject type categories. To recap, the final model was a Linear Support Vector Classification model that utilized a TFIDF (1, 2) ranged word vectorizer with a regularization term 'C' set to 0.1, and a tolerance value ‘tol’ set to 0.1. The model was trained using 10-fold cross validation on a data set of over 590,000 AstroBin titles that were associated with subject type categories (one title to one subject type category). The data set was cleaned and augmented to a smoothing factor of 0.3 prior to training using the methods described prior.

**Table 4**

*Final Model Performance:*

Accuracy: 0.9238737062842454

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| DEEP_SKY | 0.96 | 0.94 | 0.95 | 66459 |
| GEAR | 0.91 | 0.96 | 0.93 | 5017 |
| NOCTILUCENT_CLOUDS | 0.97 | 0.98 | 0.98 | 4453 |
| NORTHERN_LIGHTS | 0.97 | 0.99 | 0.98 | 4502 |
| OTHER | 0.76 | 0.68 | 0.72 | 5807 |
| SOLAR_SYSTEM | 0.93 | 0.95 | 0.94 | 20048 |
| STAR_TRAILS | 0.92 | 0.95 | 0.94 | 4714 |
| WIDE_FIELD | 0.69 | 0.79 | 0.73 | 7264 |
| accuracy | | | 0.92 | 118264 |
| macro avg | 0.89 | 0.91 | 0.90 | 118264 |
| weighted avg | 0.93 | 0.92 | 0.92 | 118264 |

One of the most notable challenges in this classification modeling process would be the severe skew in categorical data within the provided text corpus. As described above in the methods section, great efforts were undertaken to mitigate the skew; however, the supplemental text used in the augmentation of the data was limited, leading to a high degree of synthetic (very pointed) repeats of 1-word and 2-word n-grams during the modeling process. The concern here was overfitting on false signal within heavily supplemented categories. Due to this concern a conservative smoothing factor was opted for (smoothing factor = 0.3) to augment the data for the final model training, even though improved accuracy was observed as the smoothing factor was increased (again, possibly due to overfitting on false signal from heavily supplemented subject type categories). This lower smoothing factor left the training data only slightly less skewed, but as shown above in figure 8, major improvements in accuracy and precision across sparse subject

type categories were observed after augmenting the training data with a smoothing factor of 0.1 or higher.

**Figure 11**

*Results of the Final Training Data Augmentation (smoothing factor of 0.3)*

Original Data ----------
Table of subject_type Counts:

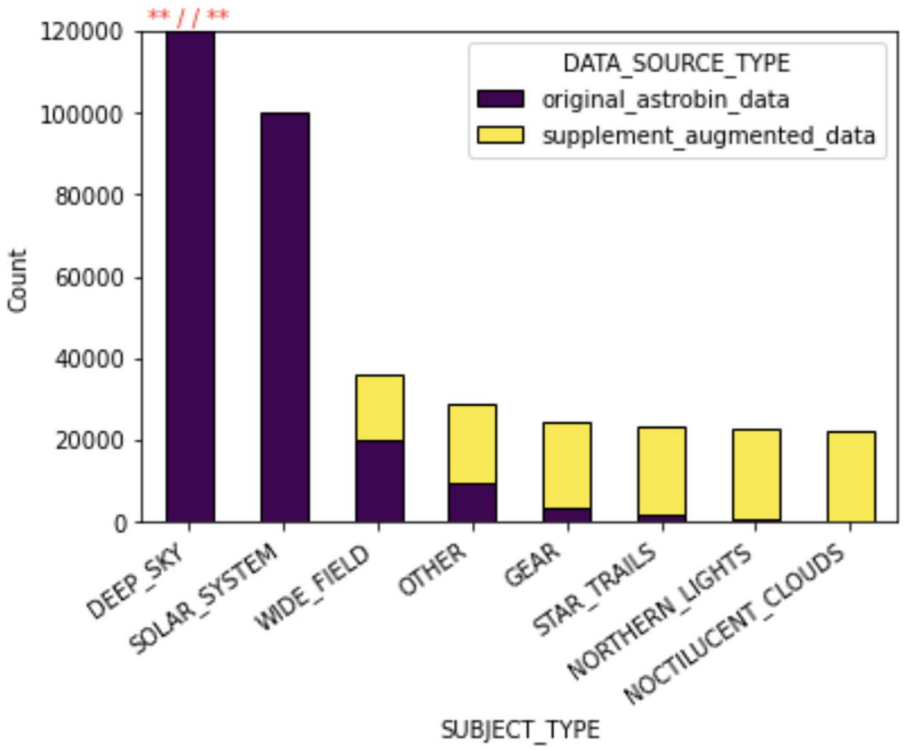
| | Subject | Count | Percentages |
|---|---|---|---|
| 0 | DEEP_SKY | 444255 | 0.751298 |
| 1 | SOLAR_SYSTEM | 111233 | 0.188111 |
| 2 | WIDE_FIELD | 20108 | 0.034005 |
| 3 | OTHER | 9670 | 0.016353 |
| 4 | GEAR | 3587 | 0.006066 |
| 5 | STAR_TRAILS | 1628 | 0.002753 |
| 6 | NORTHERN_LIGHTS | 674 | 0.001140 |
| 7 | NOCTILUCENT_CLOUDS | 162 | 0.000274 |

