Dark Matter and Stellar Stream Detection

Philip Ekfeldt, Brian Kelly, Ademola Oladosu, Tony Xu

Abstract

Stellar stream and dark matter detection are active areas of research in astrophysics. Historically, this detection has been a manual process and has led to the discovery of stellar streams such as GD-1 and Palomar 5. If astrophysicists initially have strong confidence that certain stars are part of a stellar stream, probabilistic methods may determine the likelihood that an adjacent star also belongs to the same stellar stream. Using machine learning methods, we seek to develop an automated process that finds additional stars that are part of the stellar stream. After initially trying standardized classification methods, we establish an ensemble of a subset of kNN methods (ESkNN) as an effective means of classifying stars from simulated streams and real streams. Since these methods generally fail to capture non-linearities in the data, we also use a feedforward neural network approach that we expected to help capture non-linear properties. Our evaluation metric for these two metrics is the F1 score. The best model produced by ESkNN method had an F1 score of 0.75, when evaluated on GD-1 stars. The deep learning method (DeepSets) achieved an F1 score of 0.66. While both approaches can be tuned to prefer precision or recall, the ESkNN tends towards highly precise results (99%) whereas the DeepSets method tends towards results with high recall. We also attempt a metric learning approach that could help uncover a distance learning metric that is an improvement upon Euclidean distance.

1 Introduction

1.1 Problem Definition

Stellar streams are groups of stars that share a specific stellar structure. They can form when a small galaxy or a globular cluster outside of the Milky Way is accreted to the Milky Way. Depending on the trajectory of this external entity, its path may become disrupted, causing the stars to 'wrap' around the galaxy. This can lead to thin 'thread-like' structures, where the stars spread along a 'line' (along their orbit regarding the Milky Way) in 3D. These streams help astronomers and astrophysicists explore various galactic phenomena, such as dark matter detection, accretion history and galactic substructures [4][10][2][9]. Astronomers detect such streams from sky surveys via photometric data that separate metal-poor stars from nearby neighbors. Other methods incorporate radial velocity data for finer detection limits [7].

The original project had two main goals: on the one hand, we sought to develop tools for automating the detection of such streams. On the other, we sought to develop methods to better characterize the actual population of the (known) streams (i.e. obtain more confidence in the classification of the stars that do or do not belong to a given stream). The value of this latter pursuit would be useful for detecting sub-structures within the streams. These sub-structures could also illuminate interactions that may exist between dark matter and the stream. The approach for dark matter imprint detection would have used deep learning clustering algorithms. This would have required a deep learning framework to detect both stellar streams and the holes in those streams induced by dark matter interactions.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

The unsupervised streams discovery problem would require the development of techniques relying on deep clustering. Unfortunately, after exploring the data and trying out some classical clustering methods, such as DBSCAN and HDBSCAN described in the related works section below, it became obvious that this problem is highly complex and potentially intractable. Classical methods suffered from instability and high noise in terms of false cluster detection (stars may end up clustering for other reason than being in the same streams). Therefore, we chose to focus on the problem of better characterizing stream populations for the rest of the project. This enables us to use supervised approaches. The remainder of the report will focus on this problem.

1.2 Related Work

Given the profound interest in stellar streams, there has been some work in creating more streamlined systems for detecting stellar streams. The STREAMFINDER algorithm computes a proxy metric for the probability that a given star is within a stream using a six-dimensional phase-space hypertube.[12] Despite demonstrating remarkable sensitivity in stellar stream detection, this algorithm is computationally very expensive because it simulates many trial orbits and then picks the most realistic one for further calculations.[12] Density-based Spatial Clustering Algorithm with Noise (DBSCAN) is a clustering method that constructs clusters based on the density of points in a region. DBSCAN is robust to noise and outliers, and does not require the user to specify the number of clusters.[6] However, the number of clusters is chosen using an arbitrary cutoff.[5] Hierarchical DBSCAN uses information from a hierarchical diagram to more intelligently choose the cutoff for the number of clusters.[5]

While DBSCAN and HDBSCAN have both been used to identify stellar streams, the studies integrate spatial data to get orbit information to construct a phase space for each star.[13][3] However, this project sought to automatically detect stellar streams without the computational cost of integrating the orbital information for the stars in the data set. Early exploratory work with DBSCAN and HDBSCAN proved unfruitful (noise filtered was small percentage of total noise) on the entire data set due the ratio of background stars to stellar stream stars. By downsampling the background stars significantly, we were able to get better class predictions but we did not have a structured approach to generate the same quality of results in a larger data set. This further contributed to why the unsupervised problem would have been intractable to finish in one semester.

1.3 Project Scope

In light of the scope transformation, our new problem statement can be reformulated as follows: if we have discovered some stars that we believe are part of a stellar stream, what can we say about other stars that are in the spatial neighborhood? To answer these questions, we try three major techniques.

Firstly, we explore standardized classification techniques and see if these approaches capture enough of the internal structures to perform adequate class prediction. Particularly, we are interested in the performance of kNN given its general performance in solving spatial problems. Secondly, we explore a feedforward neural method for class prediction, hypothesizing that the non-linear dependencies we anticipate may be captured through this technique. Thirdly, we also hypothesize that an optimal distance metric may be learned. If this were correct and there were a preferable distance metric to Euclidean distance, this could be used to optimize the results obtained from both the traditional methods and the deep learned methods.

