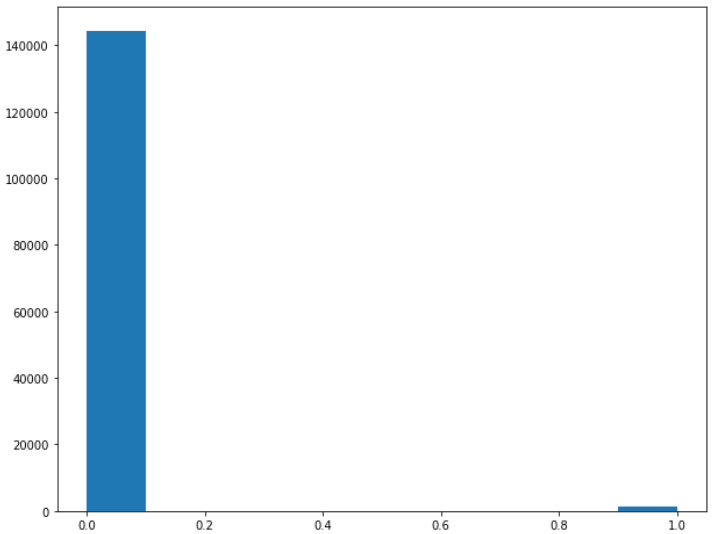
CS5011 – Assignment
June 2019

Introduction: The goal of protein homology prediction task in KDD Cup 2004 is to predict which proteins in database are homologous to a native (query) sequence. Each database protein sequence is described by 74 features that measure the similarity between the database protein sequence and the query sequence.
Protein structure play an important role in biological functions. Experimental approach to protein structure determination is both slow and time-consuming. Protein homology prediction is a key step of protein structure prediction. The objective of the problem is to find those proteins in the database that are homologous to the query protein so that homologous proteins can be used as structural templates.
There are 153 queries in the train set, and 150 queries in the test set. For each query there are about 1000 returned documents, only a few of the documents are correct matches for each query. The goal is to predict which of the 1000 documents best match the query based on 74 attributes that measure match. A good set of match predictions will rank the "homologous" documents near the top." (from KDD Cup 2004 website)

Information about the features are available here
Leaderboard ranking mechanism are available here

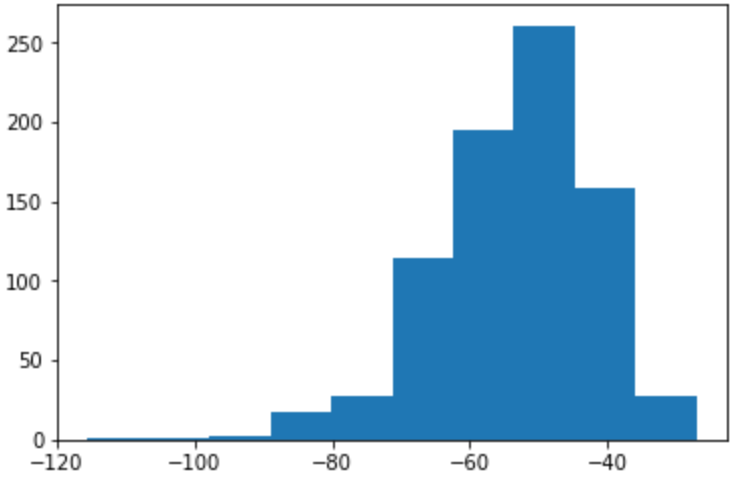
Experiments
Highly skewed data distribution induces class imbalance and in this problem there are more number of negative (non-homologous) samples than positive. The figure below shows the class distribution (0 for non-homologous, 1 for homologous)



