Development of a SVM classifier of protein secondary structure topology into signal, transmembrane and globular structures

REPORT IN FULFILLMENT OF THE COURSE 'PROJECT IN MOLECULAR LIFE SCIENCE/KB8024' AT THE STOCKHOLM UNIVERSITY

Revant Gupta | Masters in Molecular Techniques in Life Science | 17.03.2017

Acknowledgement

A very big thank you to all involved in creating and implementing this project course. It was a very challenging and gratifying experience. I have been following machine learning courses for a long time and this project helped me practice on a real problem, in a very comprehensive manner. I think this experience has been very enabling when considering my future professional life. I feel that the independence in doing work that was expected was a major credit to this course. While there are always improvements to be made, I feel I am very happy overall.

I especially want to thank Erik Sjölund for some very interesting and informative discussions. His advice sparked ideas that I would not have considered or thought of otherwise. I was able to enhance my productivity and learning thanks to his input.

I also want to thank my classmates for making this an entertaining experience. I greatly appreciate the level of collaboration and organisation we were able to achieve. We had a very smooth working experience and I felt we learned a lot from each other as well. I hope my assistance helped my peers, gain better understanding of the task and its importance.

Index

SVM classifiers are frequently used in biology and have become very sophisticated in terms of problem specific optimisation. Sci-kit learn, is a popular python library that provides a broad range of machine learning algorithms, ability to create pipelines, various scoring metrics and other useful options. Compared to the basic steps of choosing a kernel function and optimising the hyperparameters, advanced strategies include combining classifiers (SVM or otherwise) in hierarchical or parallel schemes to improve quality. For SVMs, two stage models that are often used to sort data into signal only and transmembrane only classifiers and then using a consensus mechanism to arrive at a prediction. (Gubbi et al. 2006) Combination of techniques, for example using linear discriminant analysis for dimensionality reduction, are performed before using a SVM. (Kahsay et al. 2005) Furthermore, there have been attempts to use better scoring functions to improve prediction quality, as well designing custom kernels. (Leslie et al. 2002; Mukherjee & Mukherjee 2002; Vert 2002) Another significant consideration while improving prediction performance is the peptide representation. The successful recreation of biological differences of the proteins in the vectors is essential for a good predictor. Some common strategies are, adding evolutionary information using frequency matrices like PAM or PSIBLAST-profiles, transforming sparse data into sparse data representation or combining multiple representations. (Liu et al. 2014; Liu et al. 2010; Ancona et al. n.d.) Successful classification of signal vs transmembrane peptides remains a challenge due the commonality of features between the two. (Nugent & Jones 2009) Two methods that predict transmembrane and signal topologies using a single model, are PHOBIUS and PHILIUS. They based on HMMs and dynamic Bayesian learning respectively. (Reynolds et al. 2008)

SVM based methods have reached comparable levels of accuracy to other techniques like HMMs, NNs etc. for secondary structure topology predictions, specifically signal and transmembrane domains. (Nugent & Jones 2009; Mukherjee & Mukherjee 2002) It may be advantageous to use SVMs as it has been suggested that examining marginal support vectors provides information about the proteins at the boundaries of the classification. These proteins may exhibit characteristics that are common to both categories or at an intermediate value and hence may be biologically significant. (Mukherjee & Mukherjee 2002)

Algorithm	Eukaryotic	Human	Gram+	Gram-	E. coli
NN	97%	96%	96%	88%	89%
HMM	94%	-	96%	93%	-
SVM- Linear	96%	96%	97%	94%	91%
SVM- Poly	97%	97%	97%	94%	91%
SVM- Linear P250	98%	97%	97%	95%	92%

Table 1 : Classification accuracy for SVMs, NNs, and HMMs for signal peptide (Mukherjee & Mukherjee 2002)

Prediction Method	Test Database	
	Möller	TOPDB
MEMSAT-SVM	78%	67%
OCTOPUS	69%	64%
MEMSAT3	77%	66%
ENSEMBLE	61%	51%
PHOBIUS	67%	62%
HMMTOP	64%	57%
PRODIV	46%	37%
SVMTOP	70%	42%
TMHMM	60%	56%
PHDhtm	45%	49%

Table 2 Prediction results comparing MEMSAT-SVM with other contemporary methods (Nugent & Jones 2009)

DEVELOPMENT SUMMARY

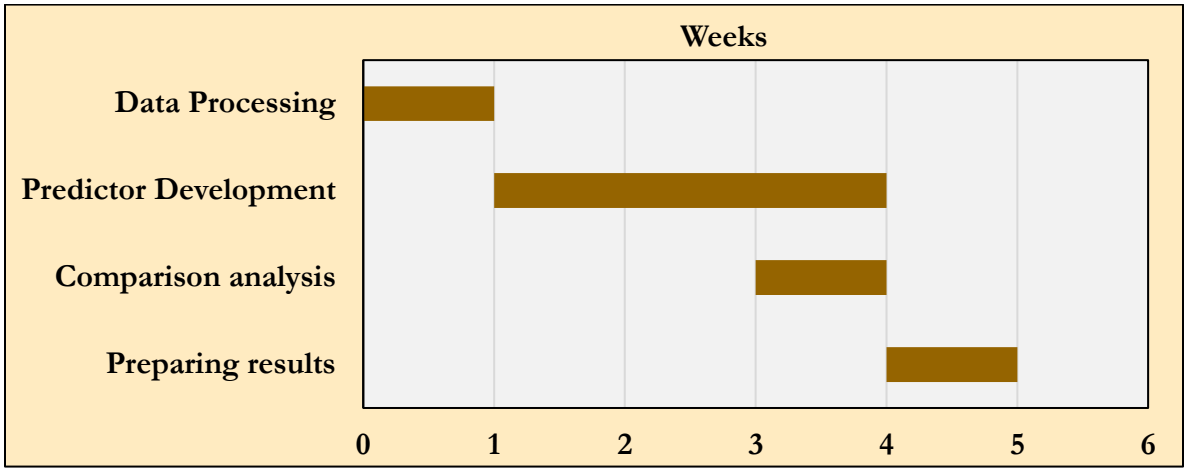
- Creating a data parser and sequence encoder.
- Testing various kernels and window sizes.
- Enabling replacement of hard encoding with PSIBLAST-profiles.
- Optimising the regularisation parameter.
- Use bagging to boost performance.
- Train a random forest and simple decision tree for comparison.
- Test models on 50 proteins.
- Choosing a final model.
- Generating predictions from novel sequences.

