Asteroid Orbits

Michael Harrington

Springboard : Capstone 3

Problem Statement

Can a machine learning model be made to classify the orbit type of Asteroids? How accurate a model can be created in this case, particularly when it is one that is an extremely imbalanced multi-class problem.

Dataset

My academic background in, and the general passion for, physics and astronomy was the impetus that led me to this dataset and from there to the project idea. I considered a variety of different datasets related to these fields (including ones concerning stars, exoplanets, galaxies, as well as telescopes). I settled on one concerning Asteroids in our solar system. It was found on kaggle.com and can be explored in the following link: Asteroid Orbits. The initial source of the dataset is the IAU's (International Astronomical Union) Minor Planet Center[1]. I liked the dataset because it was both extremely robust (over 1 million rows and 30 plus features) and in a pretty usable, mostly clean state.

[1] Link to their website - Minor Planet Center

Context and Goal

Asteroids are the rocky objects in our solar system that directly orbit the sun that are considered too small to be planets. They range from near spherical dwarf planets hundreds of kilometers in diameter (such as Ceres, Pallas and Haumea) to objects on the scale of meters. There are millions of these objects scattered throughout the solar system, though a majority do reside in the Asteroid belt between Mars and Jupiter. As we as a species continue to venture forth and explore more often and further out into the solar system the importance of knowing where these objects are and the orbits they take is obvious (for scientific, economic, and safety reasons).

The goal of this project is to create a classification machine learning model to accurately classify the orbit type of these objects. The target feature (Orbit_type) contains eleven distinct classes. The eleven classes are [MBA, Hungaria, Hilda, Amor, Distant Object, Jupiter Trojan, Phocaea, Aten, Apollo, Alira, Close Object [2]]. Thus it is a large domain multi-class problem (as opposed to a binary classification problem). The key factor of the dataset is that the target feature is extremely imbalanced. One class of orbit type (MBA to be specific) contains approximately 90.5% of the observations in the dataset. Obviously this means it will be vital to keep that fact in mind during modeling and evaluation. A model that simply predicts MBA for all Asteroids would therefore have an accuracy of about 90.5%. So a model will have to perform better than that to be of any practical use whatsoever (unless resampling or other techniques are used). Other evaluation metrics will also be especially important in this case as well.

A model that accurately predicts the orbit type of Asteroids could potentially be useful for scientists or researchers at Universities, Governing bodies, Scientific organizations (such as NASA, ESA, ROSCOSMOS, JAXA) or the ever growing number of Aerospace companies (including Blue Origin, SpaceX, Astra, Virgin Galactic). This is especially true going into the future as the ability to go into space is increasingly democratized.

[2] Close Object is shorthand for Object with Perihelion distance < 1.665 AU for readability

Data Cleaning and Exploratory Data Analysis

Data Cleaning

Initial examination showed a dataset that was in a pretty useful, mostly clean state. But there were a few key things I wanted to do before moving on to EDA and later modeling. The dataset contained quite a few cataloging and reference related features that clearly wouldn't be useful for modeling (such as the name of object, number of object, orbit computer, along with a variety of reference codes and flags). These features were all dropped from the dataset. I also created a few additional features for the dataset that I thought could potentially contain new or useful information (specifically Semi-minor axis, Approx. orbit length, and approx average orbital velocity and lastly velocity at both perihelion and aphelion). [3]

As mentioned before, the dataset was quite large (over a million rows) and thus a bit unwieldy for standard local machine learning. I decided to just use a fractional sample of the dataset going forward to speed up computational time. I used 20% of the initial dataset and checked to confirm that the sample maintained approximately the ratio of classes in the target feature orbit type (the downside of this was that the extreme minority classes had even fewer examples despite ratios remaining the same). Also a few changes were made simply for clarity (e.g. changing column names etc.). Finally missing values were addressed by imputing the median for numeric features and simply dropping rows with missing categorical features (of which were only a small fraction).

Exploratory Data Analysis

This being a classification problem and specifically an imbalanced multi-class one at that it was important to always keep that at the forefront when examining the data. I wanted to always have the target feature, orbit type, as the focus. In **Figure 1** the distribution of the orbit

[3] Perihelion and Aphelion are the nearest and farthest point of an object in orbit, respectively (from the Sun in this case). Semi-minor axis is half the shortest diameter of an ellipse (as opposed to the semi-major axis).

type classes can be seen. The large majority of Asteroids being in the MBA (Main Belt Asteroids) class is reinforced. About 90.7% of the Asteroids in the final dataset were MBA with the other classes being mostly around 1% each or less. Atira was an extremely small outlier with only 8 observations amongst the approximately 200,000 Asteroids in the dataset. This will all be vital to keep in mind during modeling and evaluation.

I want to note here that techniques such as down-sampling the dominant class or up-sampling the minority classes along with other options were considered but ultimately not implemented as they didn't end up being necessary in this case (which is made clear in the Modeling section). These techniques can potentially be very useful but also have downsides such as the loss of information in, or introducing bias to, your dataset.

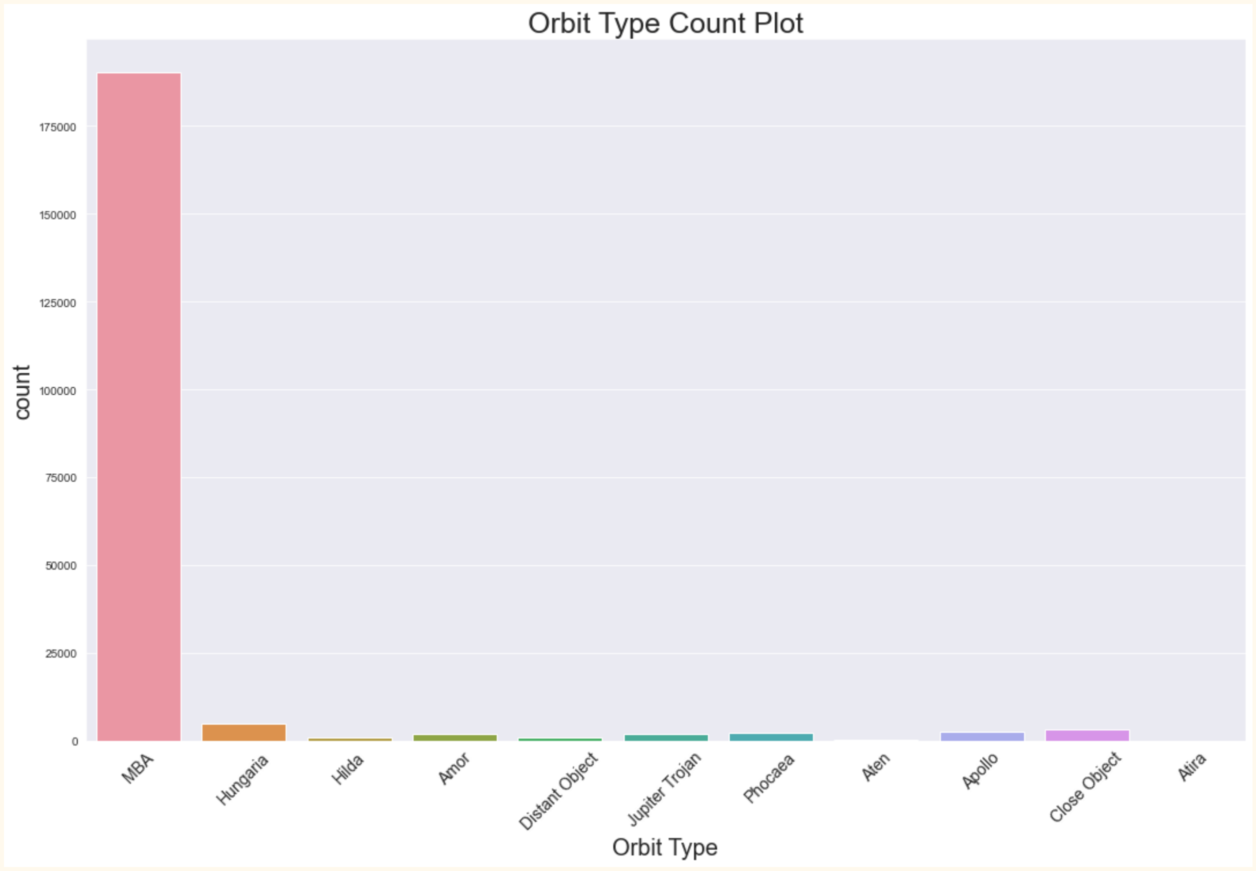


Figure 1: Visualizing the imbalanced nature of the target feature (Orbit type).

