# 8SS: A new approach to 8-state secondary structure prediction

**Abstract**

8-state secondary structure prediction is a challenging problem that has until recently remained relatively unexplored. Most early secondary structure prediction methods focused on the much less complex helix, strand, or coil 3-state prediction. Here, we present a new 8-state secondary structure predictor, 8SS, with 3 versions, each using one of either support vector machines, simple decision trees, or random forests. By incorporating evolutionary sequence information using PSI-BLAST and information about adjacent amino acid residues using various window sizes, our method successfully predicts most secondary structure elements. After optimization of several input and model parameters, our best predictor has a Q8 accuracy of ~69% when tested on a set of 718 proteins, which is comparable to other currently available state-of-the-art predictors such as RaptorX-SS8 and SSpro8.

**Introduction**

Secondary structure prediction is a challenging problem wherein the goal is to forecast the local secondary structure conformation of a protein segment based on its primary sequence of amino acids. As it is known that the primary sequence of a protein contains all the necessary information for the protein to fold into its secondary and tertiary structure, it should therefore be possible to assign secondary structures to segments if the protein sequence is known (1). There are two main types of prediction strategies: those that classify into helices, sheets, and coils, known as HEC or Q3, or those that classify into 8 different states including 3 types of helices, extended strands, beta bridges, turns, bends, and coils, known as Q8. Naturally, due to the 5 additional possible states, Q8 prediction is significantly more complicated than Q3 prediction, usually limited by poor prediction of less common features that are not alpha-helices, extended beta strands, or coils.

Since the mid 1970's, prediction of secondary structure has been a hot research topic. First generation prediction methods, like the one published in 1974 by Chou and Fasman, relied on the propensities of amino acids to form certain secondary structures to guide predictions. Unfortunately, due to lack of further information, this method was only about 50-60% accurate in Q3 (2). The next major improvement to this prediction came through the GOR3 method, which incorporated information on neighbouring amino acids in the form of a sliding window to improve Q3 accuracy above 60% (3). An important contribution to this field was published in 1983 in the form of the dictionary of protein secondary structure, or DSSP, that served to standardize definitions for classification of secondary structure into either 8 or 3 classes (4). Many years later, in 1993, prediction finally broke the 70% Q3 accuracy barrier with the PhD method (5). The increase in accuracy was due to incorporation of multiple sequence alignments using a position specific scoring matrix (PSSM) as well as a multi layer neural network to make decisions. Probabilistic calculations are first performed to convert sequence to structure, then structure to structure based on nearby probabilities, and finally to make a winner takes all decision (5). Current popular methods such as PSI-PRED (1999) are based on the PhD method but incorporate PSI-BLAST for multiple sequence alignment along with larger reference databases for improved accuracy (6).

Though the accuracy of the latest predictors hovers around 70% for 8 state prediction (Q8) and 80% for 3 state prediction (Q3), researchers are still keen to solve the remaining 20-30% (6). For example, one article released in 2016 by Wang et al. uses deep convolutional neural fields for secondary structure prediction, with claimed accuracies of 84% and 72% for Q3 and Q8 respectively (7). Continued advances

in both database coverage and machine learning will likely allow predictors to reach even higher accuracies in the near future.

Herein, we present a new 8-state secondary structure predictor built using python-based machine learning alongside a cross-validated dataset of 718 proteins with known secondary structure. After optimization including testing of different window sizes and incorporation of evolutionary information using PSI-BLAST, this method yields an accuracy of around 70% for Q8 prediction when testing on a subset of our 718 proteins, which is competitive in comparison to the best currently available 8-state predictors.

**Method**

The dataset used to build, train, and test this prediction method is a set of 718 protein chains with known sequence and secondary structure that have been aggregated from the PDB. Each sequence has an average length of 222, and the total number of amino acids in the dataset is 159507. The complete dataset is available for review at https://github.com/manuelester/bioinformatics-project-course/blob/master/Final%20Scripts/8SS%20Predictor/dssp_8_state.3line.txt .

From the original dataset, 3 element blocks were created for each protein, containing the protein ID, sequence, and corresponding secondary structure. For each sequence, additional evolutionary information was obtained using a multiple sequence alignment through PSI-BLAST with the UniRef90 database, E-value of 0.01, and 3 iterations (8). The PSI-BLAST conditions were chosen based on those used by default in PSI-PRED, which is a well established reliable secondary structure prediction method (6). The resultant PSSM was then reimported and appended to each 3 element block, extending it to 5 elements using the substitution matrix (SM) and frequency matrix (FM) PSSM. Normalization of SM values was done using the sigmoid function as in PSI-PRED (6). FM values were normalized to between 0-1 by dividing each value by 100, the maximum possible frequency of an amino acid in any given position.

To avoid bias in the training and test sets due to related sequences being grouped together in the original dataset, the protein sequences were randomized prior to cross-validation splitting. A cross-validation fold-value of 5 was chosen to match the popular 80-20 training-test split for cross-validation datasets (9). Furthermore, cross-validation sets were divided into equal parts on the protein level and not the amino acid level, so as to minimize similarity bias between training and test sets that can occur when placing related adjacent residues into separate sets.