Availability:

The predictor along with the training & testing datasets used can be found at the following github repository along with detailed instructions.

https://github.com/aron0093/KB8024

Timeline:



DEVELOPMENT DESCRIPTION:

- **Creating a data parser and sequence encoder:**

The training data set consisted of 1065 sequences in the following format,

```
>Q6ZQA6
MKCFFPVLSCLAVLGVVSAQRQVTVQEGPLYRTESSHITIWCNVSGYQGPSEQNFQWSIYLPE….
SSSSSSSSSSSSSSSSSSGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG….
```

The secondary structure topologies consisted of S- Signal, G – Globular and T – Transmembrane.

A custom python module was created and each step of the process defined as a separate function,

1. `pre_vec_parser`
    - Inputs raw data as a `pandas DataFrame` from a specified file path and stores data as `Title, Sequence` and `Structure` columns.
    - Appends null strings equal to the window size at either end of the sequence and stores in `Sequence_windowed` column.
    - Performs regex substitution of special characters with "" on `Title`.
    - Returns a `pandas DataFrame`.

2. `cv_data_gen`
    - Splits data for cross validation into specified number of sets.
    - Splits at sequence level, preserving sequence information post encoding.
    - Data can be split in the original order of input or shuffled.

3. `skl_parser`
    - Inputs `pandas DataFrame` from `pre_vec_parser`.
    - Outputs encoded sequences, structures as separate feature, labels `numpy arrays`.
    - Features are 20*(2n+1) dimensional vectors where n is the window size.
    - Inputs window size to create appropriate `numpy arrays` which are preferred over `lists` to reduce memory consumption.

The features and labels were encoded using the following dictionaries,

```
aa_dic = {  'A':[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,],
            'C':[0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,],
            'D':[0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,],
            'E':[0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,],
            'F':[0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,],
            'G':[0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,],
            'H':[0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,],
            'I':[0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,],
            'K':[0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,],
            'L':[0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,],
            'M':[0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,],
            'N':[0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,],
            'P':[0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,],
            'Q':[0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,],
            'R':[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,],
            'S':[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,],
            'T':[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,],
            'V':[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,],
            'W':[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,],
            'Y':[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,],
            'B':[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,] }


structure_dic = {'S':-1, 'M':1, 'G':0}
```

Each amino acid is treated as a 20 dimensional vector, with the null character used to encode windows at the edges of the sequence and is represented using a null vector. The choice of the null character being represented as a null vector is due to the following reasons,

- The subsequent replacement of hard encoding with PSIBLAST-profiles or PSSMs would be smoother if amino acids were represented by a 20 dimensional vector. This would enable the direct use of PSSMs as vectors.
- Since the null character is added for convenience, mathematically it is appropriate to treat it as the origin, thus implying zero distance on any dimension, while each dimension represents an amino acid.

To encode windows, relevant amino acid vectors are concatenated to produce a 20*(2n+1) dimensional vector that preserves the sequence order. (Cai et al. 2003) This is in contrast to some representations like, the amino acid composition where the sequence order is lost. (Liu et al. 2010) Labels are corresponding structures of the central amino acid in a window.

Note: A window has been defined as the +/- number of amino acids around the target. Therefore for a window size n, the feature vector is of size 20*(2n+1).

Location: /bin/dense_data_parser.py; /SignalP/scripts/dense_data_parser.py

- **Testing various kernels and window sizes:**

Choosing or creating a kernel function is a decision that probably has the highest impact on the result. Given any set of data the primary step is to establish a baseline using a linear kernel. (Ben-Hur et al. 2008) While there does exist data that is not linearly separable, a simplistic example being the output of a XOR gate, performance of a linear kernel tends to increase as data size increases. This is due to the fact that more samples tend to form easily separable clusters.

On the other hand for low number of samples, a Gaussian kernel is commonly used. (Liu et al. 2010) Furthermore many custom kernels are created that better reflect the separability of particular data. For this project, only default kernels available from the sci-kit learn library were tested.

- Based on the literature survey, linear, rbf and sigmoid kernels were tested on 500 proteins.
- The proteins were obtained by resampling the data in a consistent way to preserve the overall distribution pattern of the various classes.
- For the linear kernel, `LinearSVC` implementation was used as the underlying C library (liblinear) is well optimised, resulting in a lower time complexity than the default implementation.
- For the other kernels, the `OneVsRest` wrapper was used with the `njobs` parameter to obtain results in a reasonable time frame. In all cases, the one vs rest strategy was employed.

It has been shown that one vs rest is reasonably good at classification though in a biological context where it is common to share features with multiple classes, one vs one might have a better

performance. (Su et al. 2007) However due to the prohibitively high amount of time required to train SVMs using the one vs one strategy, it was ruled out during the preliminary testing.

Since the training data was highly biased for globular domains and the predictor was aimed at signal & trans-membrane domains, the performance was evaluated for each class separately. The `precision_recall_f1score_support` scoring function was used.

Given the lower presence of signal and transmembrane peptides, identifying positives for either class was a priority, therefore the f1-score with macro averaging was chosen as the performance metric. All tests were performed with 3 fold cross validation. It is defined as the harmonic mean of precision and accuracy,