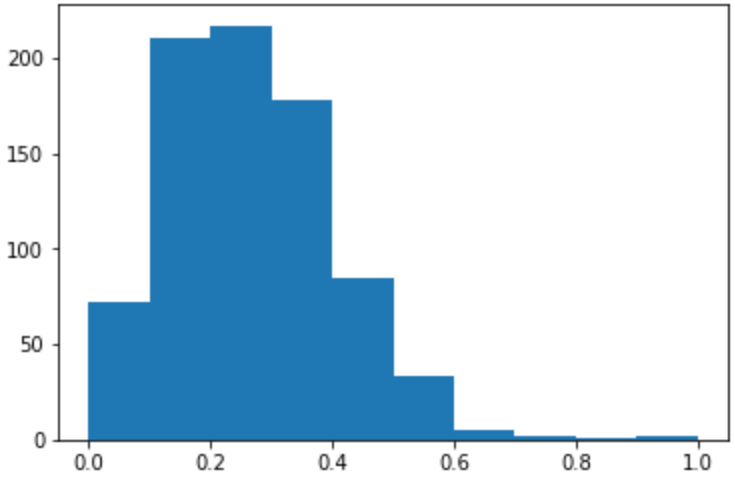
A series of experiments were performed to perform this highly imbalanced binary classification task.

Block structure of the data - normalize within blocks

The training data given can be divided into 153 blocks with approx. 1000 examples each. Since the features have been measured in different scales (as given in the metadata), we have to normalize them in order to extract their discriminatory power. But as it has been pointed out by Ron Elber who provided the data for this challenge, each block is unique in itself [1]. So it's better to do normalization within each block. Similarly, we normalized the test data on a per-block basis.



Block 279 feature 10 before normalization



Block 279 feature 10 after Normalization (0-1)

Experiment 1: Applying SVM on the entire dataset

The first experiment that we did involved applying SVM on the entire dataset with and rbf kernel and the class_weight parameter as ‘balanced’. The predictions on the unseen test set provided were uploaded using the online portal and this fetched a leaderboard **rank of 127.**

Experiment 2 - Downsampling majority class

The second experiment that I did was downsample the majority class (i.e the non-homologous labels). Then a series of nine classifiers (Naive Bayes, ADABoost, Gradient Boost, SVM, Extratrees, Decision trees, Random Forests, Logistic regression

and KNN) were trained and the test results are reported. As is evident from the results in the notebook, the best validation accuracy was around 55%. The best model was also tested using the test data provided for the competition and this fetched a very low **rank of 136.**

Experiment 3- Generate synthetic data for the minority class

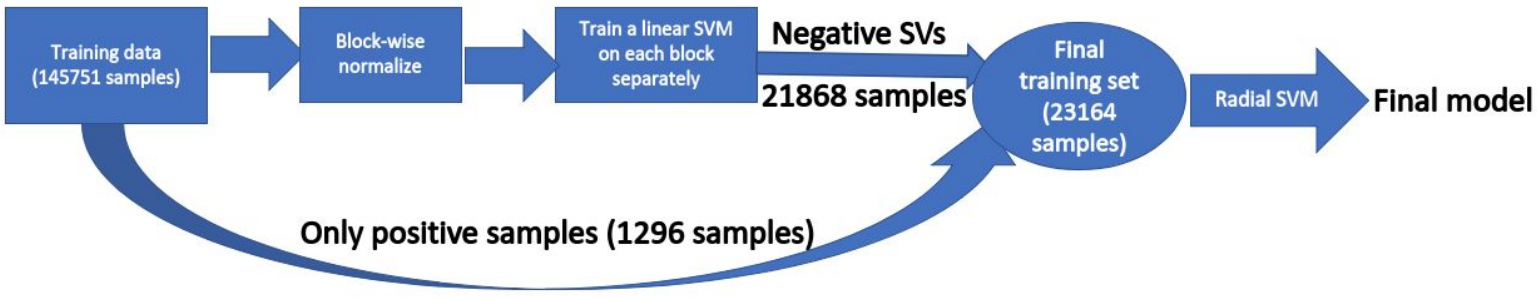
The next experiment was done using SMOTE and the best test accuracy increased to 0.99 using the Extra Trees classifier. The leaderboard **rank of 128** using this method also saw an improvement over the previous one.