Going beyond examining our target feature, I believe it is always important to have a good look at all the numeric features in a dataset. This was done in several ways including with simple histograms and pairplots of various features. From these it was clear that most of the features had relatively clustered distributions but with a handful of large outliers. Upon further examination it was made clear that most of these observations were made up of Asteroids with an orbit type of distant object. This makes sense as distant objects (as implied by their name) are generally further from the sun and thus have more extreme orbital values and characteristics (such as orbital period, semi-major axis, aphelion etc.). Because of this I didn't really attempt to address these outliers as they were important distinguishing traits.

Whenever you have a dataset, such as this one, with many numeric features I also find it informative to create a correlation heatmap relating all those features. **Figure 2** displays the correlation heatmap for the dataset. While not as informative as in a regression problem (where the target feature is also continuous and numeric), it is still a useful tool. The heatmap shows that while many features aren't too heavily correlated, there are a few relating to elliptical orbit characteristics (such as semi-major axis, orbital period, aphelion, perihelion, semilatus rectum, semi-minor axis etc.) that do show pretty strong collinearity.[4] Despite this I decided to keep all these features to be safe and kept it in mind when modeling. It could very well be that many of these features provide the same information to many machine learning models.

Next I explored the relationships between the numeric features I thought were of particular importance and significance to the orbit type. I based this both off my prior knowledge of Astrophysics and the correlation heatmap. Among these I found **Figure 3** to be particularly interesting. It shows a plot of semi-major axis vs. aphelion and highlights some pretty obvious trends or clustering depending on orbit type. For example Jupiter Trojans, Hungaria, and Hilda Asteroids all show strong clustering whereas MBA, Apollo, Amor have a wider spread of values with a nearly linear positive relationship. The general trend shows the

[4]They may impart much of the same information to a model.

positive correlation between the aphelion distance and the size of the semi-major axis of the orbit (which aligns with expectations).

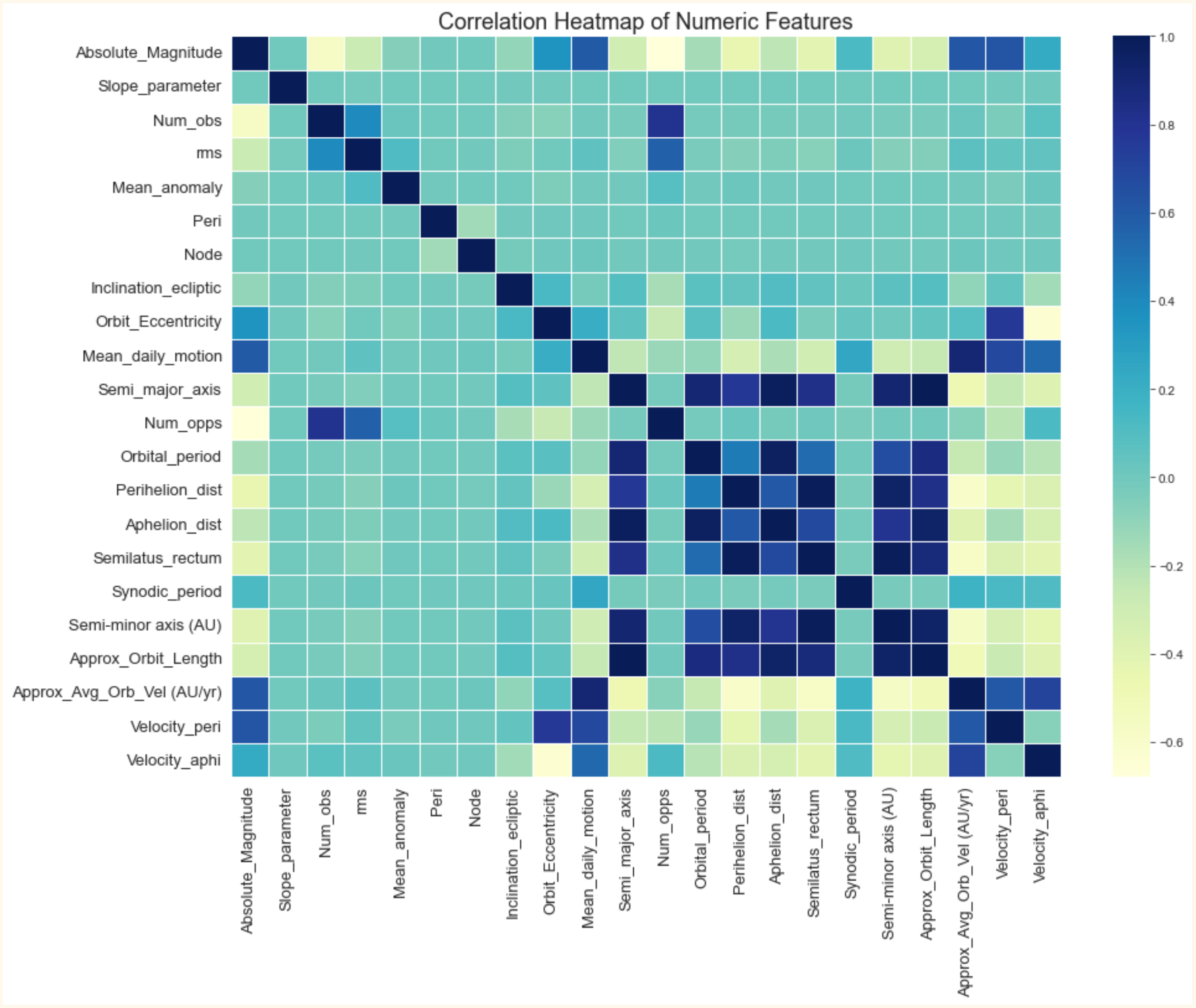


Figure 2: Correlation Heatmap of numeric features in dataset. The darker the shade the higher the correlation.