TOTAL = 591317

Augmented Data ----------
Table of subject_type Counts:

| | Subject | Count | Percentages |
|---|---|---|---|
| 0 | DEEP_SKY | 333153 | 0.563408 |
| 1 | SOLAR_SYSTEM | 100037 | 0.169176 |
| 2 | WIDE_FIELD | 36250 | 0.061304 |
| 3 | OTHER | 28942 | 0.048945 |
| 4 | GEAR | 24686 | 0.041747 |
| 5 | STAR_TRAILS | 23314 | 0.039427 |
| 6 | NORTHERN_LIGHTS | 22646 | 0.038297 |
| 7 | NOCTILUCENT_CLOUDS | 22290 | 0.037695 |

TOTAL = 591318




A few comments should be made on the model's performance across the different subject type categories. The results of the final model indicate it performs best on

‘NORTHER_LIGHTS’ and ‘NOCTILUCENT_CLOUD’ subject type titles; however, this is likely due to few reasons:

1) Both categories were over 97% supplement / augmented data (‘NORTHER_LIGHTS’ was only 2.98% original and ‘NOCTILUCENT_CLOUD’ was less than 1% original), most likely leading to some sever overfitting in the model on these two categories.
2) When one considers the actual image subject type, these two categories are very specific relative to some of the other categories. Implying that there is likely to be less diversity in image descriptions simply because the image target (the subject type) is very limited in scope. This is as opposed to, say, ‘DEEP_SKY’ or ‘WIDE_FIELD’ where the image subject types could be one or multiple of near infinite space objects, each object able to be described with various names and IDs.

… This being said, the lack of diversity anticipated in the titles of these two subject type categories when implemented on the live website should make this overfitting a minimal concern.

**Table 5**

*Breakdown of Original AstroBin Data Percentages Across Subject Type Categories*

| data_source_type<br>subject_type | original_astrobin_data | supplement_augmented_data |
|---|---|---|
| DEEP_SKY | 100.00 | 0.00 |
| SOLAR_SYSTEM | 100.00 | 0.00 |
| WIDE_FIELD | 55.47 | 44.53 |
| OTHER | 33.41 | 66.59 |
| GEAR | 14.53 | 85.47 |
| STAR_TRAILS | 6.98 | 93.02 |
| NORTHERN_LIGHTS | 2.98 | 97.02 |
| NOCTILUCENT_CLOUDS | 0.73 | 99.27 |