2 Methods & Data

2.1 Data

The Gaia data set we are working with consists of information on 1.7 billion stars, which was obtained from the second data release. Of these, 361 million have a 2-parameter astrometric solution, which means that we have information about the position in the sky combined with the mean G magnitude (a measure that is directly proportional to the logarithm of the star's distance). The remaining 1.3 billion stars have a 5-parameter astrometric solution, which includes the positions on the sky, parallaxes, and proper motions. Given that proper motions are expected to be similar for stars in a cluster, we would restrict our data set to these 1.3 billion stars.

Due to the intractability of analyzing the entire Gaia data set (even with the restriction to 1.3 billion), we sought to establish proofs of concept on simulated streams. We utilized a data set of over 300 simulated streams, each of which contained 6-3000 stars that are arranged spatially in a manner similar to existing known stellar streams. These are also assigned to a specific spatial coordinate. Our scaled-down analysis allows us to construct data sets that draw on a combination of these simulated stellar streams and background stars taken from the Gaia database. Knowing an approximate ratio of background stars to stellar stream stars that exist in the vicinity of known stellar streams, we can develop miniature data sets that fit this profile. This is intended to mimic the star distribution in a given neighborhood. This way we have approximately 300 data sets on which we can train and test various approaches.

We also have a cut of Gaia that includes one known stellar stream (GD-1) and the corresponding stars in that neighborhood. This includes 1979 stellar stream stars and approximately 7 million background stars, which is approximately a ratio of 3500:1 for a noise to signal ratio. After determining model performance on our stellar streams, we evaluate and report our final model performance on this specialized cut consisting of real stars. A 3-D representation of the arrangement of the GD-1 stars is shown in Figure 1. Although spherical coordinates were used in the analysis, Cartesian coordinates are shown for convenience.

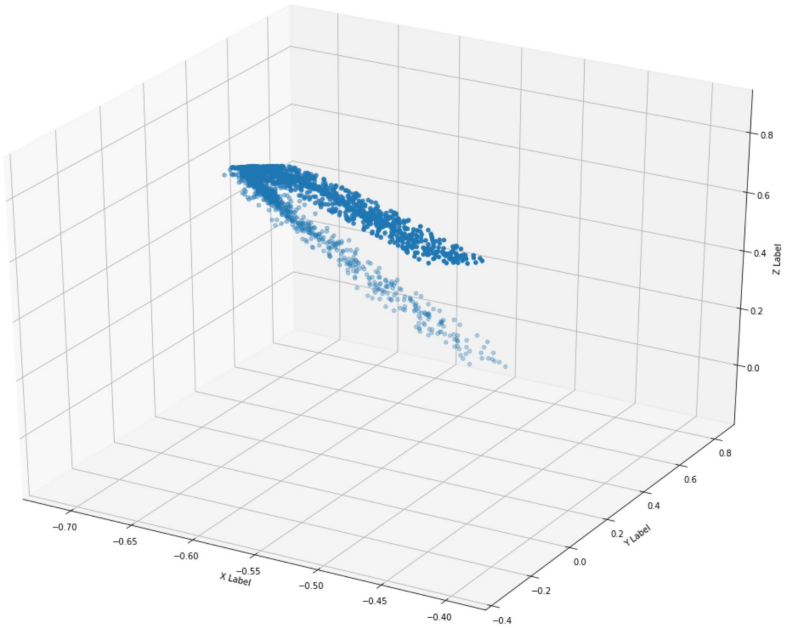

Figure 1: Visualization of the Gaia GD-1 stellar stream.

Below is an explanation of some of the key variables that will be used in our analysis

1. **Key Variables**
   - **Celestial Reference System**
     The project primarily focused on four spatial features from the Gaia data set that were taken with respect to the Barycentric Celestial Reference Stem, which is a common reference system in astrophysics: *ra* is the Barycentric right ascension; *dec* is the Barycentric declination; *pmra* is the proper motion (movement of an astronomical object relative to the sun frame of reference) in the direction of right ascension; *pmdec* is the proper motion in the direction of declination.[8]
   - **Color**
     Each star in the data set also contained several spectrometry metrics: *g*, *g_bp*, and *g_rp* are the mean absolute magnitudes (brightness of a star as seen from a distance of 10 parsec) for the green band, the green band minus that of the red band, the green band minus that of the blue band, respectively.[8]

2. **Isochrone Filter**

   Using software provided by our mentors, the age and metallicity for the stellar streams were determined from the g_rp and g_bp values for each star in GD-1. These values were then used to obtain the distance from Earth for the stars in the stream and background. Thresholds were created as described in the methods section below to reduce the ratio of GD-1 stellar stream to noise stars in the Gaia data set from 1:3500 to 1:250. For the mock streams, noise stars were added in a ratio of 1:250 as described in the methods below in order to construct realistic simulations.

3. **Cartesian transformations** (x, y, z, v_x, v_y, v_z, w_x, w_y, w_z)
   Using the distance obtained above, the coordinates were transformed from spherical to Cartesian coordinates. In one of our methods, we briefly offer a comparison of the results obtained when the features are constructed as spherical vs. constructed as Cartesian coordinates. The radii from the isochrone filter were used for this calculation.

