1 Introduction [15 points]

- Group members:
  1. Sri Aditya Deevi
  2. Palak Purohit
  3. Princekumar Kothadiya
- Kaggle team name: **Life Search**
- Ranking on the private leaderboard: 19
- F_2 score on the private leaderboard: **0.84993**
- Colab link: https://colab.research.google.com/drive/1teQnXMrI0GGUs2iR-9j2TVrb_oJ7jJp5?usp=sharing
- Piazza link: https://piazza.com/class/lbv0docn6037fw/post/390
- Division of labor:
  - Data Preprocessing* : Sri Aditya Deevi, Palak Purohit and Princekumar Kothadiya
  - New Feature Definitions* : Sri Aditya Deevi, Palak Purohit and Princekumar Kothadiya
  - SVM + Random Forest (With Sample Weighting) * : Sri Aditya Deevi, Palak Purohit
  - Decision Tree with Adaboost * : Sri Aditya Deevi, Princekumar Kothadiya
  - Ensembling Method for Models : Palak Purohit
  - Dealing with Normalization * : Sri Aditya Deevi, Palak Purohit
  - LSTM based Models and Features : Sri Aditya Deevi
  - Other DNN based Models : Palak Purohit
  - TSNE Visualizations : Princekumar Kothadiya
  - Other ML Approaches and Ideas* : Sri Aditya Deevi, Palak Purohit and Princekumar Kothadiya
  - Report and Colab * : Sri Aditya Deevi, Palak Purohit and Princekumar Kothadiya

*Equal Contributions

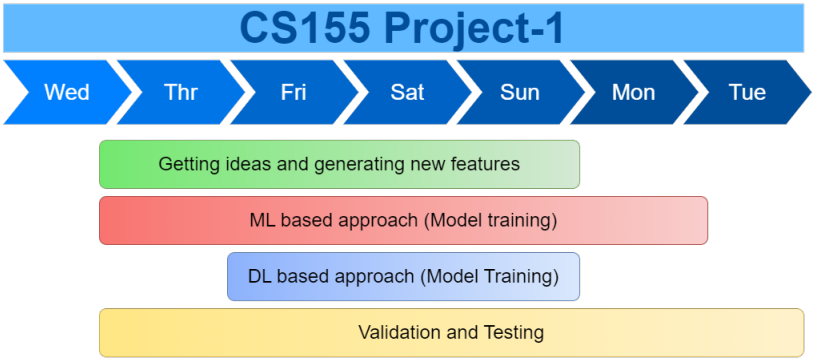
2 Overview [15 points]

Models and techniques attempted

The following is a list containing brief description of the methods we attempted *:

1. SVM on normalized given labelled data (Train-Test split : 80%-20%), with no new features.
2. Random Forest normalized given labelled data (Train-Test split : 80%-20%).
3. SVM + Random Forest normalized given labelled data (Train-Test split : 80%-20%).
4. SVM + Random Forest normalized given labelled data (Train-Test split : 80%-20%) - with Sample Weighting (more weight for lab data)
5. SVM + Random Forest normalized given labelled data (Train-Test split : 80%-20%) - with Sample Weighting (more weight for lab data) and class weighting (more weight for class 1)
6. Decision Tree with Adaboost on only Lab Data
7. Stacked BiLSTM (with BucketData Iterator) on Normalized Raw Data sequences
8. Stacked BiLSTM (with BucketData Iterator) on Normalized Raw Data sequences with Sample Weighted Loss (More preference to Lab Data)
9. SVM + Random Forest normalized given labelled data (Train-Test split : 80%-20%) - with New Features and Normalized LSTM model features.
10. Deep Neural Network on normalized given labelled data
11. Finally, various *ensembles* or combinations of different methods.