The final model performed worst on subject type ‘OTHER’ and subject type ‘WIDE_FIELD’. The ‘OTHER’ subject type was to be expected as a “worst performing” subject type as the nature of that category was more of a catch-all on the website; useful, but not a critical category to be accurate on. The ‘WIDE_FIELD’ category, on the other hand, likely struggled due to its similarity to ‘DEEP_SKY’. Again, considering the images represented in both ‘WIDE_FIELD’ and ‘DEEP_SKY’, some of the most popular space object targets are large enough and bright enough to fit both imaging styles respective to both categories. One example being the Andromeda Galaxy, a space object that when fully exposed in an image would be 6-relative full moon-lengths in the night sky. This makes it a great target for both wide-field and deep sky styled astrophotography, and consequently will be frequently described in titles across both subject type categories. Unfortunately, in this situation the skew becomes problematic, as the ‘DEEP_SKY’ subject type category was represented in over 56% of the training data set of image titles, even after augmentation. The ‘WIDE_FIELD’ subject type was only represented in 6.13%, even with supplemented augmentation title data. The impact of this skew would likely cause any shared signal (like a description of the Andromeda Galaxy) to be favored in the ‘DEEP_SKY’ category over the ‘WIDE_FIELD’ category. It should be noted, that being able to discern between ‘DEEP_SKY’ and ‘WIDE_FIELD’ images in the live website is a worthwhile endeavor, as the equipment and acquisition techniques used when taking these types of images are often radically different. Leveraging this type of distinction would benefit the AstroBin community if improvements could be made.

Regarding implementation of this model on the live site, a single subject type can be returned given a text string; however, it may be more beneficial to return the top-3 subject type options for the given text string. These options could then be provided in a radio selection on the

form during the upload process. When a user is provided this dynamic set of three categorical choices, it accomplished two things:

1) It restricts the user to pertinent choices that help make the classification of the image useful for future work and downstream process.
2) It will also provide the user with the opportunity to guide the classification of their image. Since the model is limited to only an image title text as input, it is prone to mis-interpreting the actual category intended by the user.

**Table 6**

*Example Model Selection Outputs for Test Titles:*

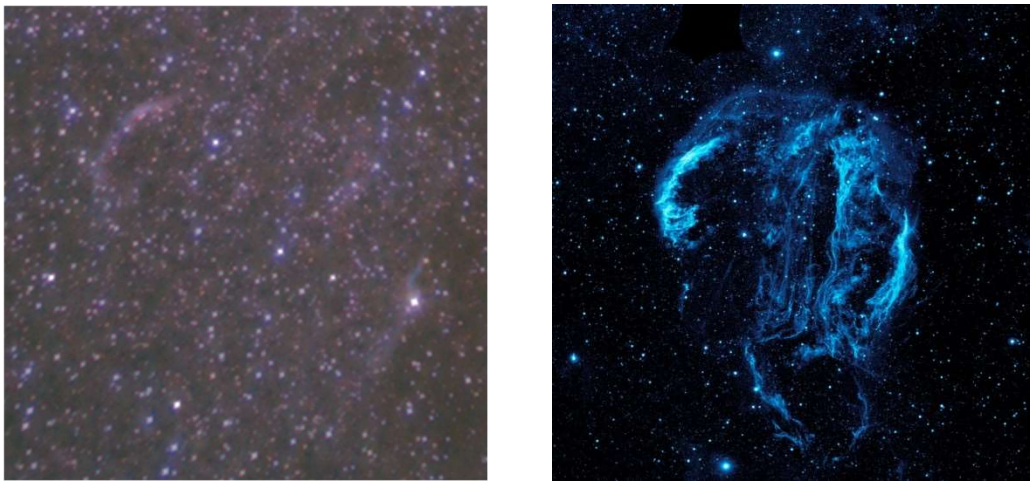
| TEST TITLE TEXT | Primary Selection | Secondary Selection | Tertiary Selection |
|---|---|---|---|
| '' | DEEP_SKY | SOLAR_SYSTEM | OTHER |
| 'Fruit loop' | DEEP_SKY | SOLAR_SYSTEM | OTHER |
| 'The CPC1100 Ready... To... Go!' | GEAR | WIDE_FIELD | NORTHERN_LIGHTS |
| 'Backyard Northern Lights' | NORTHERN_LIGHTS | WIDE_FIELD | OTHER |
| 'A mountain sky with a comet' | OTHER | WIDE_FIELD | NOCTILUCENT_CLOUDS |
| 'Hello World' | OTHER | WIDE_FIELD | SOLAR_SYSTEM |
| 'Pluto the Dog!' | SOLAR_SYSTEM | DEEP_SKY | OTHER |
| 'Pluto the Planet! (it's a planet)' | SOLAR_SYSTEM | OTHER | NORTHERN_LIGHTS |
| 'Milky Way in the Mountains' | WIDE_FIELD | OTHER | DEEP_SKY |
| 'My telescope pointed at M31' | WIDE_FIELD | GEAR | DEEP_SKY |