2.2 Method: Ensemble of Subset of kNN Classifiers (ESkNN)

2.2.1 Motivation

In general, we expect that stars belonging to a particular stellar stream will be in spatial proximity to other stars that belong to that stream, as measured by Euclidean distance. Data sets with this structure can often be evaluated with a kNN algorithm. In our case, each star can be assigned one of two classes (belonging to a stellar stream versus not belonging to a stellar stream) based on the most common class among its k nearest neighbors.

2.2.2 Preliminary Analysis

When training a kNN model, we typically stratify a data set on the label so as to ensure the ratio of each class is the same in both the training set and the test set on which the model will be evaluated. However, we are unsure whether training a kNN model on a data set with class imbalance will have a material impact on the class prediction of the test set. This means that an additional hyperparameter we can tune is the class balance in the training set. We refer to this ratio of non-stellar stream stars to stellar stream stars as the "training multiple".

As described in our problem statement, we are interested in seeing if we can predict the class of stars knowing the class of other stars in the vicinity. Let us consider first a stellar stream with μ stars in the cluster. Let us now consider model A which starts off knowing the correct class of 0.9μ stars and model B which starts off knowing the correct class of 0.2μ stars. Given that model A is provided more knowledge to train on, we expect it to perform better in identifying the remaining 0.1μ stars than model B would in trying to identify the remaining 0.8μ stars. How rapidly the performance of these hypothetical models declines is important so we consider the percentage of stars being trained on as an additional hyperparameter. We refer to the percentage of stars in the stellar stream that we train on as the "training ratio".

After some preliminary analysis, we recognized that different kNN models had substantially different class predictions for different stars. We hypothesized that the kNN methodology could be improved by opting to use a subset of the best-performing kNN models and having these models vote on a final class prediction for a given star. The assumption is that while one kNN model may not be satisfactory here, the collective knowledge from an ensemble of kNN models may provide better class prediction.

2.2.3 Procedure

1. For each of 10 simulated stellar streams, develop an appropriate data set of stellar stream stars and background noise (non-stellar stream stars)
2. Using the following hyperparameters, we generated 160 models and evaluate each for precision and recall
   - k = 1,2,3,4,5
   - training_multiple = 1,26,51,76,...376

- training_ratio = 0.1, 0.2

3. We select the 25 best models based on the average F1 score for each of the 10 stellar streams. We refer to these 10 streams as Group A.
4. Using 10 new simulated stellar streams, we try each of these 25 kNN models and predict the class of a fixed test set.
5. Each model's prediction is recorded as a vote and the sum of the votes represents how many models out of 25 predict the star to be part of a stellar stream.
6. Instead of using a majority vote system, we can use these ten stars to tune for a threshold that represents the minimum percentage of votes that need to classify a star as part of a stream before our ensemble method makes that prediction. The optimal threshold was determined to be 16% (4 out of 25 models).
7. Using another 10 new simulated stellar streams, we can predict the class of each star based on the combination of the 25 kNN models and the optimal threshold we've tuned for. We refer to these 10 streams as Group B.
8. We use our system of 25 best models and tuned threshold and predict the class of stars on GD-1.

   To illustrate the ESkNN method, the three plots below (Figure 2) on the left show the performance of individual kNN models. While each model falsely predicts many of the background stars to be part of a stellar stream, there is some commonality to the stars that are properly predicted. When we aggregate these results using a given threshold, we arrive at the image on the right, which has considerably fewer false positives.

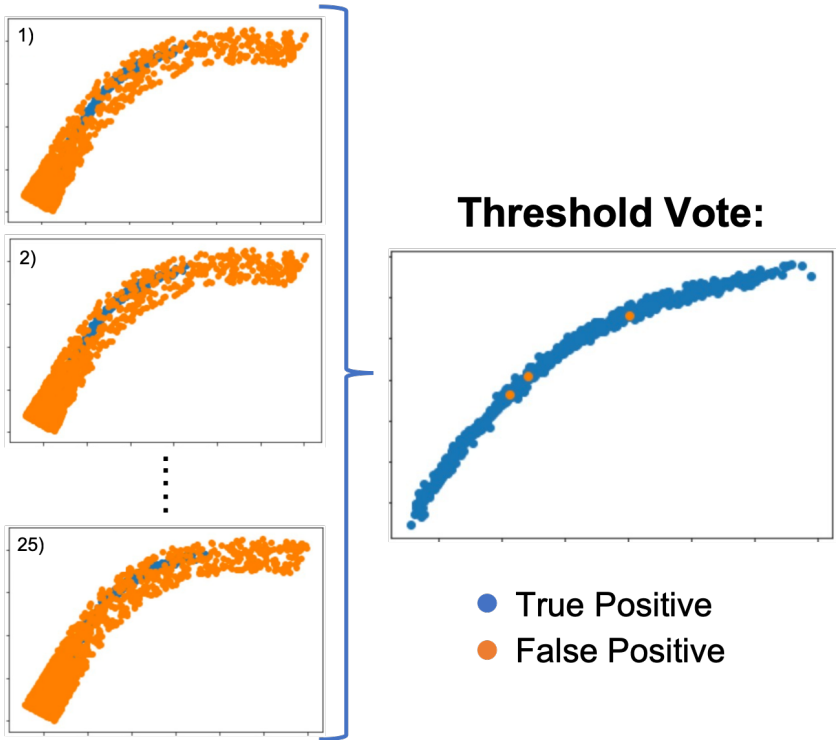

Figure 2: Visualization of the ESkNN Method

2.3 Method: Metric Learning

2.3.1 Motivation

We also examined the auxiliary effects of metric learning on our existing models. Metric learning aims to automatically construct task-specific distance metrics from supervised data. The learned

distance metric can then be used to improve performance of various tasks, such as the kNN model. Supervised metric learning takes labeled data and learns a distance matrix that minimizes the distance between points of the same class and maximizes the distance between points of different classes.