$$\text{Recall/Sensitivity} = \text{True Positives/All Positives}$$

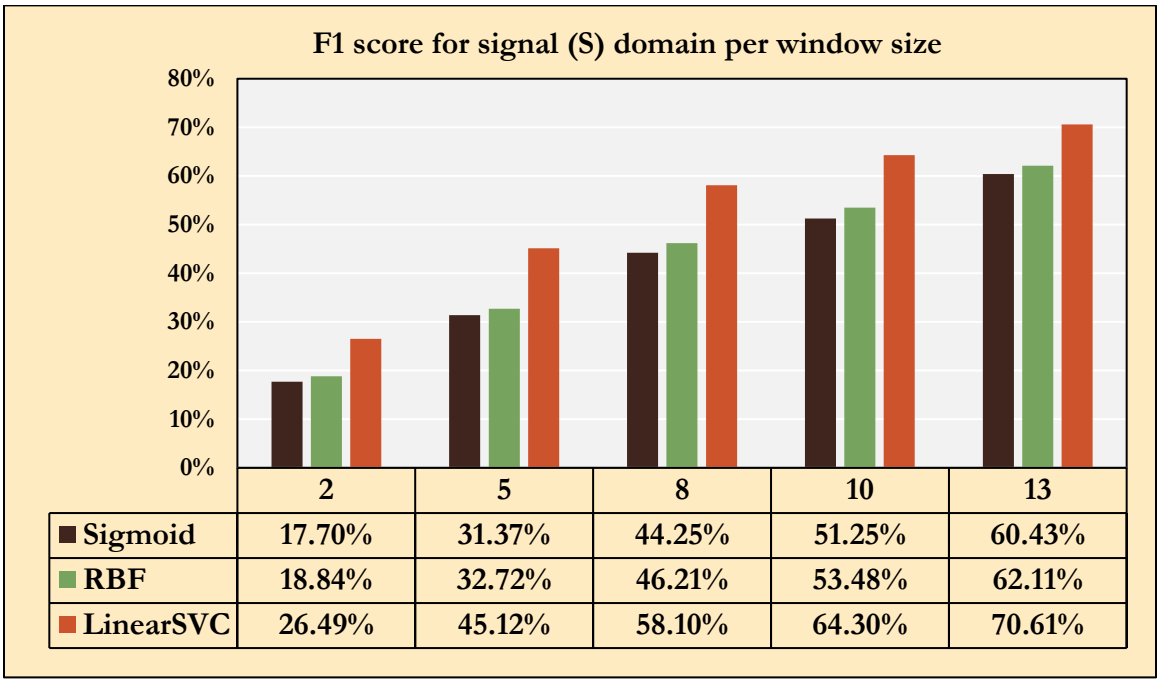
$$\text{Precision} = \text{True Positives/Predicted Positives}$$

$$\text{F1 Score} = 2*(\text{Recall}*\text{Precision})/(\text{Recall}+\text{Precision})$$

The full data and code can be found in the github repository,

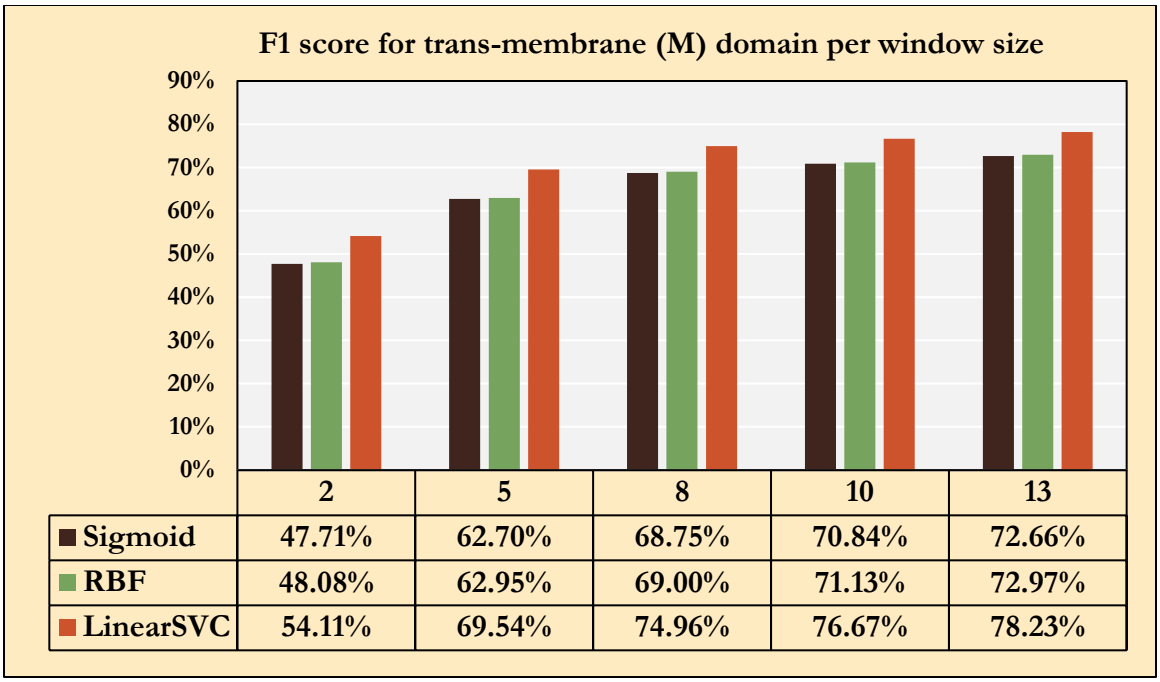
Location: /SignalP/scripts/window_kernel_CV.py; /SignalP/scripts/window_linear_CV.py; /SignalP/output/window_kernel/

The following charts summarise the obtained results,



	2	5	8	10	13
Sigmoid	17.70%	31.37%	44.25%	51.25%	60.43%
RBF	18.84%	32.72%	46.21%	53.48%	62.11%
LinearSVC	26.49%	45.12%	58.10%	64.30%	70.61%

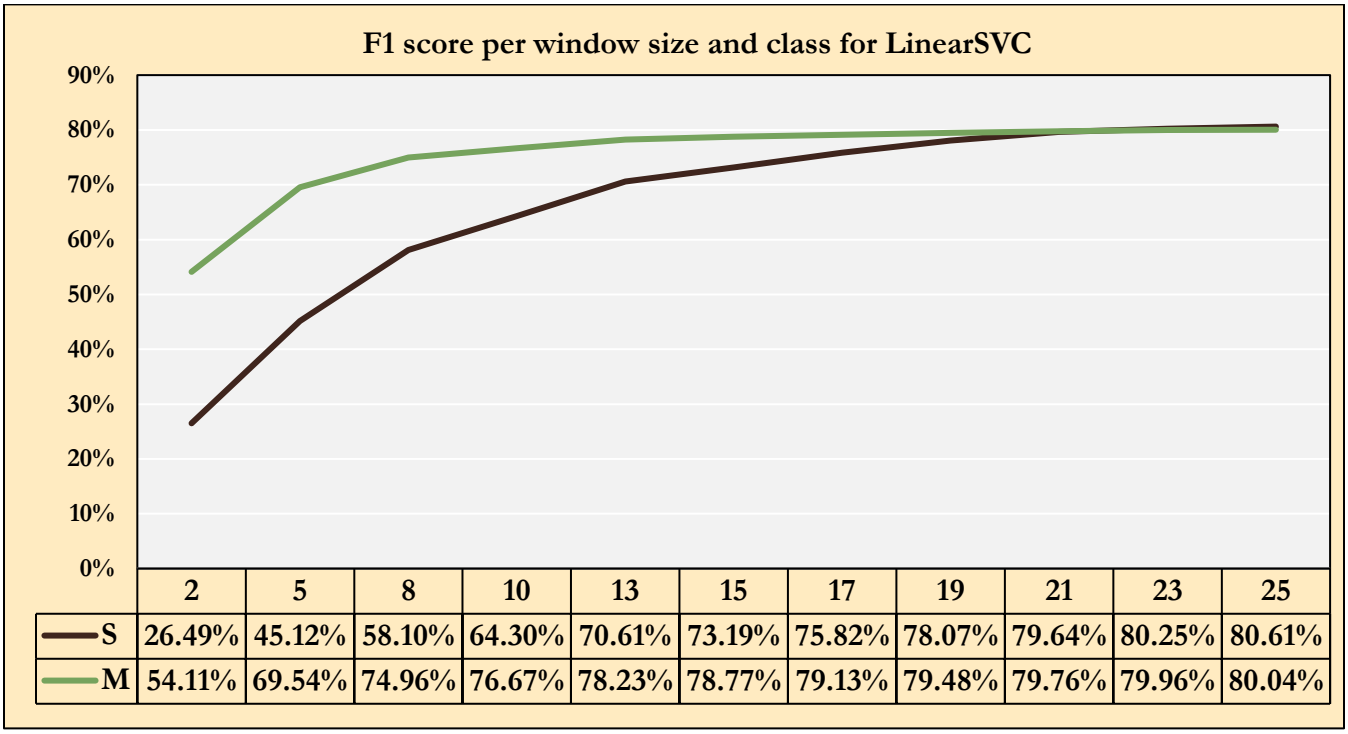
While not shown here, the globular domain also followed the same trend of increasing scores with window size with `LinearSVC` performing consistently better than the other kernels.