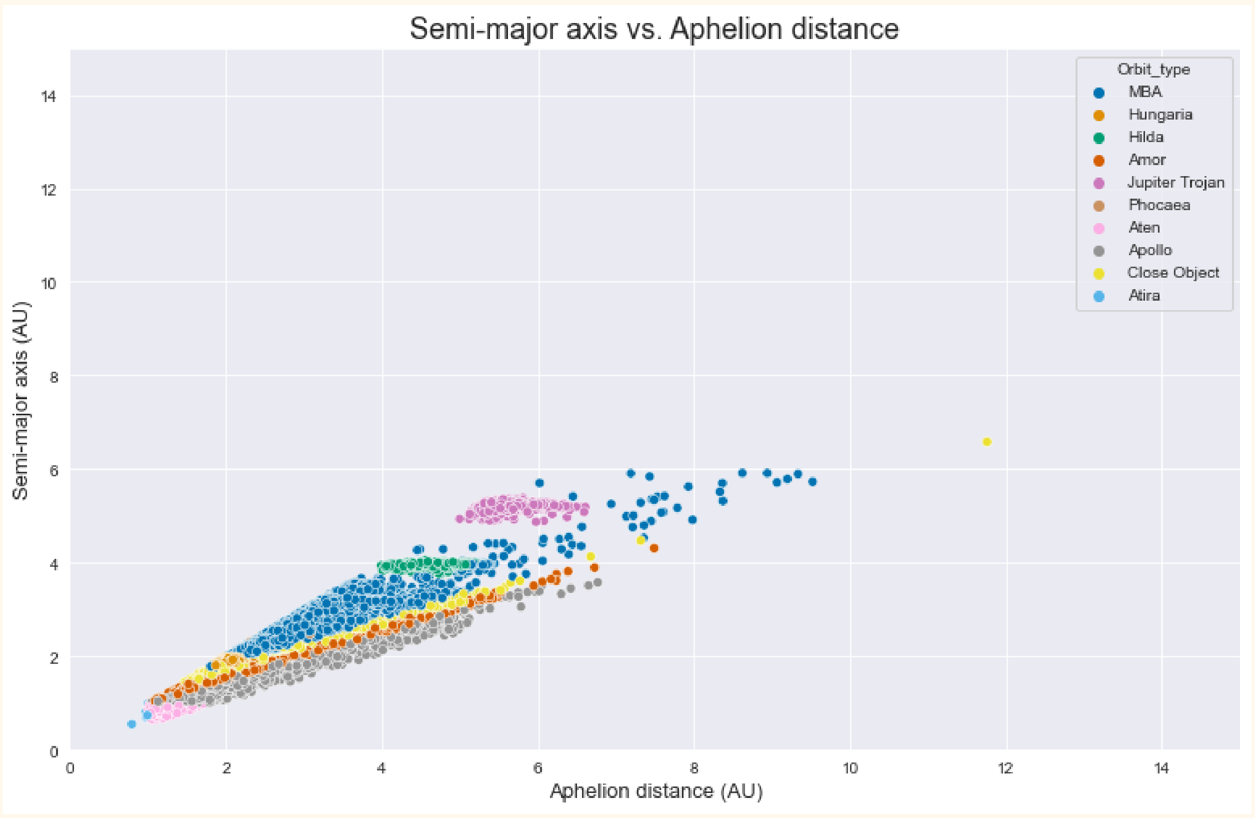


Figure 3: Semi-major axis vs. Aphelion distance (both in Astronomical Units) by orbit type.

While comparing the numeric features was informative, the main focus should still be on orbit type since our goal is to construct a model to predict it. With that in mind I wanted to visualize the distribution of some of the features separated by orbit type to search for any insights or trends. One of the best ways to compare the distribution of numeric features over a categorical one is to utilize box and violin plots.[5] Many plots of these types were created, specifically over the features I thought might be especially predictive. Three of those resulting plots can be seen in **Figures 4, 5, and 6**.

[5] Seaborn was utilized to create these and the other plots.

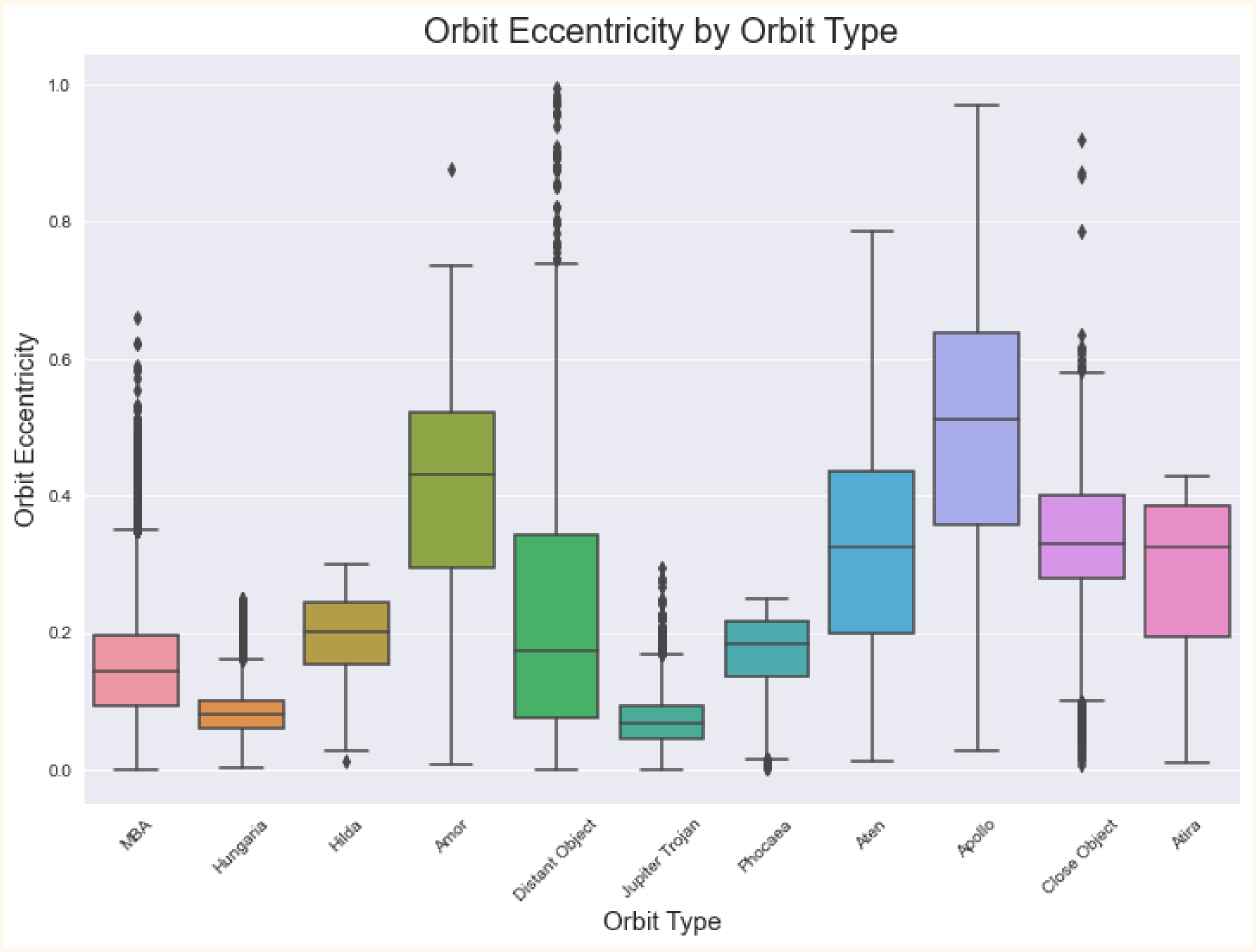


Figure 4: Orbit Eccentricity distribution by Orbit type.

Figure 4 plots the distribution of Orbit Eccentricity, which has values between 0 and 1 for a closed orbit where 0 is a perfect circle and larger values indicate a more squished, elongated orbit. There is some patterns and information here for some of the orbit types but certainly not all of them. Distant objects tend to have more extreme eccentricities which makes sense. Some, such as Jupiter Trojans and Hungaria, have generally very small eccentricities and are more clustered than others. A few classes seem to have very similar distributions as another such as Hilda and Phocaea for example.