Work timeline



*New Features are considered unless specified

3 Approach [20 points]

Data exploration, processing and manipulation

We first visualized the data using T-SNE projection for the provided features. This motivated us to generate more distinguishing features. We retained the given features as they were important. We observed that motile tracks are non-linear and more random, as compared to non-motile ones. So, in order to help the model distinguish these behaviours, we added acceleration (mean & standard deviation), absolute value of pearson coefficient, angular speed (mean & standard deviation), standard deviation of co-ordinates X and Y, error from linear predicted trajectory (mean & standard deviation) as new manually designed features. We also added an lab sample indicator feature to make the model differentiate between real and synthetic data. For DL methods, we considered the actual raw time series data. To demonstrate the efficacy of the newly added features we have generated T-SNE projection maps (See Fig. 1 and Fig. 2) and feature importance maps (See Fig. 3) and (See Fig. 4).

Overall Process : Details of Models and Techniques

We first considered a normalized version of the data with the given features, trained (and tested) an SVM model on it, to get a baseline performer. Then we felt the need to generate more features for better classification and we also decided to utilize ensembles of models for better generalization. So, we trained a Random Forest Algorithm (RF) and combined its predictions with that of the SVM algorithm. We observed that there was a big gap between the test split performance and kaggle test set performance. Later, we recognized that this discrepancy was caused due to the models overfitting on the large proportion of synthetic data in the train set which was absent in the kaggle test set. In order to combat this issue, we considered two strategies. First, was to use sample weighting for both the SVM and RF models, thereby giving more importance to lab data classfication. Second, we Decision Tree model with Adaboost on a reduced set of features and only lab data so that the Adaboost's criteria of being a weak classifier is satisfied. In addition to this, we recognized there might be some loss of information during data normalization, so we also trained another RF algorithm trained on unnormalized data with all the newly added features. All of these strategies lead to performance boosts on the kaggle test set, so finally we considered an ensemble of the above-mentioned models.

Parallely, we also explored Deep Learning Approaches as they are *generally* good at automatically creating effective hidden representations. As the data was time series the natural choice was to use RNNs. In particular, we used a stacked, bidirectional and stateless LSTM network to capture good distinguishing features in the data useful for classification. The method was made efficient and effective by processing sequences parallely using Bucket Data Sampler. But as we know DL models are data hungry and the small proportion of lab data in the training set led to overfitting. For the fully connected network, to solve the task at hand more effectively, we were using F2 score as a metric for backpropagation. The model overfitted on the simulated data and hence gave a poor performance on the lab data. On reducing the depth of the network, the performance further decreased.

So, we concluded that for this specific dataset DL is not the way to go. We also tried other techniques such as class weighting and Gradient Boosting etc in the ML regime but these techniques did not lead to good models, so we did not consider them in our final ensemble.

4 Model Selection [20 points]

Scoring

Model	Score on Test Set	Only Lab Data	Optimum Hyperparams
SVM	0.99578	0.96153	kernel = rbf; C = 10
RF#1	0.99706	0.92843	n_est = 100; max_depth = 10; min_samp_leaf = 5
RF#2	0.9963	0.92412	n_est = 100; max_depth = 10; min_samp_leaf = 5
AdaBoost + DT	0.88785	0.9732	n_est = 50; max_depth = 6; min_samp_leaf = 5; lr = 0.1

Table 1

We used an ensembling strategy where, given all the classifiers in the ensemble, we considered :

$$\text{Final Predictions} = \max\left(Clf_1(\mathbf{x}), Clf_2\mathbf{x}), \ldots, Clf_N(\mathbf{x})\right)$$

This is because getting False Negatives is bad as compared to False Positives for getting a high F2 score. The best performing model was an ensemble of the following:

(i) SVM – Trained on Normalized Stratified (based on Lab and labels) data with sample weighting. (Slightly Overfit on Sim Data but learnt to separate motile from non-motile)

(ii) Random Forest #1 – Trained on Normalized Stratified (based on Lab and labels) data with sample weighting. (Slightly Overfit on Sim Data but learnt to separate motile from non-motile)

(iii) Decision Tree with Adaboost – Trained on Normalized Lab Data and reduced feature set (So that satisfies Adaboost criteria of being a weak classifier) (Good Generalization to Lab Data but performance was slightly inferior due to very less data).