For example, the model classification of the right-side image title in figure 12, "What is the difference between a fruit loop and the large hadron collider?", returned the following top 3 category selections: 1st - 'DEEP_SKY', 2nd - 'SOLAR_SYSTEM', and 3rd - 'OTHER'. The 'SOLAR_SYSTEM' category is a little surprising given the context of the image and the title, but then considering the 'SOLAR_SYSTEM' subject type was the second most represented category in the title training text, this result begins to make more sense. For comparison, the left-side image title in figure 12, "Cygnus Loop Nebula", returned the following top 3 category

selections: 1st - 'DEEP_SKY', 2nd – 'WIDE_FIELD', and 3rd - 'OTHER'. Arguably, the category 'DEEP_SKY' is the best fitting subject type categories for both images, and this argument is supported by the final model output. This outlines the promising results which could be achieved in implementing this classification model on the live website.

**Figure 12**

*Example AstroBin image titled, "What is the difference between a fruit loop and the large hadron collider?" alongside NASA's Galaxy Evolution Explorer image of the same space object titled, "Cygnus Loop Nebula".*



*Note: LEFT) A* test image taken by the author of this capstone project, uploaded to the AstroBin website after training data was scraped from the website. Featured space object: The Cygnus Loop. Image taken with an unmodified Sony a6000 camera and a stock Sony E 35mm F1.8 OSS prime lens (a basic home / family portrait camera lens combo, not well suited for deep sky astrophotography) on a small Vixen Polarie star tracker. 135 x frames stacked, 25 seconds each. RIGHT) The same space object imaged by space orbiting telescope, GALEX - Explorer 83, in ultraviolet wavelength; https://www.nasa.gov/image-article/cygnus-loop-nebula/.

However, this example does allude to a particular point that should also be made. Choosing a title for an image can often be considered a very creative part of the process to an otherwise very technical hobby (a sentiment eloquently conveyed by astrophotographer, Dylan O'Donnel in his YouTube video, "Naming Conventions for Astronomy" [warning, explicit and rude humor]), so

any modeled classifications based solely on title should not be expected to align 100% of the time with the user's actual image. Hence it is suggested that multiple top categories from the model get incorporated as options in the image upload process, rather than completely automating categorization of images from a single output from the model.

**The Deliverables**

Both the classification model and the synonym filter were provided to the client for implementation. The synonym filter was simple enough to hand off, as it was just a single file that could then be added and referenced in a configuration file on the client's end. The model was a bit more involved as it required the final model to be exported into a joblib file type using the Python joblib library. Once the model was delivered, it required an additional helper method layer to use and implement the model in a practical manner. This was accomplished importing the delivered model (again using the Python joblib library) and using it with the self-made subject_type_classifications helper method. This helper method took the imported model along with a new image title text string and returned the desired number of top 'n' possible classifications predicted. Additional bonus deliverables were also provided to the client, as they could potentially prove useful for future work. These bonus deliverables were the Soam class along with a cleaned SOAM data structure, as well as a SIMBAD query helper method that could be used to batch a large queue of object ID queries.