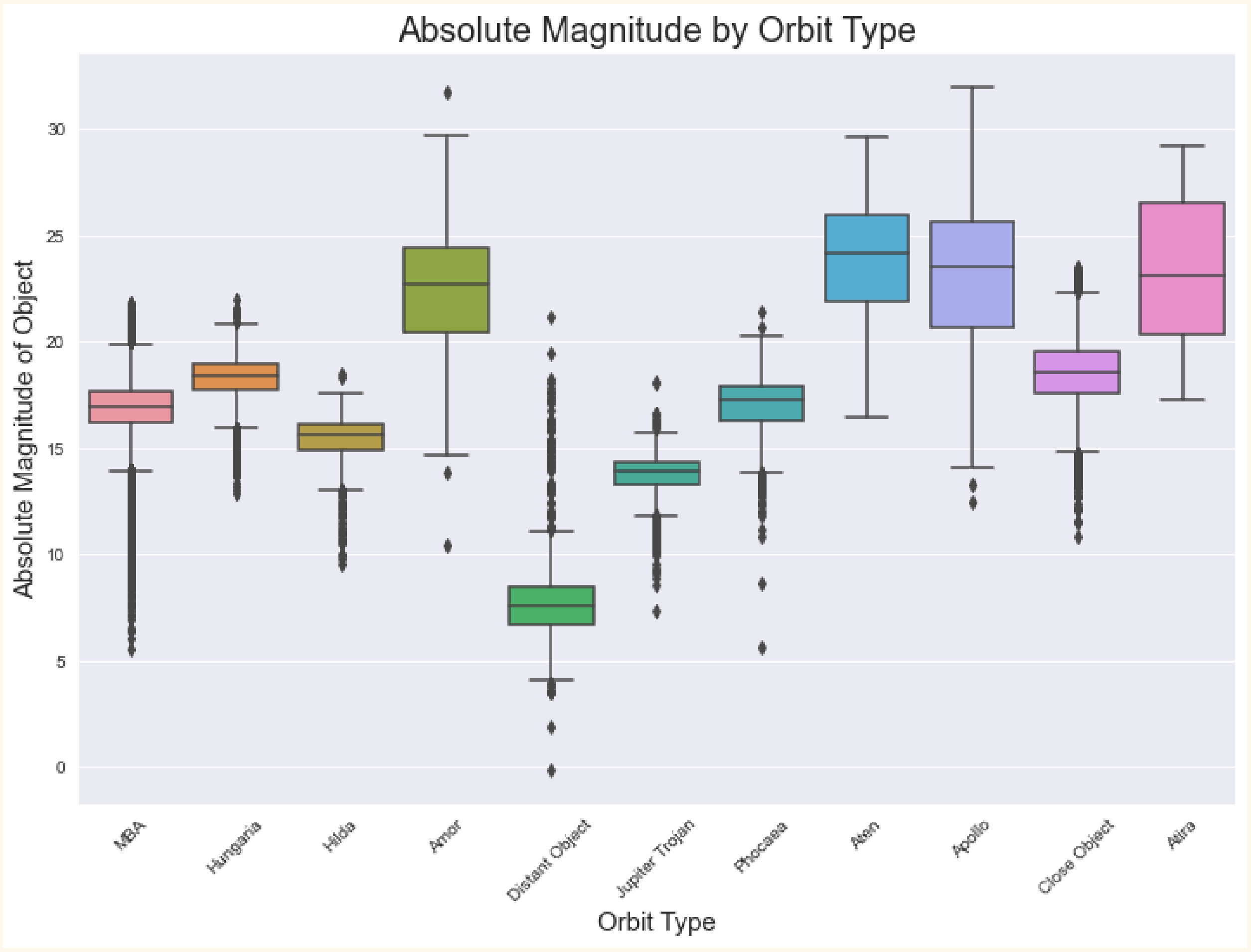


Figure 5: Boxplot of Absolute Magnitude of Asteroids by Orbit Type.

The boxplot above highlights the brightness or Absolute Magnitude, where unintuitively higher values indicate less bright objects, of the Asteroids in the dataset. There are some clear differentiations here, for example distant objects tend to have noticeably greater intrinsic luminosity (probably because they tend to be larger in order to be visible at all given their greater distances from the sun). The classes generally seem to have unique distributions, this may end up being an important feature for modeling and predicting the orbit type.

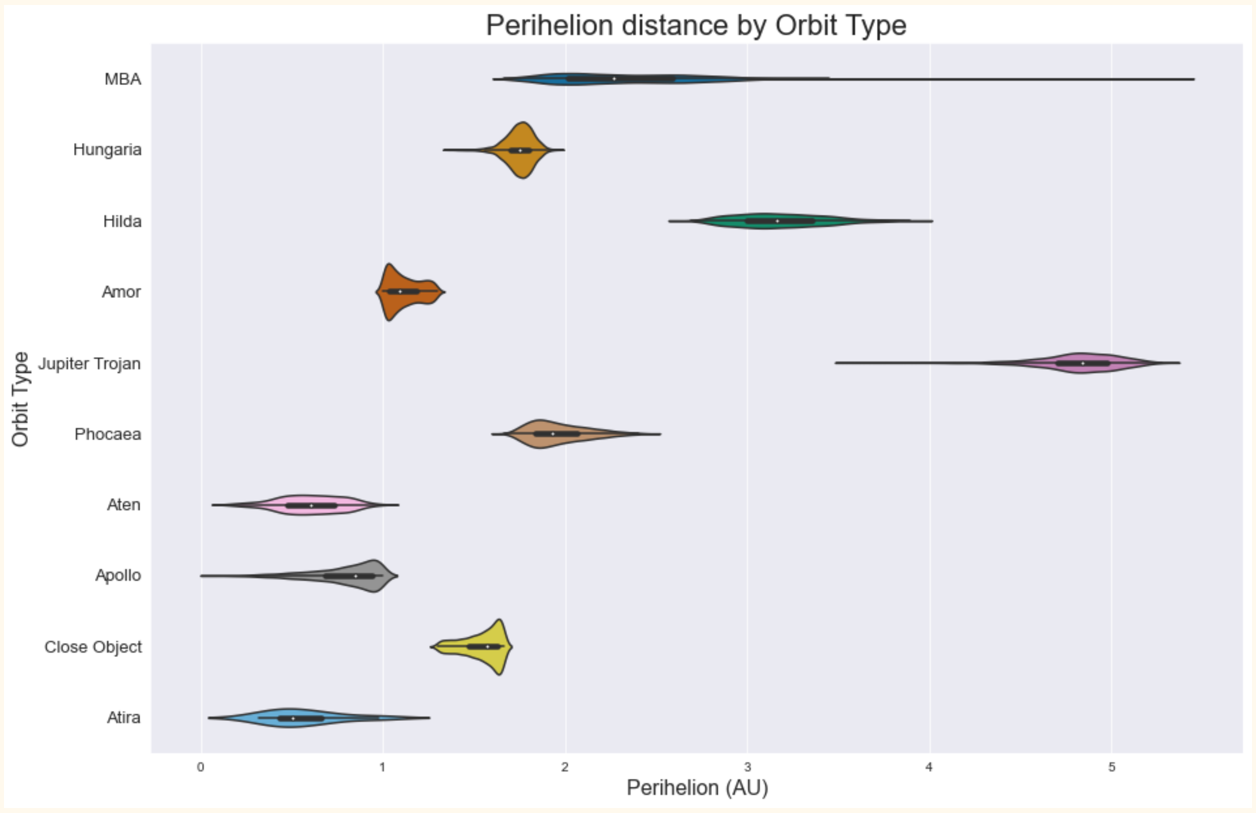


Figure 6: Boxplot of Perihelion distance by orbit type.

Figure 6 shows a violin plot of the perihelion value distributions. Distant objects are omitted due to them having many extreme values. There is definitely some clear differentiation here. MBA has a large spectrum of values whereas the other classes are often more strongly clustered. Each class seems to occupy a relatively unique distribution. I think this will probably be an important predictive feature come modeling.

Pre-Processing and Modeling

With the dataset thoroughly cleaned and explored its time to move on to modeling. Before doing so the data must be pre-processed and thus properly prepared for modeling. After preprocessing a variety of different models were created using many different classification algorithms and then evaluated and compared. The primary methods assessed were Logistic Regression, Decision Trees, SVM and Random Forests. In this section I compare their effectiveness. For reference the preprocessing and modeling were done with sklearn and keras.

Pre-Processing

The pre-processing was rather straightforward and simple for this dataset. First the data was separated into our target feature, orbit type obviously, and the predictor features (all the rest were utilized in this case). The dataset was almost entirely made up of numeric features but the few categorical features were one-hot encoded and the one datetime feature was converted to numeric. Next a train/test split was performed on the target and feature arrays, with a split of 70/30, to properly evaluate the models on unseen data. The predictor feature arrays, X_train and X_test in this case as per usual, were then standardized using standardscaler. The final train set included approximately 140,000 objects and the test set approximately 60,000.

Logistic Regression

The first algorithm attempted was Logistic Regression due to its relative simplicity. It's important to remember that LogReg is primarily used in binary classification problems whereas this is obviously a multi-class one. To account for this the multi-class parameter was properly set to *multinomial*. With this in mind I didn't necessarily expect LogReg to perform as well as the other options.