	2	5	8	10	13
Sigmoid	47.71%	62.70%	68.75%	70.84%	72.66%
RBF	48.08%	62.95%	69.00%	71.13%	72.97%
LinearSVC	54.11%	69.54%	74.96%	76.67%	78.23%

As the data indicates, the performance of `LinearSVC` was consistently better over a range of window sizes and given the much greater time complexity of the other kernels, `LinearSVC` was chosen as the kernel for further development.

The scores were also positively correlated with window size and therefore `LinearSVC` was further optimised for window size. The scores follow a logarithmic trend and start plateauing around size 25.



	2	5	8	10	13	15	17	19	21	23	25
S	26.49%	45.12%	58.10%	64.30%	70.61%	73.19%	75.82%	78.07%	79.64%	80.25%	80.61%
M	54.11%	69.54%	74.96%	76.67%	78.23%	78.77%	79.13%	79.48%	79.76%	79.96%	80.04%

- **Enabling replacement of hard encoding with PSIBLAST-profiles:**

The fundamental premise of structure prediction from sequence information is that the particular order of amino acids is important for the peptide's functionality. (Liu et al. 2014) Therefore adding evolutionary information derived from homologous proteins is crucial, especially when the training set is small.

PSIBLAST-PSSMs enable us to relate positions in the sequences to frequency probabilities of amino acids at that position. From a mathematical perspective, the support vector so generated can be considered a more accurate feature for the corresponding label in the 20 dimensional space.

Both substitution and frequency profiles were tested. PSIBLAST was run using Uniref50 with 4 iterations. Uniref50 was chosen to include wider number of homologues and generalise the predictions as much as possible. While using Uniref50 may include functionally different proteins in its clusters, signal peptide domains are generally more conserved (Williams et al. 2000) and therefore may be better represented by this methodology. This is hypothesised to be due to the increase in contrast with the globular domains which will have more diverse representations.

Running PSIBLAST with 4 iterations is a time consuming process. To counter this and obtain results in a reasonable time frame, the following process was followed,

1. `data_divide`
    - Fasta format files for each sequence are created and stored in a specified number of folders.
2. `pssm_gen_ssh`
    - Creates bash scripts to run PSIBLAST for files in each folder created by `data_divide`.
    - Executes PSIBLAST on a given number of servers over SSH using `paramiko`.
    - Stores generated PSSMs in a specified folder.

Generated PSSMs were used to encode sequences,

1. `skl_pssm_parser`
    - Performs the same functions as `skl_parser` (see point 3, Creating a data parser and sequence encoder) except using stored PSSMs as vectors instead of hard coded vectors from the dictionary.

Location: /SignalP/scripts/pssm_func.py; /SignalP/scripts/pssm_script.py; /bin/pssm_data_parser.py; /SignalP/scripts/pssm_data_parser.py;

- **Optimising the regularisation parameter**

The regularisation parameter was optimised for different window sizes for models with default encoding, vectors derived from the normalised substitution matrix and vectors derived from the normalised frequency matrix. The `GridsearchCV` function with 3 fold cross validation was used. Average F1 score for only signal (S) and trans-membrane (M) classes was considered.

The best scores however were obtained for window size 25 as expected from earlier testing.

Encoding	F1 Score (S+M)	Regularisation Parameter
Default Encoding	80.79%	4
Normalised substitution matrix	77.86%	1
Normalised frequency matrix	74.93%	16

Table 3 Optimal C parameter per encoding type with corresponding F1 score.

Interestingly, the default encoding had consistently superior scores though given the small size of the training data, this may not be significant. Models based on PSSMs, while having a lower score on the training set, do have greater generalisation and therefore are preferable.

The full data for these tests can be found at the repository,

Location: /SignalP/output/Linear_grid_search/; /SignalP/scripts/Linear_Grid_Search.py; /SignalP/scripts/Linear_Grid_Search_pssm.py

- **Use Bagging to boost performance:**

Bagging (Bootstrap Aggregating) is a strategy that boosts the performance of a predictor by combining several weak predictors to create a strong predictor. It particularly counteracts overfitting and data bias. Given the small size of our data set and its bias, this was a suitable strategy. Bagging works well as a stabilising influence on unstable classifiers trained on small data sets. (Skurichina & Duin 1998)

10 individual predictors were used in our model with maximum training data per model equal to a tenth of the whole data.