Experiment 4 - KmeansSMOTE

K-Means SMOTE is an oversampling method for class-imbalanced data. It aids classification by generating minority class samples in safe and crucial areas of the input space. The method avoids the generation of noise and effectively overcomes imbalances between and within classes. I applied kmeans SMOTE to the entire dataset. The results of nine classifiers (Naive Bayes, ADABoost, Gradient Boost, SVM, Extratrees, Decision trees, Random Forests, Logistic regression and KNN) were trained and the test results are reported (in the jupyter notebook). As expected, due to severe imbalance the recall was low throughout except for decision trees and extra random trees performing the best. The results of the predictions in the unseen data was uploaded into the portal and that fetched a low **rank of 127**

Experiment 5: Sample size reduction for negative class

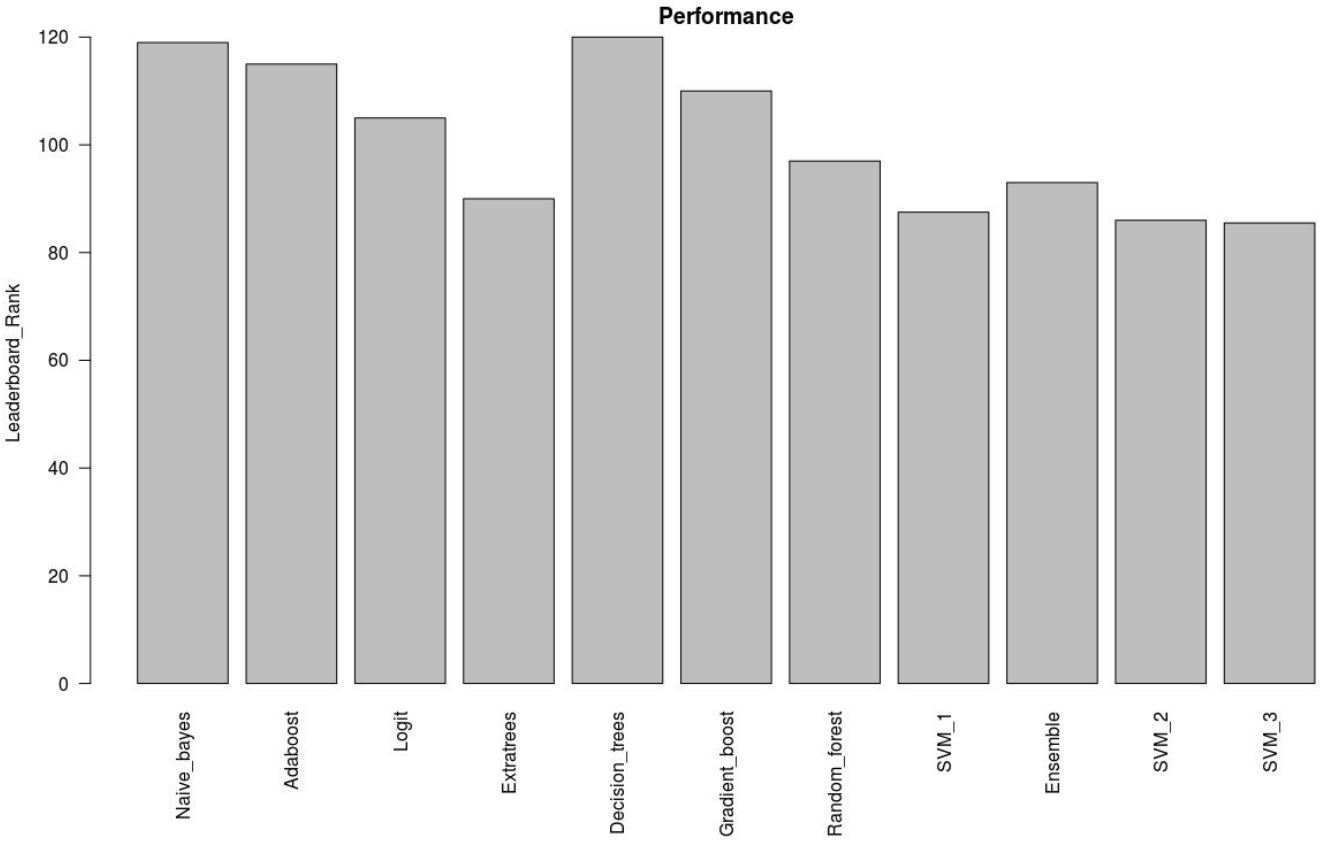
Since there is a severe imbalance in the dataset, we have to try and remove some noisy samples form the negative class that is contributing to the low recall. Due to the block like structure of the dataset, we decided to utilise SVMs to undersample the majority class. First, the intrablock normalized data was taken and SVM with linear kernel was trained on each block separately. The assumption here was that multiple SVMs trained on different subsets of the data would approximately cover most the support vectors that we would get by training an SVM on the entire dataset. So after training on each block separately, the support vectors were extracted from each block and stored. This process was repeated for all the blocks. After that only the negative support vectors were extracted. This would form our negative training set and would be the most informative set of samples. All the other negative samples were removed. All the



positive examples in the initial dataset would make up the positive training set. Now we would train a radial kernel SVM on the reduced dataset and test our predictions.