However, the results ended up being quite impressive. The best LogReg model performed remarkably well on the test set. The model yielded an accuracy score of **0.988273** on the test set. Once again it is vital to remember that in this case, with one orbit type class accounting for **90.7%** of all the objects, we are looking for particularly high accuracy scores. To better account for this we also need to utilize other evaluation metrics. Thus it is especially useful to closely examine both the resulting confusion matrix and classification report of the model on the test set. The classification report shows a macro average precision, recall and f1-scores of **0.86**, **0.83**, and **0.84** respectively. These are pretty solid results. The full matrix and report results, along with all the modeling in depth, can be seen in the linked jupyter notebook.[6]

The normalized confusion matrix can be seen in **Figure 7**. One can see that the model performed very well on most of the classes (nearly perfect on many such as Jupiter Trojan) but specifically didn't perform well on Phocaea and missed entirely on Atira (which to be fair had the extremely small sample of three objects). This is an interesting result that shows the difficulty a linear model, or really any model, has in predicting an extreme minority class. This shows that it might be worth considering upsampling this class but I decided first to see how the other model algorithms performed first.

The model feature coefficients were examined as well. Orbit eccentricity, velocity_aphi, and velocity_peri, approx_avg_orb_vel and absolute magnitude were the top five features. The full feature coefficients can be seen in the modeling notebook. Also hyperparameter tuning was attempted but yielded a model that performed only slightly better on the test set but took several orders of magnitude longer to both fit and predict. Those results can also be seen in the notebook.

[6]Pre-processing and Modeling

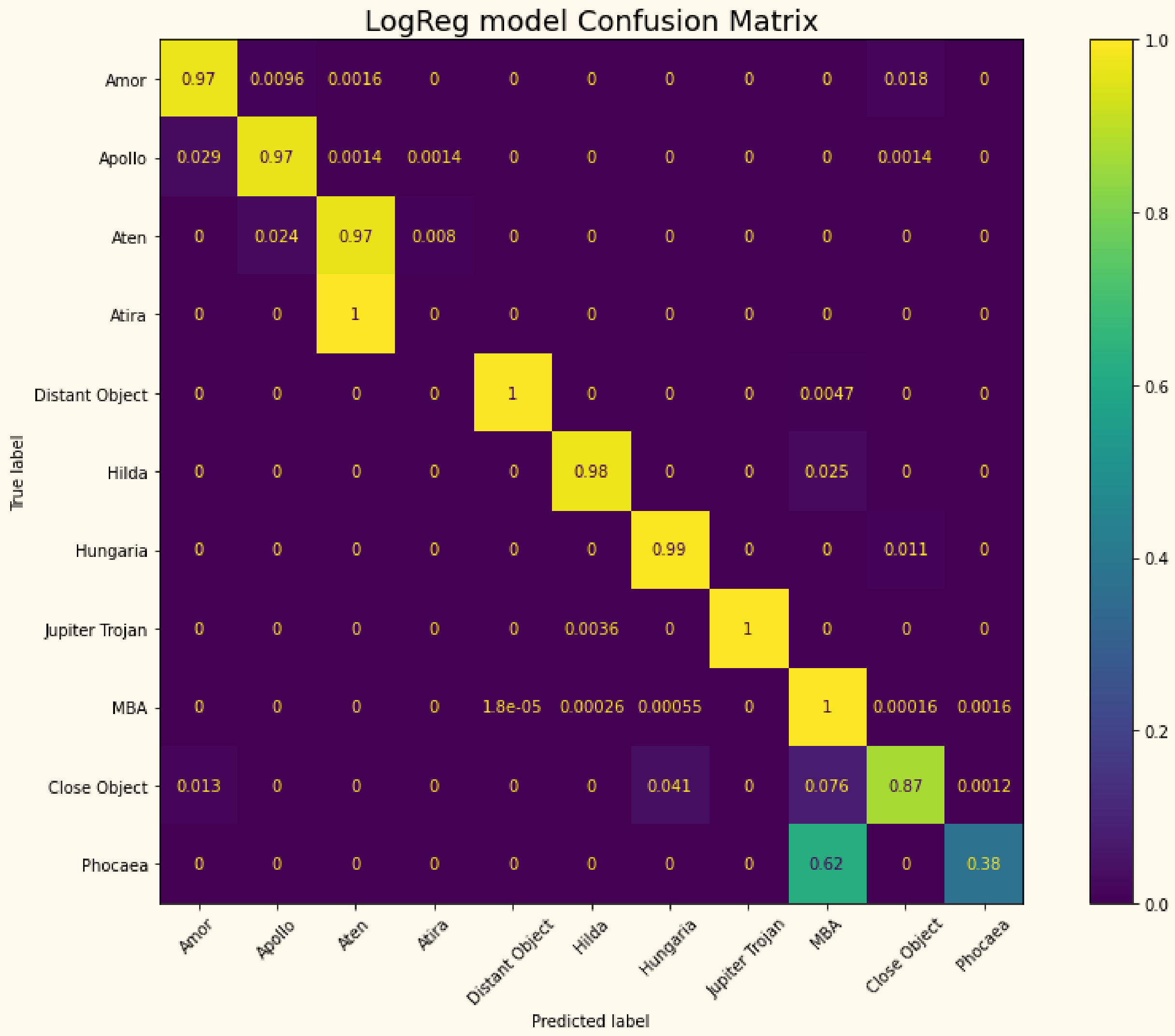


Figure 7: Normalized confusion matrix of the best Logistic Regression model.

Decision Tree

I went with a Decision Tree next due to its consistent effectiveness in classification applications. The resulting model performed astonishingly well on the test set. The model, with criterion hyperparameter set to entropy, had an accuracy score of **0.9998** on the test set. The results are very impressive and nearly perfect. As can be seen in the classification report the macro average precision, recall, and f1-scores were **1.00, 1.00** and **1.00.** To be clear these numbers were rounded up, obviously the model wasn't perfectly accurate. The normalized confusion matrix in **Figure 8** shows just how few mis-classifications there were. The model even correctly predicted the Atira class and did much better on Phocaea in comparison to the LogReg model. The model remarkably only mis-classified ten of the approximately 60000 Asteroids in the test set.