(iv) Random Forest #2 – Trained on Unnormalized Stratified (based on Lab and labels) data with sample weighting. (Slightly Overfit on Sim Data but captured the information lost in normalization)

Validation and test

For SVM and Random Forest models, we split the whole labelled data into an (80% - 20%) train-test split. For Decision Tree models, we considered only labelled lab data and divided it into an (80% - 20%) train-test split. We used cross-validation within the train-split to determine the best hyperparameters for all the models in the final ensemble. For all the models in our ensemble, we checked the performance on the whole lab data and used that as a proxy for the kaggle test score.

5 Conclusion [20 points]

Insights

We determined the features that were important exploratory data analysis, visualizations and sanity checks by training and testing the various ML models considered. T-SNE plots and Feature Importance maps also were indicative of the effectiveness of various features. The top features can be Mean step speed, Std. Dev. of Step Speed, Mean step acceleration, Std. Dev. of Step acceleration, Absolute Value of Pearson Coefficient, Std. Dev. of X, Std Dev. of Y, Mean angular speed, Std. Dev. of angular speed, track length, e2e distance and duration. We corrected some features that negatively affected the performance as follows:

- Replaced Pearson Coefficient with its absolute as we only care if there is a linear correlation or not.
- Replaced Mean of Trajectory Slopes with its standard deviation as it is more relevant for finding the non-linearity.
- Some other features such as angular displacement, squared displacement error (mean and variance) etc. did not aid in improving performance so we discarded them.

The project was really interesting and we learnt a lot in the process. The following are some of our key takeaways:

- Overfitting can happen in multiple ways. In this project, one major issue was dealing with the overfitting on synthetic data. This was different from the typical overfitting.
- Boosting methods such as Adaboost can significantly boost the performance of **weak** classifier. These methods tend to have a very good generalization performance.
- Manual Feature Design can take us a long way, especially in low data regimes. But this domain-knowledge intensive, requires a lot of thinking and trial-error cycles to master.
- Deep Learning is not the solution to all problems. In low data regimes (lab data) such as this, DL models tend to overfit because of their data hungry nature. This was seen in our LSTM and other DNN experiments.

Challenges

The main challenges we faced are:

- Dealing with imbalance of Real and Synthetic Data (Overfitting associated)
- Effective Feature Engineering
- Dealing with Class Imbalance
- Limited Submissions on Kaggle

6 Extra Credit [5 points]

T-SNE Projection Maps

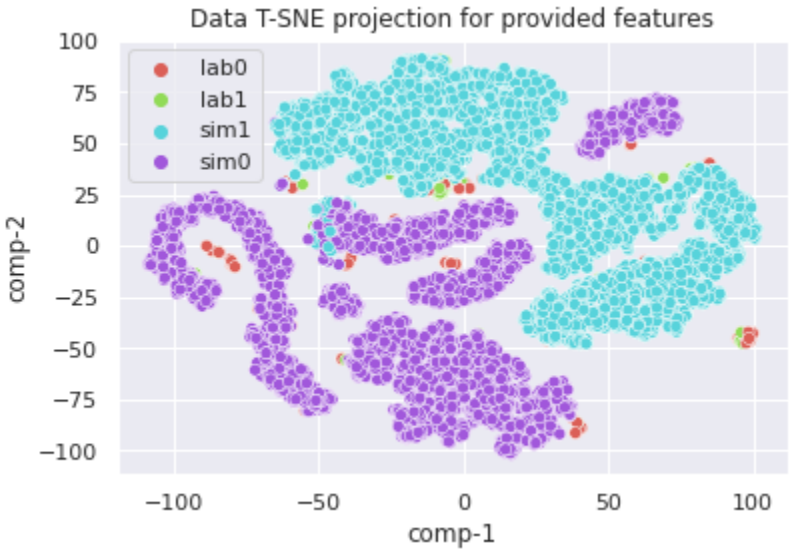


Figure 1: Data is not well separated with the provided features and not grouped (especially lab data) that implies the features are not enough to get the good decision boundary and classification