2.3.2 Preliminary Analysis

For the implementation we used, the metric learning algorithm learns a Mahalanobis distance, parametrized by a positive semi-definite matrix M: [14]

$$D(x, y) = \sqrt{(x - y)^T M (x - y)}$$

This learned distance metric is then utilized to improve the performance of our models.

Because the algorithm utilizes leave-one-out regression to learn the optimal distance metric, performing metric learning on our entire data set takes a significant amount of time. To address the computationally expensive nature of the algorithm, we decided to first downsample the amount of noise points utilized in metric learning. Rather than the 1:250 stream-to-noise ratio reflective of the actual GD-1 stream, we chose to use a reduced 1:50 stream-to-noise ratio. This downsampling significantly speeds up the the metric learning and was able to yield an improvement in performance for the kNN on some stellar streams.

2.3.3 Procedure

1. For each of ten simulated stellar streams, develop an appropriate training data set of stellar stream stars and background noise (non-stellar stream stars)
2. Fit metric learning using all training stream points from step 1 and a downsampled version of the training noise points from step 1
3. Pass learned metric to kNN and train kNN on full data set
4. Train regular kNN without metric learning on full data set
5. Evaluate by comparing kNN model with metric learning against kNN model without

2.4 Method: DeepSets

2.4.1 Motivation

Classical machine learning methods for classification, such as logistic regression, SVMs, and k-nearest neighbors cannot usually capture non-linearities unless you apply feature engineering or kernels. Even in those cases, subject matter expertise is often required to determine potential non-linearities to consider.

By applying deep and wide feedforward neural networks, we can create a model which trains the best non-linear function to apply for the problem based on the data and loss function, without having to do a lot of feature engineering. It will capture the best feature interactions to solve the problem at hand, which in our case is to apply binary classification to stars in the vicinity of a known group of stellar stream stars.

To do this, we take inspiration from DeepSets [15]. In their paper they present methods on how to design neural networks which take unordered sets as input. For our use case, the network will take as input a set of known stream stars as a reference set in addition to a star whose class is unknown and we wish to understand whether or not it belongs to a stellar stream.

To best incorporate the interactions between the reference points and the points we want to classify, we arrived at an architecture where each reference point is concatenated with the classification point. Each point, both reference and target, are defined by 7 features: ra, dec, pmra, pmdec, g, g_rp, and g_bp. The architecture, shown in Figure 3, is designed to calculate a similarity score between a reference star and the star to be classified through the feedforward and projection, and then take the average similarity over all reference stars to get the probability of the star being part of the stellar stream. The idea is that the feedforward network will train to recognize similarity between reference stream stars and other stars, and that this similarity calculation will generalize to new unseen streams. For this type of network we want to look at three hypotheses; whether this type of network separates

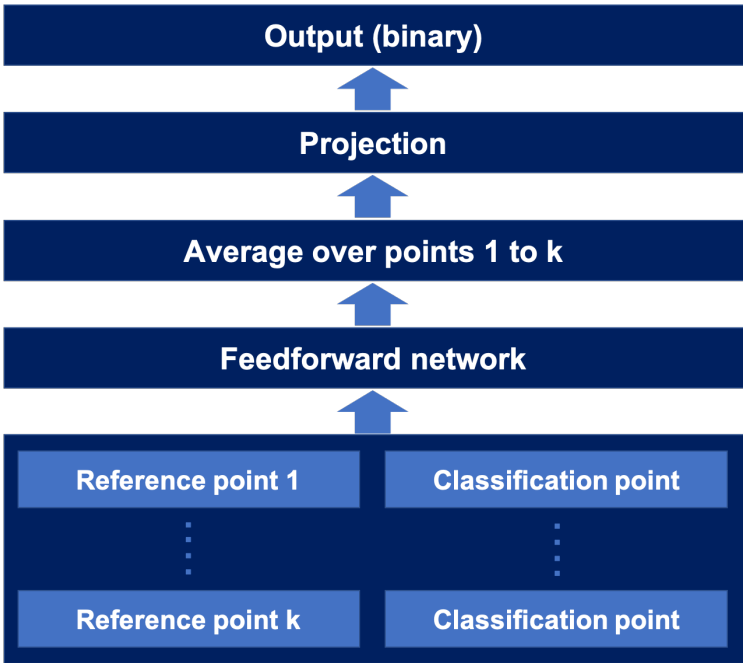

Figure 3: Schematic of the DeepSet architecture used for binary stellar stream classification using a reference set of known stream stars.

stellar stream stars from background noise stars; whether the type of similarity model we are training generalizes to new unseen streams; and whether pretraining on simulated streams with background noise from a different source make the performance on real streams better.