## Chapter 5 – Future Work and Recommendations

First, a recommendation. It is advisable to not fully rely on title text alone to classify an uploaded image, especially if the intent is to fully automate the image classification process. If other downstream entries could be incorporated into the classification model, then more confidence could be had in an automatic selection of image categories. Also, based on the sever

skew of images within the existing categories, it may be prudent to further analyze the types of images being uploaded into AstroBin and determine if there were any other ways to categorize images. Would it make sense to merge the 'NORTHERN_LIGHTS' subject type category with the 'NOCTILUCENT_CLOUDS' subject type category, making, for example, a larger 'ATMOSPHERIC_PHENOMENA' subject type category? And/or would it make sense to split the 'DEEP_SKY' category into sub-categories like 'GLOBULAR_CLUSTERS', 'EMISSION_NEBULA', 'DARK_NEBULA', etc. This type of future work was proposed by the client early on, but the ability to first model existing subject type categorization was prioritized for this capstone. In order to continue image categorization analysis, additional data (like the equipment used, text description fields, and possibly the image file itself) will be needed. Methodology for the future analysis could include K-Nearest Neighbor (KNN) methods and / or Principal Component Analysis (PCA).

Another aspects which could be built on top of improved image classifications, would be recommender systems. Several types of possible recommender system implementations come to mind on a website like AstroBin. The client has already suggested a recommender system which proposes new targets for users to image based on user inputs like previous image types uploaded, other images liked / viewed, equipment used, general global location (northern vs southern hemisphere), and general time of the year (spring, summer, fall, or winter). Recommender systems could also be applied to the advertisement windows on certain web pages, tuning them to specific users, what they are looking at, and what pertains to their interests based on the profile activity. Additionally, this type of system could open the doors to more scientific collaborations among AstroBin members, where AstroBin may be interested in organizing group efforts and directing certain users to certain group projects through a recommender system.

Other future projects proposed by the client entail unlocking the historical context within the uploaded images. These projects would be ambitious and would lean heavily on robust and consistent image meta data. Projects include:

- Automated detection of proper motions of fast-moving stars.
- Automated detection of variable stars or novae
- Automated detection of movement in nebulae (see https://www.astrobin.com/ija7jc/B/. for an example of this)

In summary, regarding this future work, data wrangling efforts on the AstroBin website will always be an area of continuous improvement (this same thing can be said for most any other endeavor, not just AstroBin). As image meta data continues to get generated, compiled, and organized, the more it can be leveraged to accomplish unprecedented functions online and within the Astronomy community.

## Conclusion

To conclude this capstone project, the client was provided with an Elasticsearch synonym filter file containing a comprehensive set of space object names and their aliases, focused on space objects which were visible from Earth and of interest to astrophotographers. This synonym filter can now be implemented into the Elasticsearch configuration on the AstroBin website, so when a single space object ID is used to search a space object, all alias text features of that space object are also returned in the given search. Additionally, an image title text classification model was also provided to the client. In chapter 3 of this capstone, various augmentations of the title training data were explored at various smoothing factor settings (where a smoothing factor of 0.0 was not augmented at all, and a smoothing factor of 1.0 was fully augmented to have the title counts be equal across all subject type categories). Through several scaled iterations of sampled

training data and walking through various steps of smoothing factors and model variations; it was discovered that the most optimal modeling combination was a Linear Support Vector Classifier where the training data was augmented with a smoothing factor of 0.3 and the TFIDF vectorizer was based on 1- and 2- word n-grams. Additional tuning of this model further optimized its performance in image title text classification using a regularization term, 'C', of 1.0, and a tolerance value, 'tol', set to 0.1, resulting in a model which performed at a +92% accuracy rate within 10-fold cross-validation. One improvement that could be made with the classification model would be to further analyze the image categories and determine if the AstroBin website could benefit from alternative image subject types that pertained more specifically to specific clades of users. Overall, this capstone was successful in its deliverables to the client and leaves a clear path forward for future work and improvements for the AstroBin website.

## References


Amjoud, A. B., & Amrouch, M. (2022). Transfer Learning for Automatic Image Orientation Detection Using Deep Learning and Logistic Regression. *IEEE Access*, *10*, 128543–128553. https://doi.org/10.1109/ACCESS.2022.3225455

A. Common. (1884). "Orion-Nebula A A Common". In The Colour of the Stars by Malin and Murdin. (p. 29). 1984, Cambridge University Press.