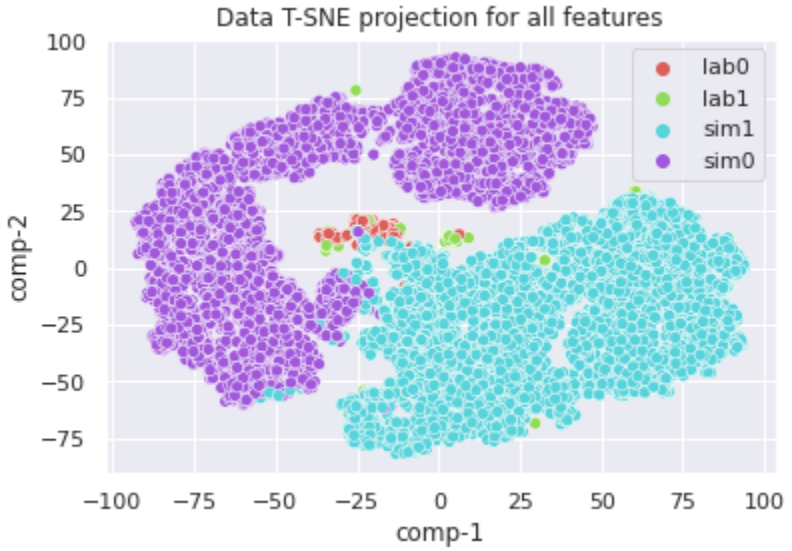


Figure 2: The newly generated features combining with provided features are much more separated than previous and also grouped together which will help to perform classification task better