2.4.2 Procedure

To pretrain the model, 49 simulated streams with added background noise were split into 40 training streams and 9 validation streams. For each stream, 10% of the stream stars were sampled as reference stars and another 10% were sampled as training stars. To complete the set for each stream, an equivalent number of background stars were sampled to create an even split. This led to a training set of $\sim$6000 stars and a validation set of $\sim$2000 stars.

Because of the variable size of the reference set for each stream a batch size of 1 was used for training the networks, i.e. pure stochastic gradient descent. The optimizer used was Adam [11], and the loss function used was cross entropy loss on the binary probability output of the model.

To find the best hyperparameter setup for both the model and optimizing the training, random hyperparameter search [1] was run over 100 parameter permutations and 10 epochs for each permutation. The parameters tuned were:

- Hidden size: The hidden size of the feedforward network, range of 5 to 100.
- Number of hidden layers, range of 2 to 15.
- Dropout: Applied to all hidden layers, range of 0.1 to 0.3.
- Learning rate, log_{10}-range of -5 to -3.

The best parameter permutation was selected by validation AUC, and was found to be a hidden size of 50, 9 hidden layers, a dropout of 0.1, and a learning rate of 0.0001. The model was then retrained with this parameter for 30 epochs, with the best model selected with early stopping of validation AUC, with the best AUC found to be 0.95.

This pretrained model was then finetuned on the GD-1 data set. The data set was filtered based on a precalculated isochrone filter mask, with the remaining size being 1979 stream stars and 505,945 background stars. For finetuning and evaluation on the GD-1 data set, the data set was first split into a

10%-90% training-test split randomly with maintained class balance (background to stream star ratio, SNR^{-1}) of $\sim$250 in both sets. We chose to not go with a validation set as well since the training set is so small in the first place, and decided to do no early stopping. The training set's positive class stars were split into 50% reference points and 50% training points, and negative class noise stars were sampled from the original training set at various SNR^{-1}. For the test set, the reference set was set to be the whole set of stream stars in the training set, and the rest of the set was used as test points. A visualisation of this split can be found in 4.

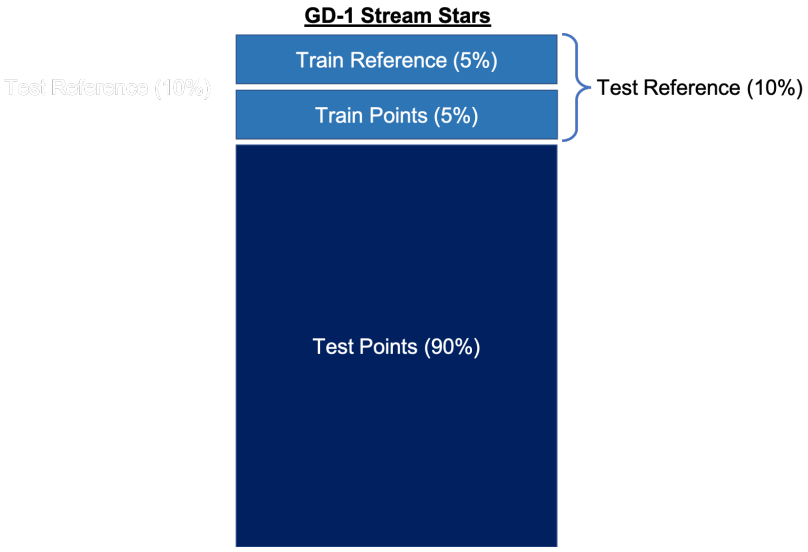

Figure 4: Diagram of the train and test split of GD-1 stream stars. Background noise was sampled at various SNR^{-1} for the training set and the SNR^{-1} for the test set was approximately $\sim$250.

Since a validation set was not used, training and finetuning on the GD-1 set was performed over 30 epochs without early stopping and evaluated only after training was completed.

Two different methods were compared: Finetuning the pretrained model on the GD-1 training set, and starting with a randomly initialized model from scratch and training it on the GD-1 training set.

3 Results

3.1 Results: ESkNN

After obtaining our baseline ESkNN results, we experimented with a few hyperparameters to evaluate their influence on the average F1 score across ten stars.

We first explored two hypothesis spaces for selecting the 25 models to vote on. The first was using low values for "k" nearest neighbors i.e. $1 \leq k \leq 5$ while the latter explored higher values of k i.e. $k > 5$. There was no difference in the F1 scores observed on Group B stars, as seen in Table 1.

Hypothesis Space	F1 (Group B Stars)
low k for kNN	0.606
high k for kNN	0.589

Table 1: Comparison of F1 results from Different Hypothesis Spaces.

We next explored the number of models being used to vote in the ensemble. In general, using fewer voters tends to provide a better F1 score. However, using very few models gives us less flexibility in tuning for false positives and false negatives. Using more models specifically gives us robustness to false positives, which is valuable for certain use cases. Results can be found in Table 2

Number of Voting Models	F1 (Group B Stars)
1	0.754
5	0.741
15	0.607
25	0.606
75	0.629

Table 2: Comparison of F1 Scores when Number of Voting Members are Varied.

We proceeded to convert our coordinate system from the angles that were provided in the declination and the right ascension axis to Cartesian coordinates. Similarly, the proper motion provided in the declination and right ascension directions were converted to velocities in the x,y,and z axes. The conversion yielded no improvement in results, as seen in Table 3.