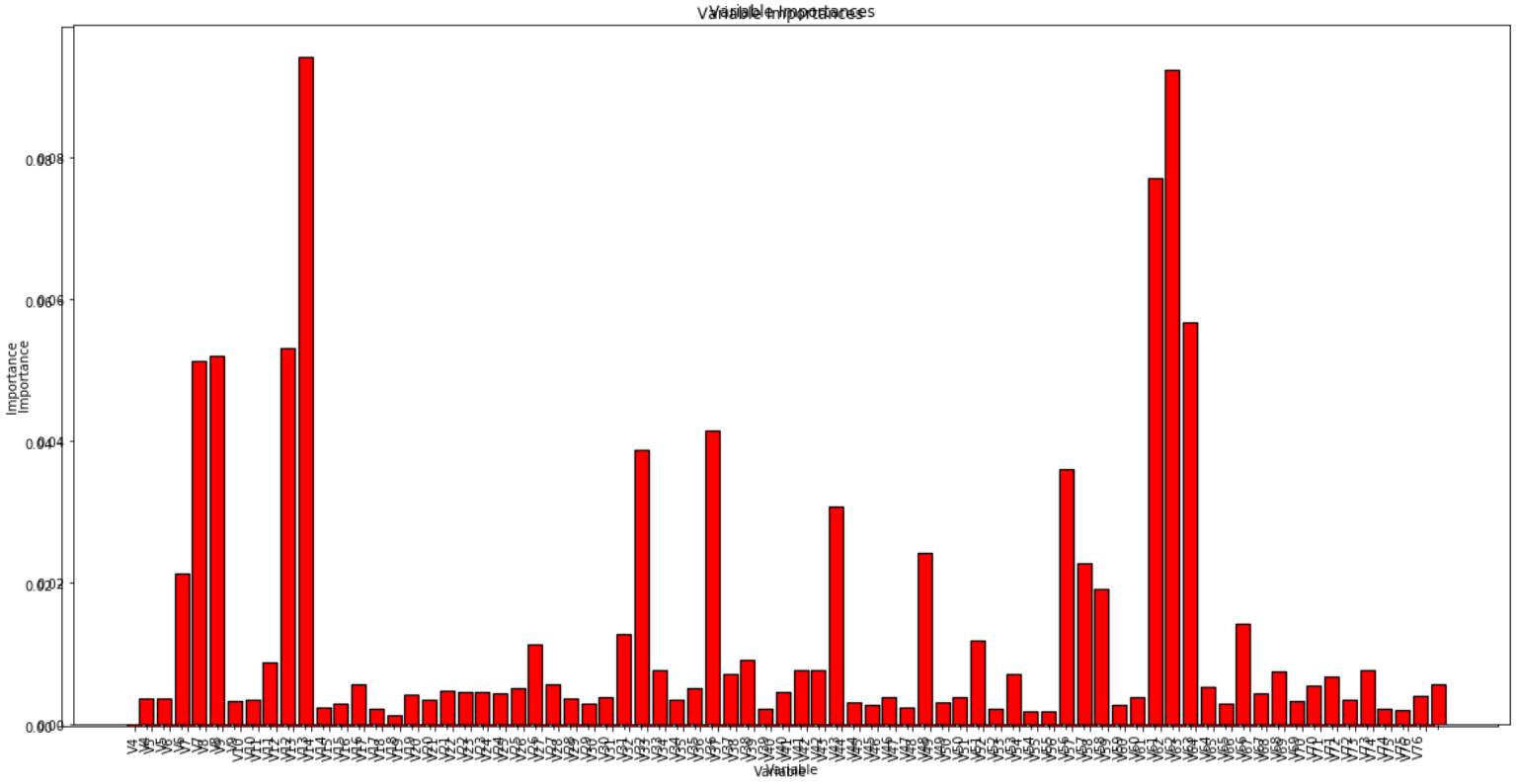
As is evident from the above figure, there are **21,868 SVs** extracted in total by doing blockwise SVM classification using linear kernel. Adding this up with the positive examples yield **23,164** samples in total, a significant reduction from **145751 samples. The final training set was then fed into a Radial SVM and the final model was constructed.**
The testing set data was also normalized blockwise and this model was run on it. The predictions were uploaded to the online portal and the **rank obtained was 87.5**- a significant improvement from the previous experiments.
Since this method was promising enough to look further into, we stopped here and tried to tune the model. The following tasks were then done and the results were reported. The best set of parameters were tuned while doing a 10-fold cross-validation using sklearn's GridsearchCV module. The best set were reported and the model was retrained using these set of parameters. But the results obtained fetched a worse rank in the leaderboard. So it was not considered in the final model.