Feature Importance Maps

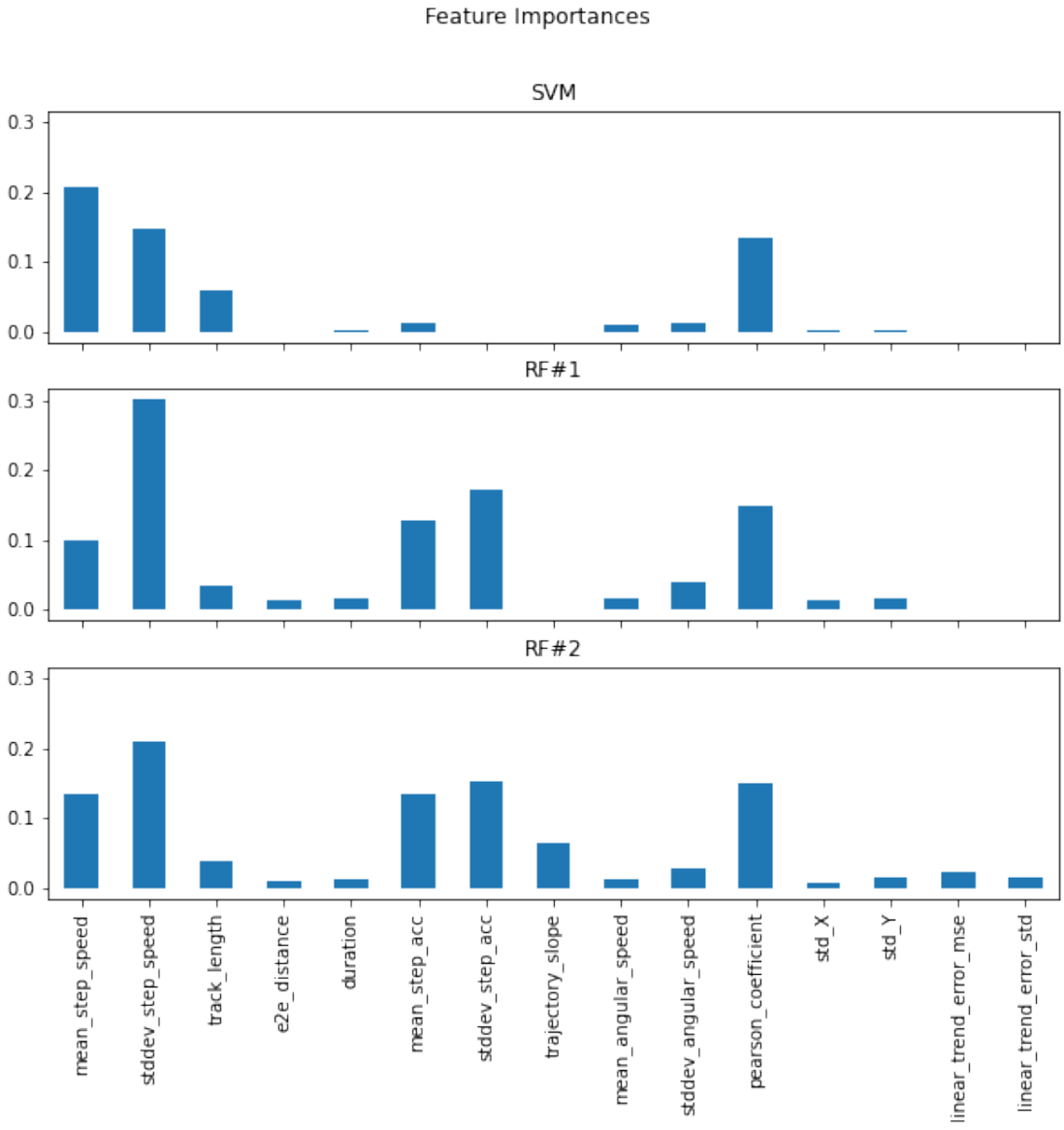


Figure 3: Feature importance map of SVM, RF and RF(unnormalized) with full features

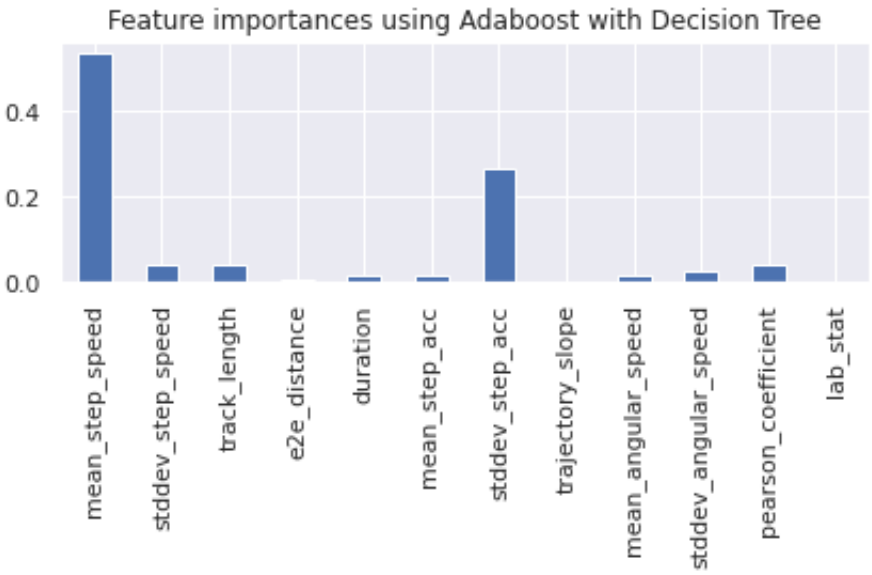


Figure 4: Feature importance map of Adaboost with Decision Tree with reduced features

Other Insights

We summarize a list of our additional insights here:

(i) **Kaggle Competition Metric**

F_2 score was rightly chosen the competition metric because in this task, the False Negatives should be penalized more than the False Positives because of the fact that declaring motile sample as non-motile can be costly.