Coordinate System	F1 (Group B Stars)
Angular	0.606
Cartesian	0.597

Table 3: Comparison of F1 Results with Cartesian and Angular Feature Representations.

We report below the performance of the ESkNN model with 25 voting models as well as the performance of those models when predicting the class of the GD-1 data set. Additionally, we retrain our ESkNN model on 10% of the GD-1 stars and report the performance of those newly selected models on GD-1. Results can be seen in Table 4.

Methodology	Training Set	F1 (Group B)	F1 (GD-1)	AUC (GD-1)
ESkNN	Group A stars	0.606	0.325	0.705
ESkNN	GD-1 (10%)	Not Evaluated	0.731	0.807

Table 4: Comparison of F1 Results with Model Trained on Simulated Streams vs. Trained on GD-1.

As we increase the percentage of votes (threshold) required to classify a star as belonging to a stellar stream, we expect the precision to be higher and recall to decrease as we gain higher confidence in our results and risk not identifying some of the stars. A graph of the trade-off is shown in Figure 5. We report our results above using the F1-score as an evaluation metric.

3.2 Results: Metric Learning

The results of the metric learning, trained on stream to noise ratios of 1:1, 1:10, 1:20, 1:50, and 1:70 are shown in Table 5. Improved results on the kNN without metric learning are shown in green while worse results are shown in red.

It appears that the effects of the metric learning vary from stream to stream. Arguably, the 1:1 and 1:10 are worst implementations, with kNN results on five streams performing worse than the regular kNN. The 1:70 benefits the most from metric learning, with improved kNN results on seven streams. Overall, there does seem to be improvement as the metric training ratio increases, but the results are inconsistent on a stream to stream basis. For example, on stream 3526, only ratios of 1:1 and 1:10 experience improvement in results, but the results are the opposite for stream 4682 - all streams experience improved results except for the 1:1 and 1:10 ratios.

3.3 Results: DeepSets

For both training from scratch and finetuning a pretrained model on GD-1, 7 SNR^{-1} were tried: 1, 5, 10, 20, 30, 50, 70. It was found that the recall vs. precision tradeoff that every model has can be tuned decently well by varying the SNR^{-1} in the training set. In Table 6 and 7, the values of the best F1 score (depending on threshold) and the AUC on the GD-1 test set for both methods can be found

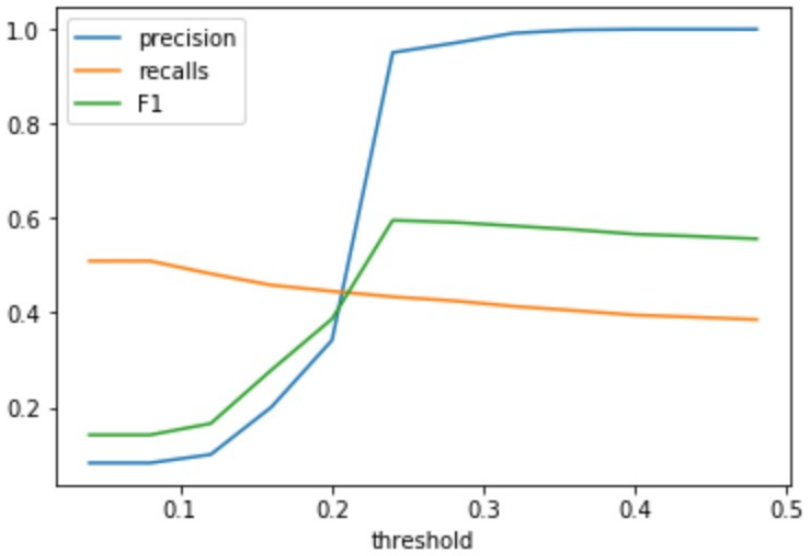

Figure 5: Precision Recall Tradeoff of ESkNN Model.

		F1 (SNR^{-1})				
Mock Stream No.	**No metric**	**1**	**10**	**20**	**50**	**70**
178	0.400	0.372	0.311	0.400	0.471	0.457
9528	0.222	0.333	0.348	0.444	0.444	0.303
1954	0.339	0.256	0.074	0.333	0.216	0.286
2408	0.400	0.400	0.571	0.429	0.444	0.200
4468	0.184	0.275	0.231	0.278	0.305	0.243
4682	0.382	0.304	0.378	0.494	0.426	0.395
5402	0.491	0.382	0.629	0.389	0.623	0.528
3526	0.344	0.483	0.452	0.182	0.211	0.320
9164	0.505	0.632	0.432	0.566	0.667	0.750
2985	0.472	0.442	0.418	0.466	0.457	0.558

Table 5: Metric learning results.

respectively. It is clear that using the pretrained models trained on the simulated streams is clearly better than just training the model from scratch.

SNR^{-1}	**Finetuned pretrained model**	**Trained from scratch**
1	0.448	0.379
5	0.482	0.36
10	0.527	0.379
20	0.618	0.356
30	0.635	0.388
50	0.657	0.37
70	0.582	0.337

Table 6: Best test F1 scores for both the finetuned pretrained models and the models trained from scratch for different SNR^{-1} in the training set.