Location: /SignalP/scripts/model_builder.py

- **Train a random forest and simple decision tree for comparison**

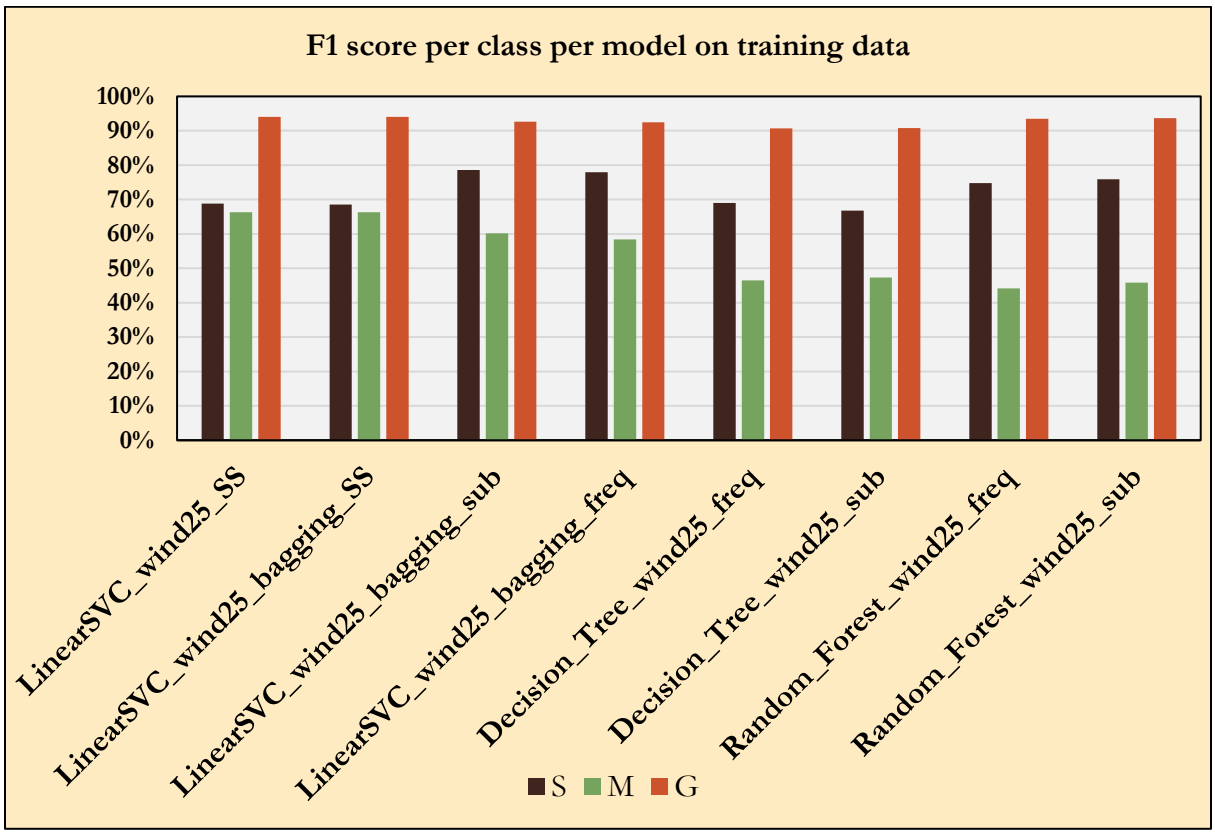
Decision trees are probably the most intuitive and understandable method for classification problems. A decision tree is a flow chart with each node representing a decision that splits it into two branches. However a major problem with decision trees is their tendency to over fit data. Random forests are bagging models of decision trees and counteract the tendency to over fit by training multiple trees of parts of the data and considering the mode of predictions by individual trees as the final classification. Decision trees and random forests are computationally much less expensive than SVMs. They also have a high accuracy for many kinds of data and are a go to method for a quick result. Both decision trees and random forest models were trained for comparison.

Location: /SignalP/scripts/Decision_Tree_builder.py; /SignalP/scripts/Random_forest_builder.py

The following models were trained,

- LinearSVC_wind25_SS
- LinearSVC_wind25_bagging_SS
- LinearSVC_wind25_bagging_sub
- LinearSVC_wind25_bagging_freq
- Decision_Tree_wind25_sub
- Decsion_Tree_wind25_freq
- Random_Forest_wind25_sub
- Random_Forest_wind25_freq

SS: Default encoding; sub: Substitution matrix; freq: Frequency matrix; wind25: Window size 25



Bagging SVMs perform better than simple SVMs, decision trees and random forests for signal peptide prediction. For trans-membrane prediction, the addition of evolutionary information degrades prediction however bagging models are still better than simple SVMs. Decision trees and random forests are particularly bad at trans-membrane prediction. Random forests in these models were composed of 128 estimators. The use of substitution or frequency matrix had comparable results. See the model_stats folder in the repository for the full classification report along with confusion matrices.

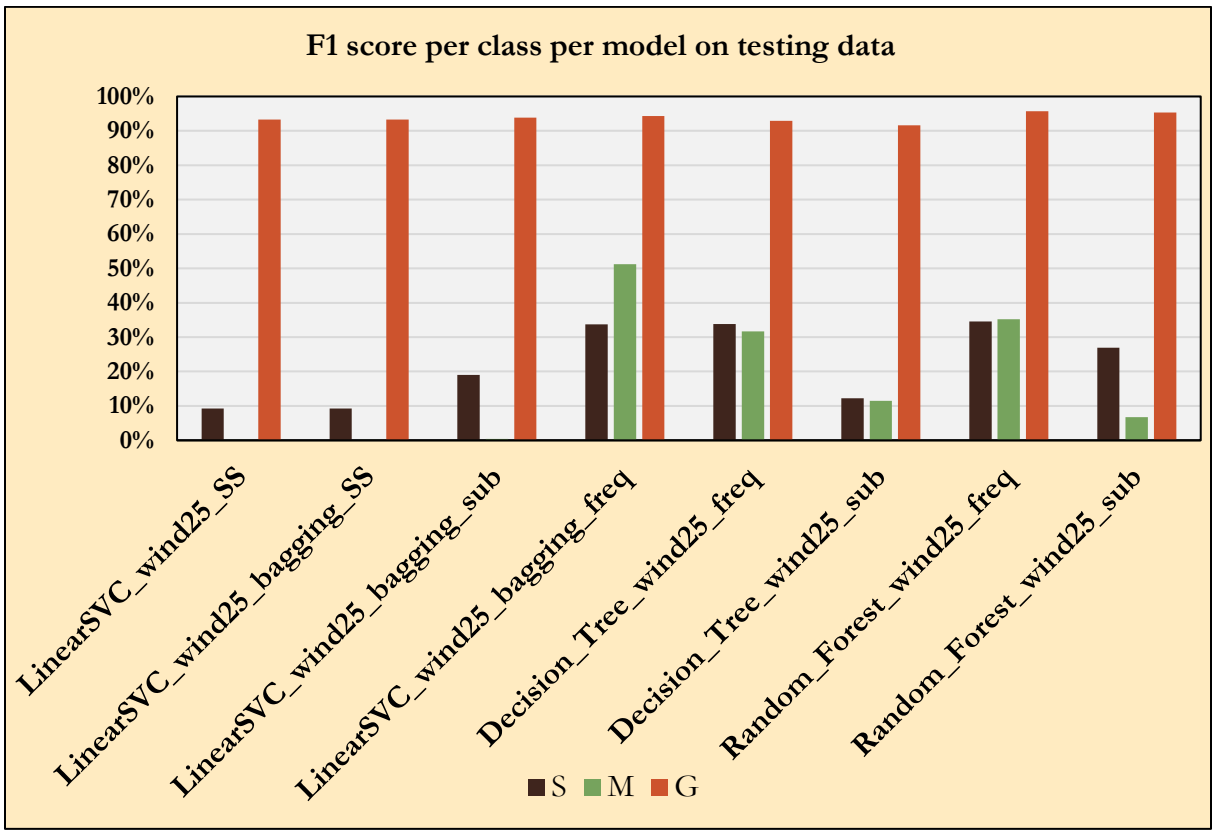
Location: /SignalP/output/model/; /SignalP/output/model/model_stats/