Balakrishnan, V., & Ethel, L.-Y. (2014). Stemming and Lemmatization: A Comparison of Retrieval Performances. Lecture Notes on Software Engineering, 2(3), 262–267. https://doi.org/10.7763/LNSE.2014.V2.134

Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

C.D. Manning, P. Raghavan and H. Schuetze (2008). Introduction to Information Retrieval. Cambridge University Press, pp. 234-265.

Deng, X., Li, Y., Weng, J., & Zhang, J. (2019). Feature selection for text classification: A review. *Multimedia Tools and Applications*, *78*(3), 3797–3816.

DeVille, B., SinghBawa, G. (2022). "Text as data: computational methods of understanding written expression using SAS". Hoboken, NJ: John Wiley & Sons, Inc., 2022

Drechsler, M. ,Strottner , X., Sainty Y., et al., Research Notes of the American Astronomical Society, Vol. 7, id. 1, (2023) DOI: 10.3847/2515-5172/acaf7e

Dylan O'Donnel. (2023, March 13). Naming Conventions for Astronomy [Video]. YouTube. https://www.youtube.com/watch?v=J9lZKgdcvsQ

HaCohen-Kerner, Y., Miller, D., & Yigal, Y. (2020). The influence of preprocessing on text classification using a bag-of-words representation. *PloS One*, *15*(5), e0232525–e0232525. https://doi.org/10.1371/journal.pone.0232525

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. Universitat Dortmund, Informatik LS8 Baroper Str. 301 44221 Dortmund, Germany. *Lecture Notes in Computer Science*, 137–142. https://doi.org/10.1007/s13928716

Kavlakoglu, Eda (2023). Classifying data using the Multinomial Naive Bayes algorithm. IBM Tutorial. https://developer.ibm.com/tutorials/awb-classifying-data-multinomial-naive-bayes-algorithm/

Liu, Z., Bensmail, H., & Tan, M. (2012). Efficient Feature Selection and Multiclass Classification with Integrated Instance and Model Based Learning. Evolutionary Bioinformatics, 2012(2012), 197–205. https://doi.org/10.4137/EBO.S9407

M. Lucibella, A. Chodos, "January 2, 1839: First Daguerreotype of the Moon", APS News - This Month in Physics History, January 2013 (Volume 22, Number 1), https://www.aps.org/publications/apsnews/201301/physicshistory.cfm#:~:text=Astronomers%20quickly%20embraced%20the%20use,%2C%20on%20January%202%2C%201839.

Marc Wenger, Francois Ochsenbein, Daniel Egret, Pascal Dubois, Francois Bonnarel, Suzanne Borde, Francoise Genova, Gerard Jasniewicz, Suzanne Laloe, Soizick Lesteven, Richard Monier (1999). The SIMBAD astronomical database: The CDS Reference Database for Astronomical Objects. A&AS. arXiv:astro-ph/0002110. https://arxiv.org/abs/astro-ph/0002110

Polus, M., & Abbas, T. (2021). Development for performance of Porter stemmer algorithm. *Eastern-European Journal of Enterprise Technologies*, *1*(2 (109)), 6–13. https://doi.org/10.15587/1729-4061.2021.225362

Pranckevicius, T., Marcinkevicius, V. (2017). "Comparison of Naïve Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification". Baltic J. Modern Computing, Vol. 5 (2017), No. 2, 221-232. http://dx.doi.org/10.22364/bjmc.2017.5.2.05

Si, C., Zhang, Z., Chen, Y., Qi, F., Wang, X., Liu, Z., Wang, Y., Liu, Q., & Sun, M. (2023). "Sub-Character Tokenization for Chinese Pretrained Language Models". *Transactions of the Association for Computational Linguistics*, *11*, 469–487. https://doi.org/10.1162/tacl_a_00560 (https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00560/116047/Sub-Character-Tokenization-for-Chinese-Pretrained)

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. Statistics and Computing, 14(3), 199–222. https://doi.org/10.1023/B:STCO.0000035301.49549.88

Turriziani, Sara(2019). This Month in Astronomical History: 50 Years of CCDs. https://aas.org/posts/news/2019/10/month-astronomical-history-50-years-ccds#:~:text=In%201976%20Jim%20Janesick%2C%20an,Bigelow.