A list of eleven classifiers were also applied to this blockwise support vector based dataset. Ten fold cross validation was performed and the final model was used to predict the unknown test data.
The rank in the final leaderboard table is summarized in the plot below.



The last three SVMs (SVM_1, SVM_2, SVM_3) were tried by changing the threshold of prediction probabilities. The best score (rank) obtained was using the threshold of 0.35. This fetched a **rank of 85.50.**

As an additional step, we also performed all the previous analysis by doing feature reduction on the new support vector based data of size **23,164**. The following plot shows the relative importance of each of the predictor variables.
The results obtained after removing the uninformative features were not that different from the ones presented above.



Final results and discussion:
The final standing based on the models we produced were **85.50**. The main idea behind **experiment 5** was to minimize the negative effect of information loss and to maximize the positive effect of data cleaning. Since this was a highly imbalanced dataset, there may be many noisy samples and random undersampling might be the first thing that people might do. But this might lead to loss of important samples- the samples that would finally define our decision boundary.

The best approach would have been to individually minimize each metric to attain a better position in the leaderboard, but we went for a single model. **The final model also gave surprisingly good results overall with a second best RMSE of 0.03476 that is within 1% of the RMSE of the winning team.**

159	-	89.0	0.86000	122.0	0.16669	44.0	59.67333	79.0	0.79622	83.500
3244	4	69.5	0.87333	90.0	0.05455	87.0	90.50667	88.0	0.78704	83.625
98	-	107.5	0.80000	87.0	0.05375	38.0	58.92667	103.0	0.73852	83.875
2263	19	106.0	0.80667	78.0	0.04826	91.0	96.32000	67.0	0.80271	85.500
shayantan	51	114.0	0.72000	2.0	0.03476	121.0	470.02000	105.0	0.72416	85.500

Average rank across all metrics (Sort it by Avg rank)

Group	New Subs.	Top 1	RMSE		RKL		APR		Avg Rank
HKUST & ICT, CAS	3	7.5	0.90667	1.0	0.03429	5.0	50.07333	5.0	0.84129	4.625
shayantan	51	114.0	0.72000	2.0	0.03476	121.0	470.02000	105.0	0.72416	85.500

Second best RMSE out of all the participants (Sort it by RMSE)

Conclusion:

The KDD Cup 2004 protein homology problem was a highly imbalanced one and was challenging to begin with. We applied all the usual sampling techniques to balance the model, but the intrablock imbalance and the way they measure the performance the final leaderboard ranking, showed us that we should extract informative samples from each block, instead of trying it on the entire dataset. But after exploring many different models (many of which were not discussed here due to consistent poor performance), we finally settled with the blockwise modelling of SVMs and then extracting the negative support vectors from each block so as to include only the most informative samples. Since the evaluation was done separately for each block, building a global model would have been futile. Finally, aggregating all the negative support vectors with the positive samples gave us our final dataset to train on. **A radial kernel SVM with probability threshold 0.35 gave us the best results so far.**

References:

1. **Description of the features: http://osmot.cs.cornell.edu/kddcup/protein_description.pdf**
2. **Winning team performances (link)**
3. **SMOTE: Synthetic Minority Over-sampling Technique (link)**