- **Test models on 50 proteins:**

"The Topology Data Bank of Transmembrane Proteins (TOPDB) is the most complete and comprehensive collection of transmembrane protein datasets containing experimentally derived topology information currently available." (Tusnády et al. 2008) In addition TOPDB was used as a training set for PHOBIUS, which is also a method to predict signal and trans-membrane topologies in one model. Data was obtained from the TOPDB website at the sequence & topologies option.

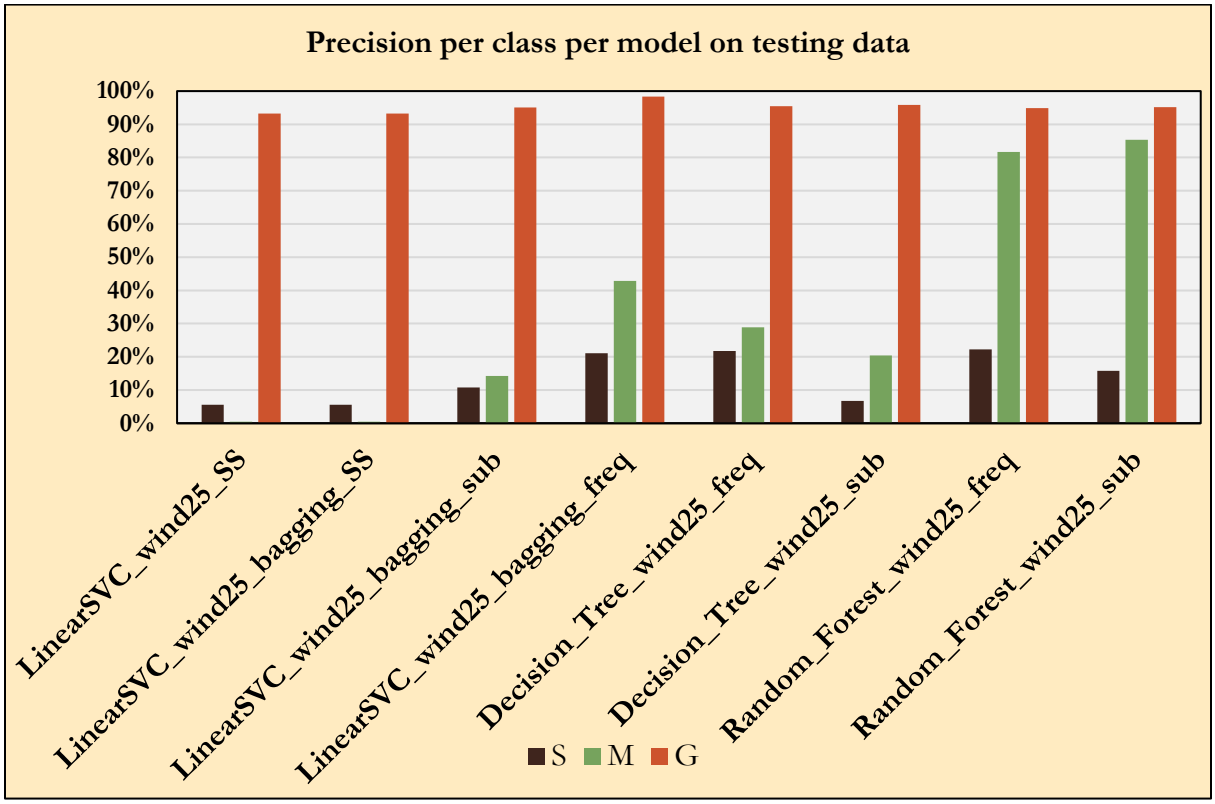
http://topdb.enzim.hu/?m=download&mid=2

Fifty proteins were selected from this data as the test set for the trained models,



There are some interesting observations derived from the data above which enabled the selection of a final model from the list of trained models mentioned earlier,

- Bagging has a relatively small effect in boosting model performance both on SVM and decision trees, with both bagged and non-bagged models having comparable performance.
- We can clearly see that using PSSMs gives better performance irrespective of the model used. Furthermore, using the frequency matrix has significantly higher performance than the substitution matrix.
- The performance results for trans-membrane domains are the most affected by the use of PSSMs and type of matrix.