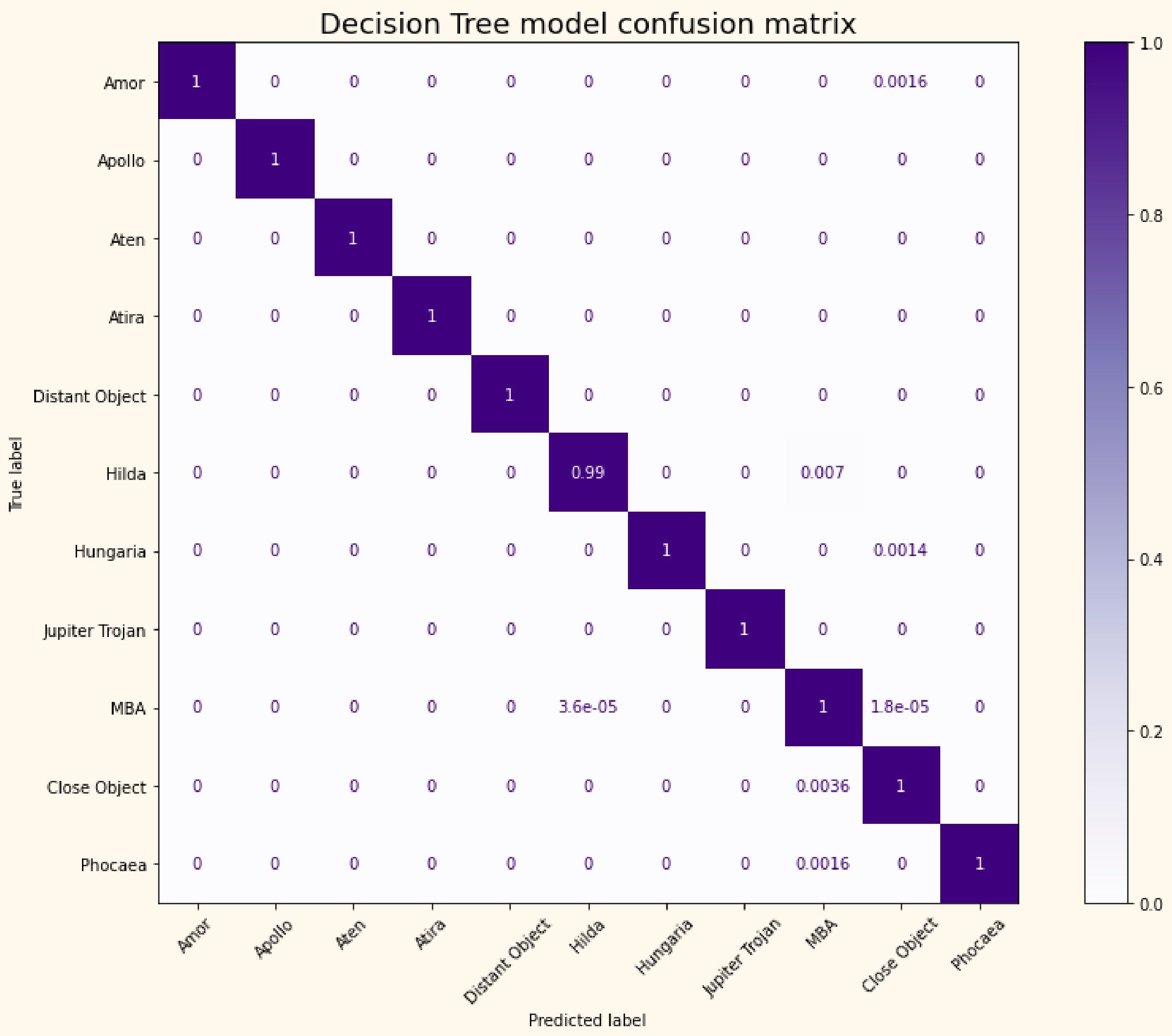


Figure 8: The confusion matrix of the best Decision Tree model

The model's feature importances can be seen in **Figure 9**. The model clearly heavily utilized the semi latus rectum feature.[7] The graph shows that only about ten features were really used by the model. This tells us that clearly some feature selection and cutting could be done for computational efficiency and the actual implementation of a model like this. One big contrast to the LogReg model is the difference in orbit eccentricity, which is a bit surprising. My theory is that this model found most of the same predictive information in the semi latus rectum feature with it being another ellipse mathematical property.

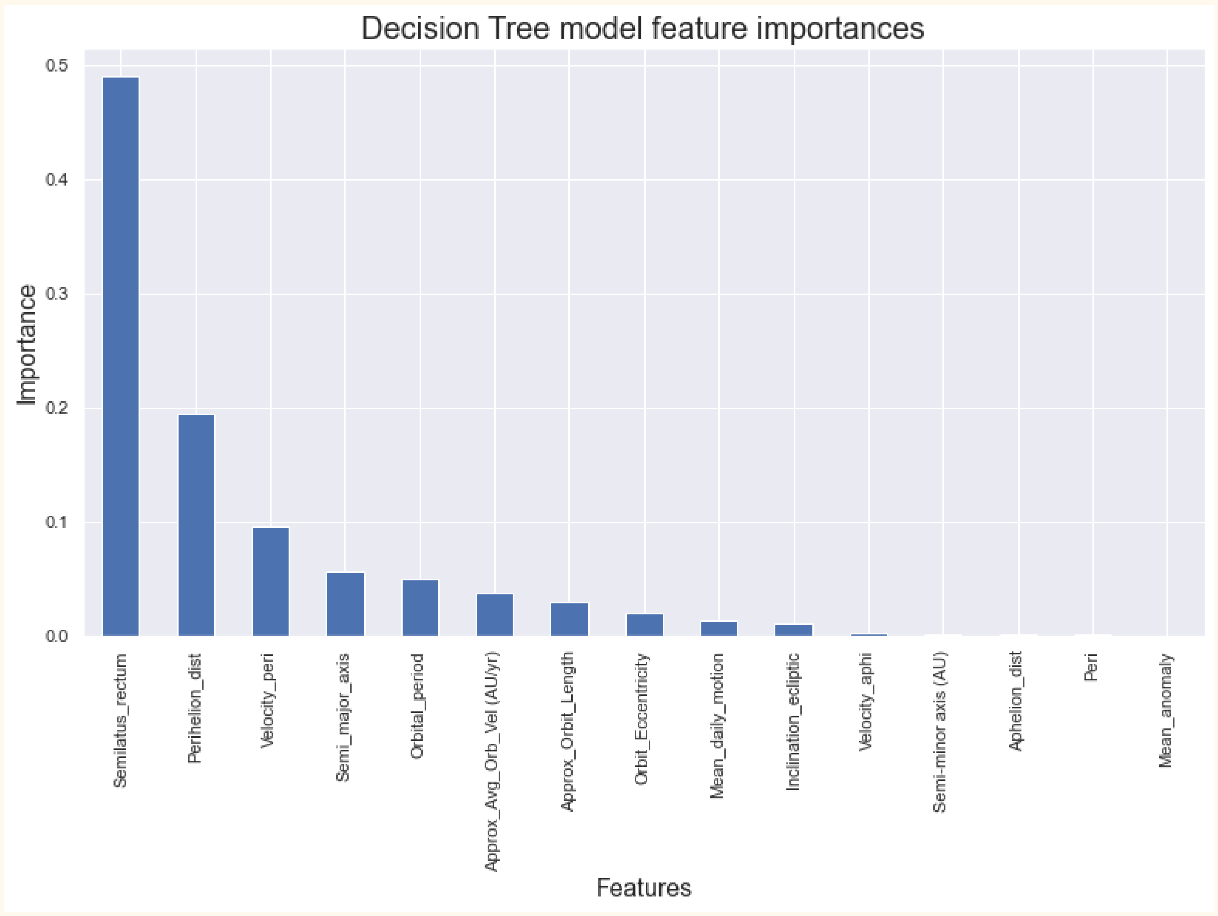


Figure 9: The Decision Tree model feature importances.

[7] Semi-latus rectum is the perpendicular distance from the focus of an ellipse to the curve (orbit in this case)

SVM

The third type of algorithm tried was a support vector machine, specifically a linear support vector classifier (linearSVC). The linearSVC in particular was utilized due to its inherent multi-class support with the multi_class hyperparameter set to crammer singer. I wanted to implement an SVM to see how another linear model would perform in comparison to the LogReg model. In the end it performed very similarly. The model ended up with an accuracy score of **0.989801** and macro average precision, recall and f1-scores of **0.87, 083**, and **0.85**. As the metrics bear out it seems to perform ever so slightly better than the LogReg model.

Just as with the other models, a normalized confusion matrix was created and is located in **Figure 10**. Interestingly the model had difficulty, like the LogReg model, in classifying Phocaea Asteroids correctly. It also classified a majority of them as MBA instead, though it's improvement there seems to account for a majority of its overall improvement over the LogReg model. The model also missed on the Atira objects. The top five features for the model were orbit eccentricity, velocity aphelion, aphelion distance, velocity perihelion, and the approx. average orbital velocity. Thus once again its overall similarity to the LogReg model makes sense as it shares three of the same top five feature coefficients (and the same top two).

Overall the SVM model performed essentially the same as the LogReg model. I'm not too surprised, I kinda just wanted to try it out and see how it would do as a point of comparison and to make sure the LogReg result was sensible for a linear model. I personally didn't have much experience with SVM models, so just implementing one was also partly the motivator to try it out here.