To illustrate the flexibility of the models in terms of balancing precision and recall, we plot the recall vs precision curves for the models trained with each SNR^{-1}. In Figure 6 and 7 the recall vs precision curves for the finetuned models and the trained from scratch models can be found for

SNR^{-1}	Finetuned pretrained model	Trained from scratch
1	0.994	0.991
5	0.995	0.990
10	0.996	0.991
20	0.997	0.990
30	0.998	0.992
50	0.998	0.993
70	0.997	0.990

Table 7: Test AUC scores for both the finetuned pretrained models and the models trained from scratch for different SNR^{-1} in the training set.

different SNR^{-1}. Depending on how the model should be used, it may be preferable to use different training SNR^{-1} and classification thresholds in different situations.

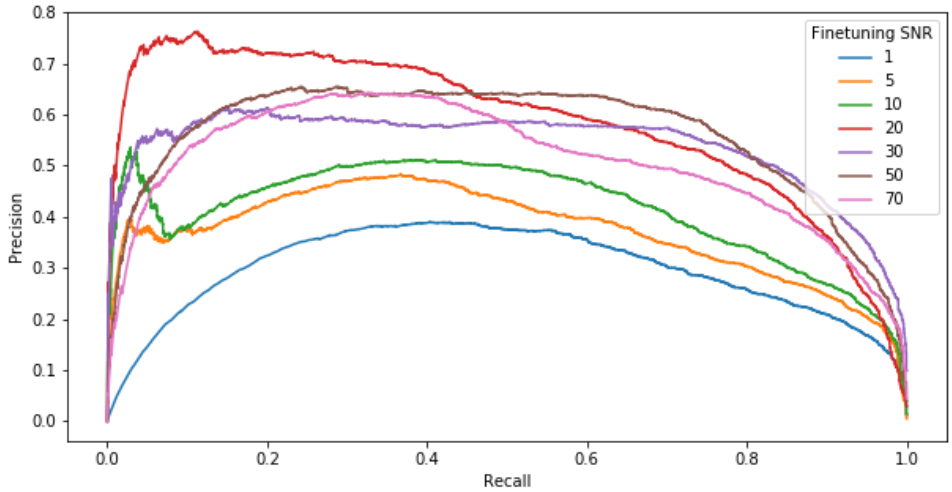

Figure 6: Recall vs precision curves for different SNR^{-1}s for pretrained models finetuned on the GD-1 training set. Note: Legend should say SNR^{-1}.

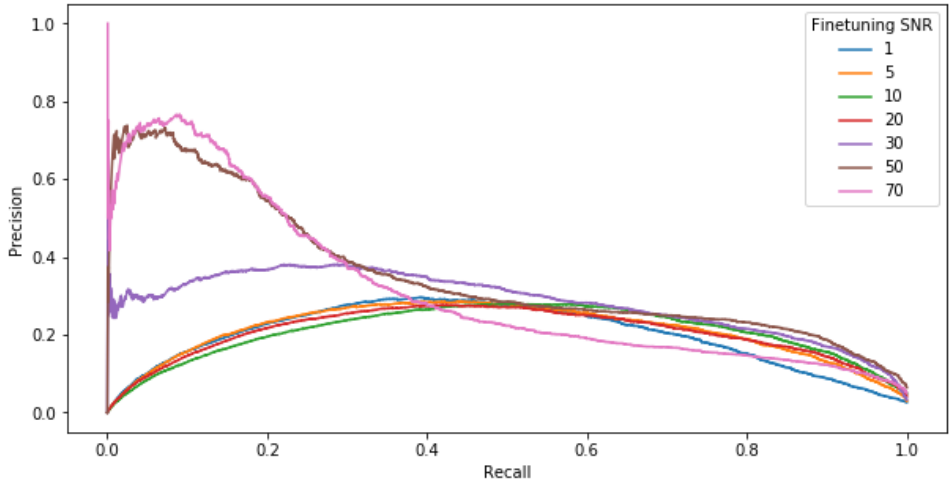

Figure 7: Recall vs precision curves for different SNR^{-1} for models trained only on the GD-1 training set without pretraining. Note: Legend should say Training SNR^{-1}.

The recall vs. precision curves are created by calculating precision and recall for different probability thresholds for classifying a data point as the positive class. To display this, we can in Figure 8 see the precision, recall, and F1 score plotted against threshold for our best finetuned model by F1 score, with $\text{SNR}^{-1} = 50$.

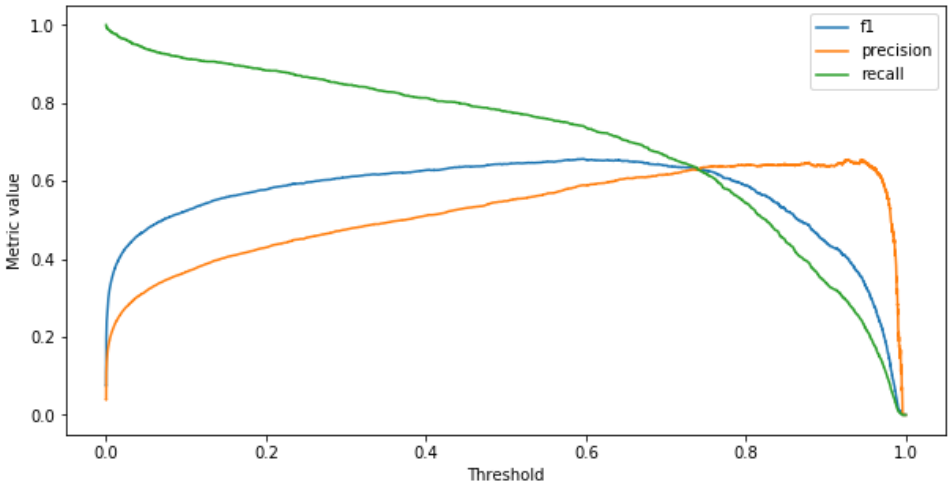

Figure 8: Precision, recall, and F1 score plotted against classification threshold for the best model found, a finetuned model with an SNR^{-1} of 50.