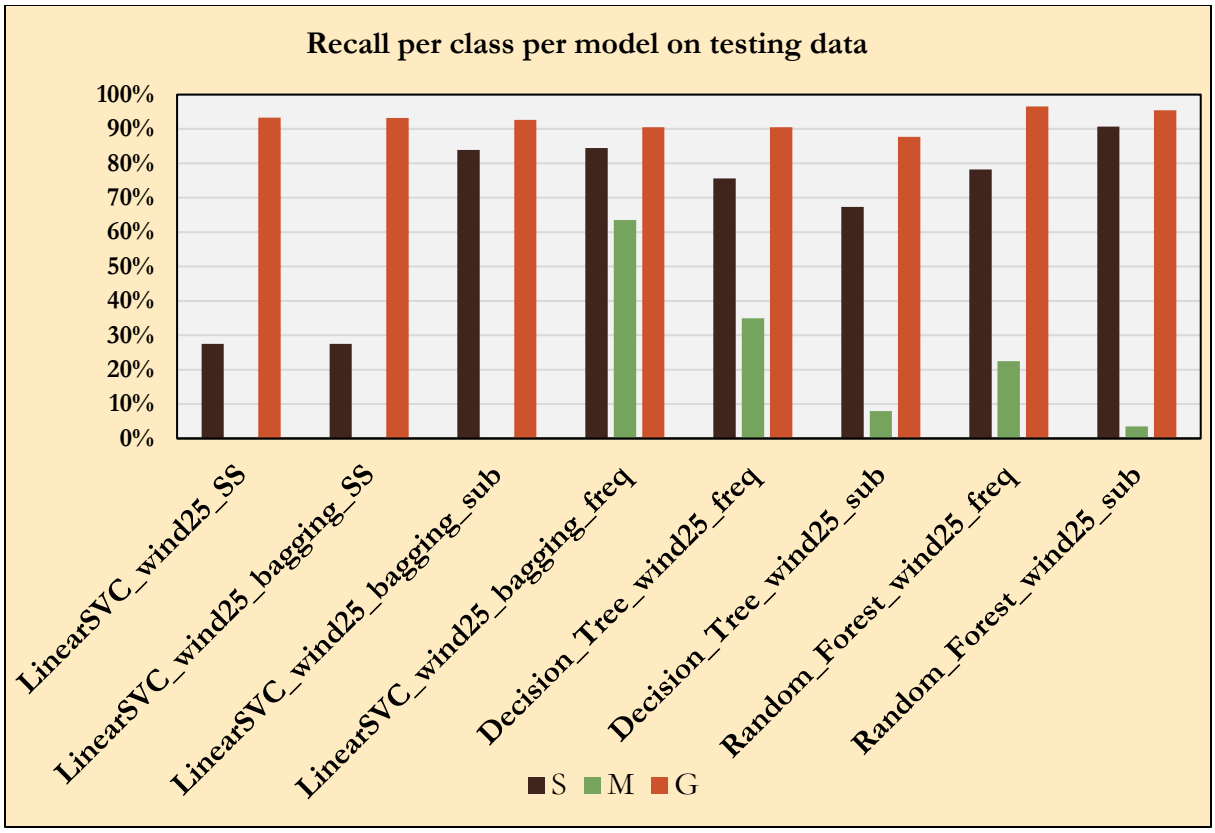
In the data above we can observe that,

- The use of the frequency matrix has a great impact on the precision scores for trans-membrane domains though the precision for signal domains is not affected as much.
- Bagging has a modest effect on boosting performance.

On the other hand, in the data shown below, for recall,

- Using any kind of PSSM translates into a huge increase in performance.
- Unlike precision both signal and trans-membrane domains are affected.
- All models using PSSMs are highly sensitive (recall) for signal peptides which implies that they are successfully able to capture almost all signal peptides.
- For trans-membrane domain, it is better to use the frequency matrix, which is consistent with the data above.
- Bagging has consistently shown only a modest increase in performance.

From this we can conclude that adding evolutionary information into our analysis boosts the sensitivity towards secondary structure topological domains which enables to capture a high number of positives, however lower precision, implies high false positives remain an issue to be resolved.



The full classification report for each model and confusion matrices can be found in the repository.

Location: /SignalP/output/test_stats/

- **Choosing a final model:**

Based on the data from both the training and testing, it is apparent that the model LinearSVC_wind25_bagging_freq has the greatest performance across all scores. It was chosen as the final model and saved as Final_model.

Location: /SignalP/output/model/Final_model

Below, we can see that,

- LinearSVC_wind25_bagging_freq has the greatest sensitivity in both signal and trans-membrane domain prediction
- LinearSVC_wind25_bagging_sub performs worse for trans-membrane domains though has a comparable sensitivity for signal domains.
- LinearSVC_wind25_bagging_SS has a much worse sensitivity than either of the models above.
- Random_Forest_wind25_freq had a reasonably good sensitivity for signal domains though it is still worse than SVM models, while it has a bad sensitivity for trans-membrane domains.