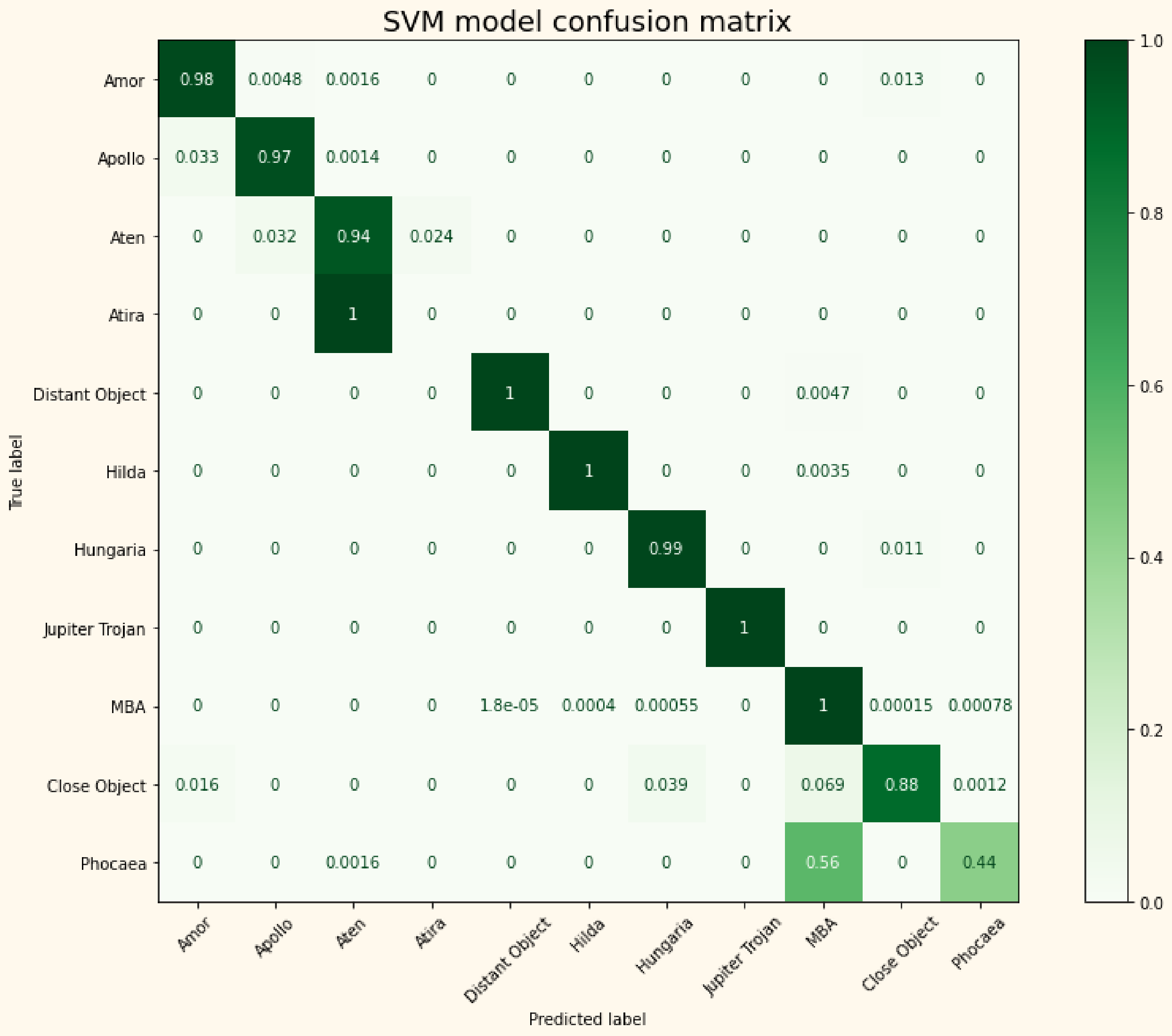


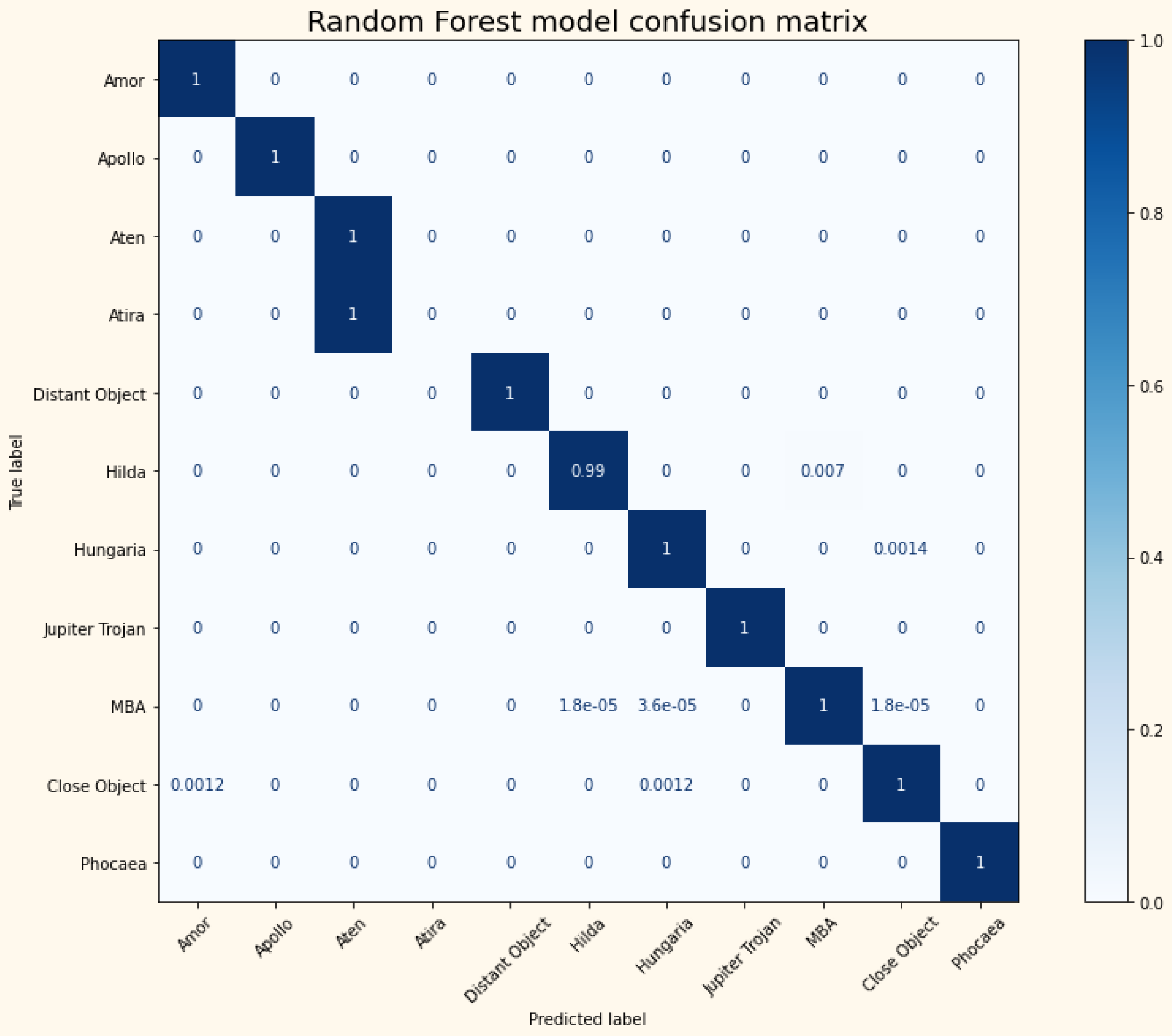
Figure 10: The normalized confusion matrix of the LinearSVC model.

Random Forest

From the beginning I knew I wanted to try a Random Forest classifier. Random Forests often perform exceedingly well in both regression and classification tasks. The incredible success of the basic Decision Tree model made me reconsider that position, but ultimately I decided to try it and see how they compared. It is always a good idea to try many different options for a larger possibility space, if only to see how consistent, logical and reproducible certain results are.

Unsurprisingly the Random Forest model performed extremely similarly to the Decision Tree model. The test set had an accuracy score of **0.999784,** just ever so slightly below the DT model. However the macro average precision, recall, and f1-scores were each **0.91**. The noticeable difference here can be attributed to the RF model missing on the dreaded Atira class. The normalized confusion matrix showing the model's results is available in **Figure 11**. Otherwise the model performs nearly perfectly like the DT model. For whatever reason this model misses on the small minority class.

A plot of the RF model's feature importance can be seen in **Figure 12**. In contrast to the DT model it seems to utilize a more even distribution of features. I am a little surprised by this, as in my experience RF models often seem to heavily weigh a relatively small number of features. But it did share the same top two features (perihelion dist and semi-latus rectum) as the DT model, albeit flipped. As in the DT model, hyperparameter tuning was considered but deemed unnecessary in this application due to the already extreme accuracy. RF models often take a very long time to tune, especially for larger datasets like this one. Overall the RF model performs incredibly well, but the simpler DT model gets the Atira class correct and also is quicker to run and relies on less features.