4 Conclusion and Future Work

4.1 Conclusion

With knowledge of approximately 10 percent of the stars in the GD-1 stream, the EsKNN method could accurately predict the class of the remaining stars in a given geographical region with a precision of 99% and a recall of 58 %. With this approach, we can be confident that the stars we predict to be part of a stellar stream are actually part of a stellar stream.

With a similar knowledge of 10% of the stars in the GD-1 stream, the DeepSets method yielded lower precisions but higher recalls of 80% or higher depending on the tuning. With this approach, we have less confidence in our prediction but are more satisfied that we have captured a majority of the eligible candidates.

Each approach can be valuable depending on the use case. In addition to the different precision-recall balance for the two types of models, both of them can be tuned to prioritize precision or recall by changing classification thresholds or changing the training procedure. A major area of exploration is to establish whether or not there are substructures in the stars that we classify as belonging to a given stream. Basically, we'd like to know if the stars that have the highest likelihood to belong to a given stellar stream share additional properties. To answer this, a high recall approach such as the DeepSets method would be preferred. In the second use case where we wish to explore a given space and construct a stellar stream given some of its members, a high precision approach such as ESkNN would be preferred. In addition to this, in the event of a new stream discovery, the high precision method can be used to verify the stars that belong to it.

4.2 Future Work

Although the DeepSets methodology tried to capture non-linearities in our data set, the ESkNN method provided the superior ability to classify stars from new clusters. We would like to explore this further in the unsupervised case. Given the performance of ensemble methods in the supervised case, we hypothesize that this approach may generalize to the unsupervised case. Combining the results from different approaches e.g. k-Means, DBSCAN, HDBSCAN (or potentially multiple models from the same approach) may yield better results than simply using one clustering algorithm.

In addition, the consistently high precision from the ESkNN could be married with the high recall method from DeepSets in a cascade model that better refines true positives. This is a problem that some members of the team will continue investigating in 2020.

5 Acknowledgements

Over the course of our project, we were happy to have worked with and received guidance from our mentors Shirley Ho, Gabriella Contardo, Miles Cranmer, and Adrian Price-Wheeler from the Flatiron Institute Center for Computational Astrophysics.

We would also like to thank the NYU Center for Data Science faculty and Capstone classroom leaders for their feedback and guidance throughout the project. It has been a wonderful opportunity to work on this project for the past months.

References

[1] BERGSTRA, J., AND BENGIO, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res. 13* (Feb. 2012), 281–305.

[2] BONACA, A., ET AL. The spur and the gap in GD-1: Dynamical evidence for a dark substructure in the milky way halo.

[3] BORSATO, N. W., ET AL. Identifying stellar streams in gaia DR2 with data mining techniques.

[4] BOVY, J., ET AL. The shape and inner milky way halo from observations of the PAL 5 and GD-1 stellar streams. *The Astrophysical Journal 833*, 1 (2016), 31–45.

[5] CAMPELLO, R., ET AL. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining*, J. Pei et al., Eds., vol. 7819 of *Lecture Notes in Computer Science*. Springer, 2013, pp. 160–172.

[6] ESTER, M., KRIEGEL, H.-P., SANDER, J., AND XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press, pp. 226–231.

[7] GRILLMAIR, C. J., AND CARLIN, J. L. Stellar streams and clouds in the galactic halo. In *Tidal Streams in the Local Group and Beyond*, H. J. Newberg and J. L. Carlin, Eds., vol. 420 of *Astrophysics and Space Science Library*. Springer, 2016, ch. 4, pp. 87–112.

[8] HAMBLY, N., ET AL. Datamodel description. In *Gaia Data Release 2*. European Space Agency (ESA) and Gaia Data Processing and Analysis Consortium (DPAC), 2019, pp. 521–576.

[9] HUGHES, M. E., ET AL. Fossil stellar streams and their globular cluster populations in the E-MOSAICS simulations.

[10] IBATA, R., AND GIBSON, B. The ghosts of galaxies past. *Scientific American 296* (2007), 40–45.

[11] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014).

[12] MALHAN, K., AND IBATA, R. A. Streamfinder i: A new algorithm for detecting stellar streams.

[13] TORRES, S., ET AL. Gaia DR2 white dwarfs in the hercules stream. *Astronomy and Astrophysics 629* (2019).

[14] WEINBERGER, K. Q., AND TESAURO, G. Metric learning for kernel regression. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, M. Meila and X. Shen, Eds., vol. 2 of *Proceedings of Machine Learning Research*. PMLR, San Juan, Puerto Rico, 21–24 Mar 2007, pp. 612–619.

[15] ZAHEER, M., KOTTUR, S., RAVANBAKHSH, S., POCZOS, B., SALAKHUTDINOV, R. R., AND SMOLA, A. J. Deep sets. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 3391–3401.