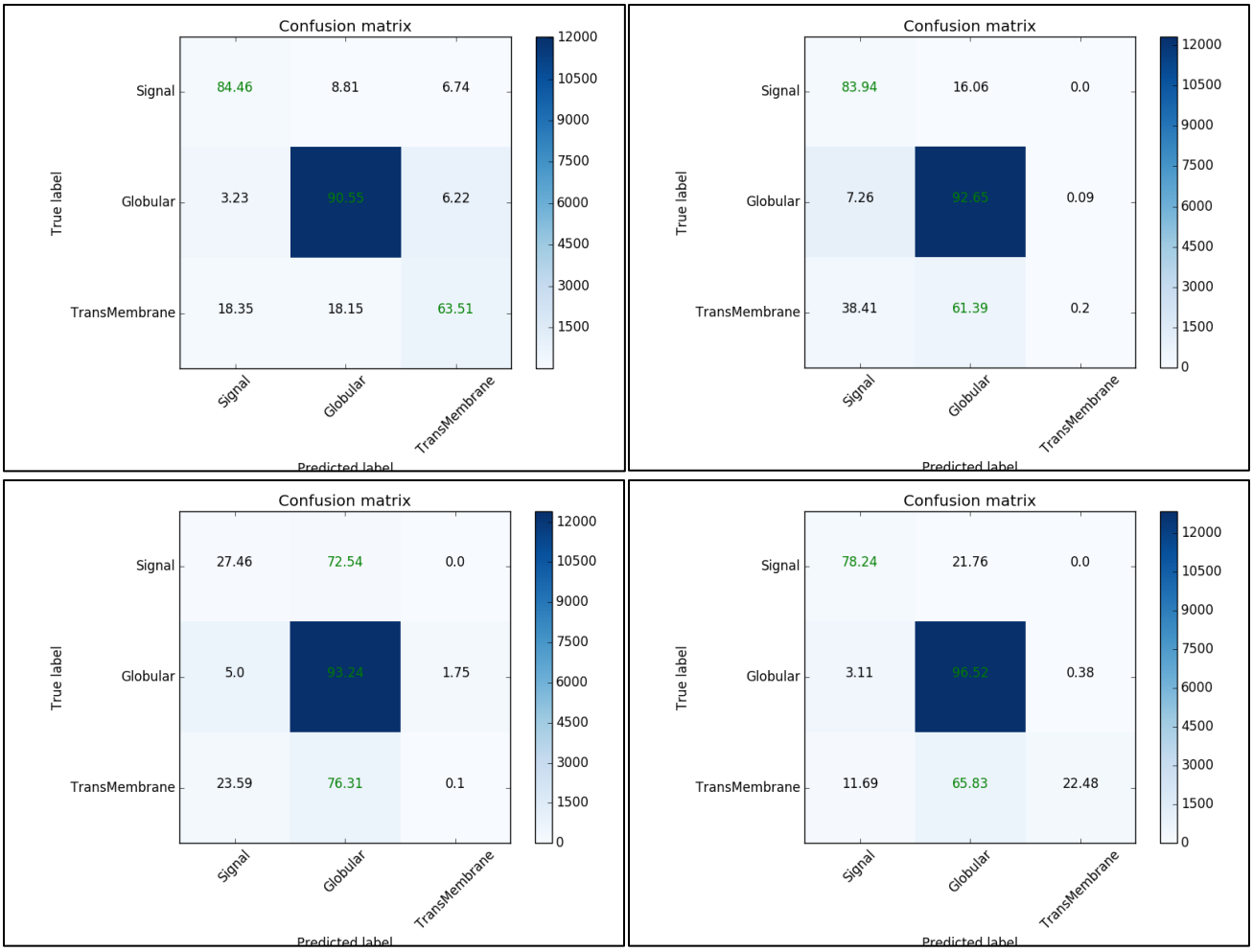


Figure 1 Confusion matrices reflecting sensitivity in clockwise order for the models a) LinearSVC_wind25_bagging_freq b) LinearSVC_wind25_bagging_sub c) LinearSVC_wind25_bagging_SS d) Random_Forest_wind25_freq

- **Generating predictions from novel sequences:**

For generating predictions a python script was written and the following process was followed,

1. Input is accepted as a fasta format file with as many proteins for which predictions are required.
2. Temporary files for each sequence are generated and PSIBLAST performed using the `subprocess` module.
3. The inputs undergo the same data processing steps as the training data and predicted class labels are converted to structure labels using a reverse structure dictionary.
4. The final predictions are refined using objective rules
    - If the first 10 residues contain more than or equal to 50 % 'S' then the entire 10 residues are converted to 'S'. This is due to the nature of signal peptides which only occur at the start of the peptide. The length could possibly be increased for further refinement.

- If there exist only a single 'S' residue surrounded by 'G' or 'M' after the first step, then they are converted to 'M'. This is done to avoid erroneous classification of 'M' as 'S'.
- If there exist only a single 'M' or 'MM' surrounded by 'G' then they are converted to 'G' or 'GG'. This is done since typically, trans-membrane proteins are either 3-5 (β barrels) or 15-20 (α helices) residues long.

Location: /bin/Predictor.py; /bin/SignalP_predict.py

CONCLUSION: The developed predictor has a good sensitivity towards signal domains however classification of trans-membrane domains is not yet satisfactory. The discrimination between signal and transmembrane peptides remains a challenge. While the model shows reasonably good sensitivity for all classes, the precision could be improved further. Further development would involve the use of combinational techniques/multi model classification. Some other strategies would include generating better representations of the data and identifying then weighting features that have a high differentiation between classes. In terms of the state of the art for SVM based predictors, they are competitive compared to other machine learning algorithms and have the advantage of generating biologically relevant information compared to more 'black box' techniques like neural networks.

References

Ancona, N. et al., 1 Sparse Vs Dense Data Representations in Kernel Methods. *Sistemi Intelligenti*, pp.1–6.

Ben-Hur, A. et al., 2008. Support vector machines and kernels for computational biology. *PLoS Computational Biology*, 4(10).

Cai, Y.-D., Lin, S. & Chou, K.-C., 2003. Support vector machines for prediction of protein signal sequences and their cleavage sites. *Peptides*, 24(1), pp.159–161. Available at: http://www.sciencedirect.com/science/article/pii/S0196978102002899.

Gubbi, J. et al., 2006. Protein topology classification using two-stage support vector machines. *Genome informatics. International Conference on Genome Informatics*, 17(2), pp.259–69. Available at: http://www.ncbi.nlm.nih.gov/pubmed/17503398.

Kahsay, R.Y., Gao, G.R. & Liao, L., 2005. Discriminating transmembrane proteins from signal peptides using SVM-Fisher approach. *Proceedings - ICMLA 2005: Fourth International Conference on Machine Learning and Applications*, 2005, pp.151–155.

Leslie, C., Eskin, E. & Noble, W.S., 2002. The spectrum kernel: a string kernel for SVM protein classification. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, 575, pp.564–575.

Liu, B. et al., 2014. Combining evolutionary information extracted from frequency profiles with sequence-based kernels for protein remote homology detection. *Bioinformatics*, 30(4), pp.472–479.

Liu, T., Zheng, X. & Wang, J., 2010. Prediction of protein structural class for low-similarity sequences using support vector machine and PSI-BLAST profile. *Biochimie*, 92(10), pp.1330–1334. Available at: http://dx.doi.org/10.1016/j.biochi.2010.06.013.

Mukherjee, N. & Mukherjee, S., 2002. Predicting Signal Peptides. *Solutions*, pp.1–7.

Nugent, T. & Jones, D.T., 2009. Transmembrane protein topology prediction using support vector machines. {*BMC*} *Bioinformatics*, 10(1), p.159. Available at: http://dx.doi.org/10.1186/1471-2105-10-159%5Cnhttp://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2700806&tool=pmcentrez&rendertype=abstract.

Reynolds, S.M. et al., 2008. Transmembrane topology and signal peptide prediction using dynamic Bayesian networks. *PLoS Computational Biology*, 4(11).

Skurichina, M. & Duin, R.P.W., 1998. Bagging for linear classifiers. *Pattern Recognition*, 31(7), pp.909–930. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0031320397001106.

Su, E.C.-Y. et al., 2007. Protein subcellular localization prediction based on compartment-specific features and structure conservation. *BMC bioinformatics*, 8, p.330. Available at:

http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2040162&tool=pmcentrez&rendertype=abstract.

Tusnády, G.E., Kalmár, L. & Simon, I., 2008. TOPDB: Topology data bank of transmembrane proteins. *Nucleic Acids Research*, 36(SUPPL. 1), pp.234–239.

Vert, J., 2002. Support vector machine prediction of signal peptide cleavage site using a new class of kernels for strings. *Proceedings of the Pacific Symposium on*, 660, pp.649–660. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.85.6588&rep=rep1&type=pdf.

Williams, E.J.B., Pal, C. & Hurst, L.D., 2000. The molecular evolution of signal peptides. *Gene*, 253(2), pp.313–22. Available at: http://linkinghub.elsevier.com/retrieve/pii/S037811190000233X%5Cnhttp://www.ncbi.nlm.nih.gov/pubmed/10940569.