True label \ Predicted label	Amor	Apollo	Aten	Atira	Distant Object	Hilda	Hungaria	Jupiter Trojan	MBA	Close Object	Phocaea
Amor	1	0	0	0	0	0	0	0	0	0	0
Apollo	0	1	0	0	0	0	0	0	0	0	0
Aten	0	0	1	0	0	0	0	0	0	0	0
Atira	0	0	1	0	0	0	0	0	0	0	0
Distant Object	0	0	0	0	1	0	0	0	0	0	0
Hilda	0	0	0	0	0	0.99	0	0	0.007	0	0
Hungaria	0	0	0	0	0	0	1	0	0	0.0014	0
Jupiter Trojan	0	0	0	0	0	0	0	1	0	0	0
MBA	0	0	0	0	0	1.8e-05	3.6e-05	0	1	1.8e-05	0
Close Object	0.0012	0	0	0	0	0	0.0012	0	0	1	0
Phocaea	0	0	0	0	0	0	0	0	0	0	1

Figure 11: The Random Forest model's normalized confusion matrix.

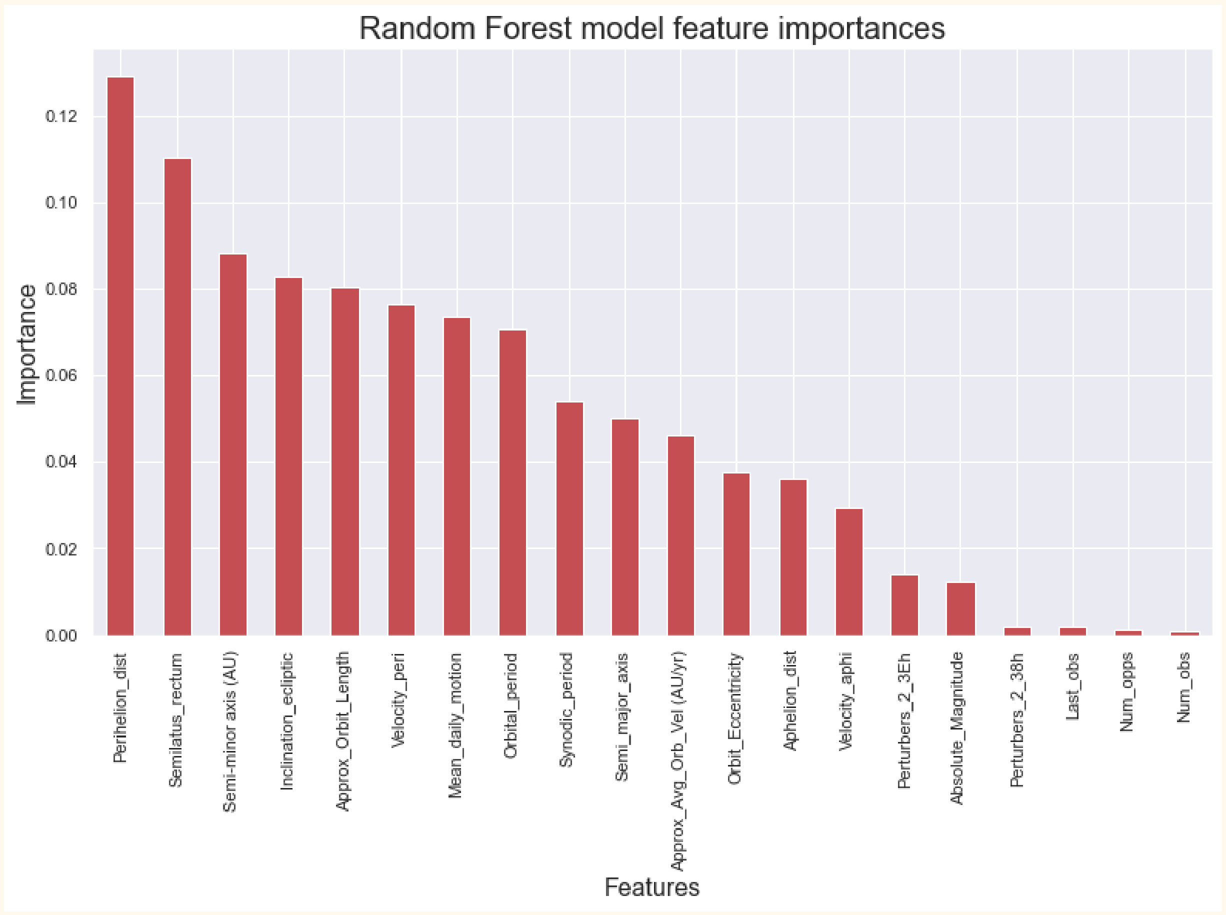


Figure 12: The feature importances of the Random Forest model.

Neural Network

Lastly I wanted to see how a few simple neural networks implemented using keras would perform on the dataset. This was mainly done as an exercise of inquiry and personal research rather than one where I would actually choose it as the final model (regardless of effectiveness). This is because I knew in the end I wanted an interpretable and explainable model.

In the end the best NN model, albeit quite a simple one, performed quite well. On the test set it, the best model, achieved an accuracy score of **0.996528** with a loss of **0.038896**. The best results were achieved with an Adamax optimizer with four inner layers. The full details can be seen in the modeling notebook. I was quite impressed with the results of the NN model, especially considering how little work was done to achieve that level of accuracy. It outperformed the linear based models but couldn't match the tree based models. I think it is probably possible to create a NN that could match them, but the tree models did it essentially right out of the box, so I don't believe it would be worth the time and effort. And as mentioned previously, even if it did match the other models, I would still choose the other option due to the inherent interpretability.

Model Metrics Comparison

Model	Accuracy score	Precision (macro average)	Recall (macro average)	F1- score (macro average)
Logistic Regression	0.988273	0.86	0.83	0.84
Decision Tree Classifier	0.9998	1.00	1.00	1.00
SVM (linearSVC)	0.989801	0.87	0.83	0.85
Random Forest Classifier	0.999784	0.91	0.91	0.91

	Accuracy	Loss	Optimizer
Neural Network	0.996528	0.038896	Adamax

Table 1: A comparison of the evaluation metrics for the best model of each classifier.

Final Model = Decision Tree Classifier

Summary and Conclusion

The goal of this project was to create a classification machine learning model to accurately predict the orbit type of Asteroids located in our solar system. A model that does this well could possibly be useful for a large variety of organizations, both public and private, in the space and aerospace related fields. The orbit type consisted of eleven distinct classes. The most important context and challenge of the problem was that orbit type was extremely imbalanced, with one class (MBA) containing 90.7% of all the Asteroids in the dataset. Thus the baseline of any model would be an accuracy of exactly 90.7% (one that would just select MBA for every Asteroid), and therefore the goal was to see how much better models could perform than that.

A variety of different models, using several different algorithms, were created and evaluated on the test set. A concise comparison of the resulting metrics for the best model of each type is located in **Table 1**. As it turns out all of the models ended up performing incredibly well. They all achieved accuracy scores far above the baseline and approached 100%. I don't have an obvious explanation for the high scores other than possibly the orbit type is easy to distinguish and highly correlated with some of the particular features. As can be seen in the table the Decision Tree model performed astonishingly well, outperforming all the other options in each metric, and in the end was the easy choice for the final model.

I do have a few ideas for further exploration or improvements given additional time and resources. Firstly I think the model could be improved with feature cutting and selection for better efficiency and easier implementation in the real world. As evidenced by the plots of feature importance in Figures 9 and 12 and the feature coefficients of the other models, the models all relied on many of the same features. So maybe cutting down to 10 or so features would be an option. Another possibility would be to upsample the minority classes and/or downsample the majority class and see how that changes things. Lastly the Atira class had so few samples that it may be best to ignore it (only 50 in the entire million plus initial dataset).