Vatri, A., & McGillivray, B. (2020). Lemmatization for Ancient Greek. Journal of Greek Linguistics, 20(2), 179–196. https://doi.org/10.1163/15699846-02002001

# APPENDIX

1. GitHub References:

   1.1. https://github.com/JsonBravo/capstone_2023_datawrangling

   1.2. README:

   This holds all Jupyter Notebooks used during my 2023 MSDS Capstone Project, as well as all data, deliverables, classes, and methods. Will leave as a public repository while the Capstone gets graded.

   A Description of Notebook files:

   0.0 -- A simple exploration notebook. The initial look at the title text data alonge with some comments regarding expected challenges.

   0.1.0 -- The development Notebook for the Soam Python class. The methods explored here were distiled and finalized in the 'soam_class' file found in the 'classes_and_methods' folder.

   0.1.1 -- The development Notebook for the final SOAM data structure.

   0.1.2 -- Doubles as an exploration as well as a development Notebook looking at Elasticsearch synonym filter application and file formatting. The final synonym filter was made at the end of this file.

   0.2 -- An exploration Notebook looking at standardizing space object names and IDs in a string to a single standard name. No real benifit observed, so this approach was not used in this capstone project.

   0.3 -- The development Notebook for the title data augmentation. The methods explored here were distiled and finalized in the 'title_data_augmenter_class' file found in the 'classes_and_methods' folder.

   1.0 -- The development notebook which all subsequent which all subsequent scaled iterations (1.1, 1.2, 1.3) were based on.

   1.1 -- This is the first scaled iteration of augmentation and modeling assessments

   1.2 -- This is the second scaled iteration of augmentation and modeling assessments

   1.3 -- This is the third scaled iteration of augmentation and modeling assessments

2.0 -- This is the development Notebook where final model tuning methods were explored and resulting models were assessed. The final model exported at the end of this Notebook was used in the subsequent '2.1_Tuned_Results' Notebook.

2.1 -- This is the development Notebook were the final Model results was assessed and reviewed.

3.0 -- This Notebook provides examples of each deliverable and how they can be utilized for implementation.

AstroCatelogues -- This is the spread sheet where all SOAM seed source data was corralled.

Synonym_Implementation_NOTES -- This is a PDF describing some challenges (and possible solutions) for implementing the Synonym Filter file (the 'astrononimcal_synonyms_112923' file found in the 'deliverables' folder).

The 'data' Folder (ZIPPED)

This folder contains the bulk data used and generated throughout this capstone project. Some notable files include:

'astrobin_titles_to_subject_types' -- A csv file provided by AstroBin, used as the training data for classification modeling.

'Augement_Equipment_OPT' A txt file of astronomy equipment details scraped from the OPT website (https://optcorp.com/) used as supplement data for the training data augmentation method.

'Augement_NOCTILUCENT_CLOUDS_GPT' -- A txt file of ChatGPT generated descriptions of Noctilucent Clouds, used as supplement data for the training data augmentation method.

'Augement_NOCTILUCENT_CLOUDS_Wiki' -- A txt file scraped from Wikipedia (https://en.wikipedia.org/wiki/Noctilucent_cloud) regarding Noctilucent Clouds, used as supplement data for the training data augmentation method.

'Augement_NORTHERN_LIGHTS_GPT' -- A txt file of ChatGPT generated descriptions of Northern Lights / Aurora, used as supplement data for the training data augmentation method.

The 'deliverables' Folder (ZIPPED)

This folder contains the data sets, files, and methods required for the deliverables, as described in the '3.0_Capstone_Deliverables' Notebook file.