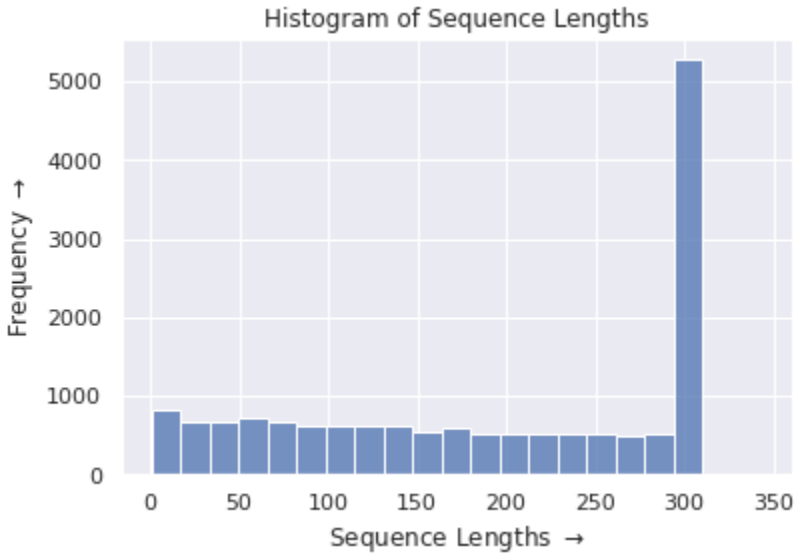
(ii) **LSTM based Models : Bucket Data Sampler**

The sequence length associated with the raw data timeseries was highly variable (See Figure below) and so dropping all such sequences would reduce the volume of dataset causing the model to overfit.

Also padding would add a lot of noise to the data possibly leading to low performance. Passing sequences one by one in a stateless LSTM is possible but we would lose the benefits of vectorization. So we used Bucked Data Sampler, which groups sequences of similar length together and pads them minimally to form batches. This gives the best of both worlds.

(iii) **Fully Connected DNN** Implementation details: The neural network was built using Keras framework and had a total of 151 trainable parameters (15 dense layers with relu activation and finally a sigmoid for the output class probabilities). We were using an Adam optimizer and a batch size of 32 for the updates.

(iv) **More Ideas**



One more idea we had in mind but couldn't get the time to implement was to generate features using an LSTM autoencoder trained with weighting mean square error giving more importance to lab data. This could extracted better features and augmented well with our manually designed features.

Challenges (More Details)

Here we present a more detailed version of the challenges we faced:

- On observing the simulated and real-world data carefully, we noticed that they have a lot of differences in their distribution. Moreover, the abundance of simulated data in the train set and the lack of lab data makes it even more challenging to make the model learn the characteristics of the real world data, without overfitting. To combat this issue, we tried several strategies such as using sample weighting, adding features that differentiate between simulated data and lab data, training on only the lab data and so on. These approaches did alleviate the problem to a certain extent.
- Another issue was the imbalance in terms of the 2 classes. Simply training the model without looking into this issue would yield a high accuracy but not a high F2 score. To account for this imbalance, we used stratified sampling to split the test and train data.
- Feature engineering was another crucial challenge. Given the raw data, it took a series of visualizations and experiments in order to get to the features which were very useful in separating the 2 classes. The model performance boosted on introducing the right features.
- Limited submissions on Kaggle were also a challenge since we had to compare models qualitatively and quantitatively before submitting. This was actually helpful in giving us the experience of a real world setting in which the test data is not readily available and so it isn't possible to tweak the slightest parameters and test again in order to overfit on the test data and increase the score.