Each set was then processed to convert sequences and features into a suitable format to use as input to the built-in machine learning algorithms found in the sklearn module of Python. Features were converted into numbers, ranging from 1-8. Sequence PSSMs were converted into a 20*W long list for each corresponding feature, where W is the chosen odd number window size surrounding the central residue N, and 20 is the length of a SM or FM entry for one residue from the PSSM.

Finally, the converted and divided dataset was used to build an SVM-based classifier for 8-state secondary structure prediction. Each of the 5 training sets was used subsequently to train an SVM model, which clusters features and builds decision surfaces to separate each feature from the others. These SVMs were then challenged to predict features from given protein sequences in the corresponding test sets. To evaluate the predictor performance, both feature-specific and average values were calculated for precision, recall, and F1 score, along with a confusion matrix containing true versus predicted labels.

Optimization of the predictor was done by iterating through model building using various parameters as well as other machine learning algorithms. The parameters tested were: window sizes from 7-17, balanced and unbalanced class weights, C-values between 1-100, model building using normalized SM and FM scores, as well as decision tree and random forests algorithms.

**Results**

To examine the reliability of SVM model performance and check for overfitting due to similarity between correspondent training and test datasets, a 5-fold cross validation was carried out with 80/20 train/test splits. As shown in Figure 1, the performance across all 5 train/test splits is consistent, with an average accuracy of 0.536 and standard deviation of 0.001. Therefore, we can be confident that the SVM does not suffer from overfitting and has reproducible performance regardless of which data from the complete dataset is assigned to the training and test sets.

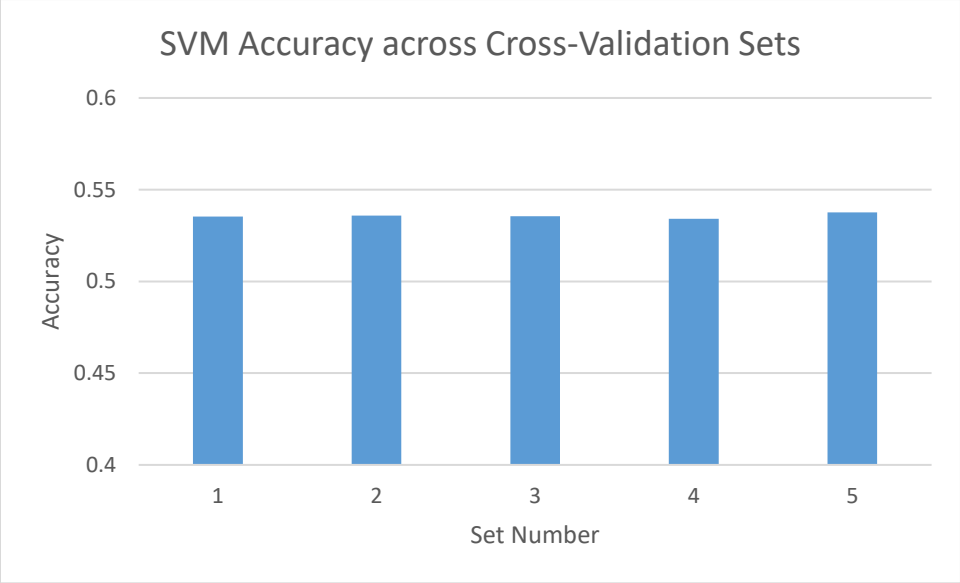


**Figure 1:** 5-fold cross-validation of SVM model.

After confirming the prediction reliability of the SVM model built using the 718 protein dataset, various parameters were modified systematically, and their influence on model performance evaluated. In accordance with the methodology used in building predictors such as DeepCNF-SS, we tested sliding window sizes around 11, from 7-17 residues, in combination with SM or FM data, and balanced or unbalanced class weights in the SVM algorithm (7). It appears, in Figure 2A, that F1 scores level off at around window size 13 for all methods except for SM balanced, which does not plateau within the tested range. Accuracy scores for all methods continue to increase with increasing window size, albeit less drastically between 13-17. It must be noted, however, that the time it takes to build SVM models increases exponentially with increasing window sizes, a factor that may be relevant when choosing the final parameters of the model. Interestingly, the maximal F1 scores were achieved using SM balanced models, which also resulted in the lowest accuracy scores. The reverse can be seen for the FM unbalanced models.

Using Python's built-in svm.SVC and svm.LinearSVC machine learning tools, we evaluated performance using C-values from 1-100, and 4 different SVM kernels: linear, rbf, sigmoid, and polynomial. C values determine the distance each decision surface should have from the closest point in each feature

cluster, with larger C values resulting in smaller distances (10). Kernels on the other hand refer to the shape of the decision surface line (10). However, somewhat unremarkably, performance was best using the default C-value of 1 and the linear SVC kernel in svm.LinearSVC (data not shown).

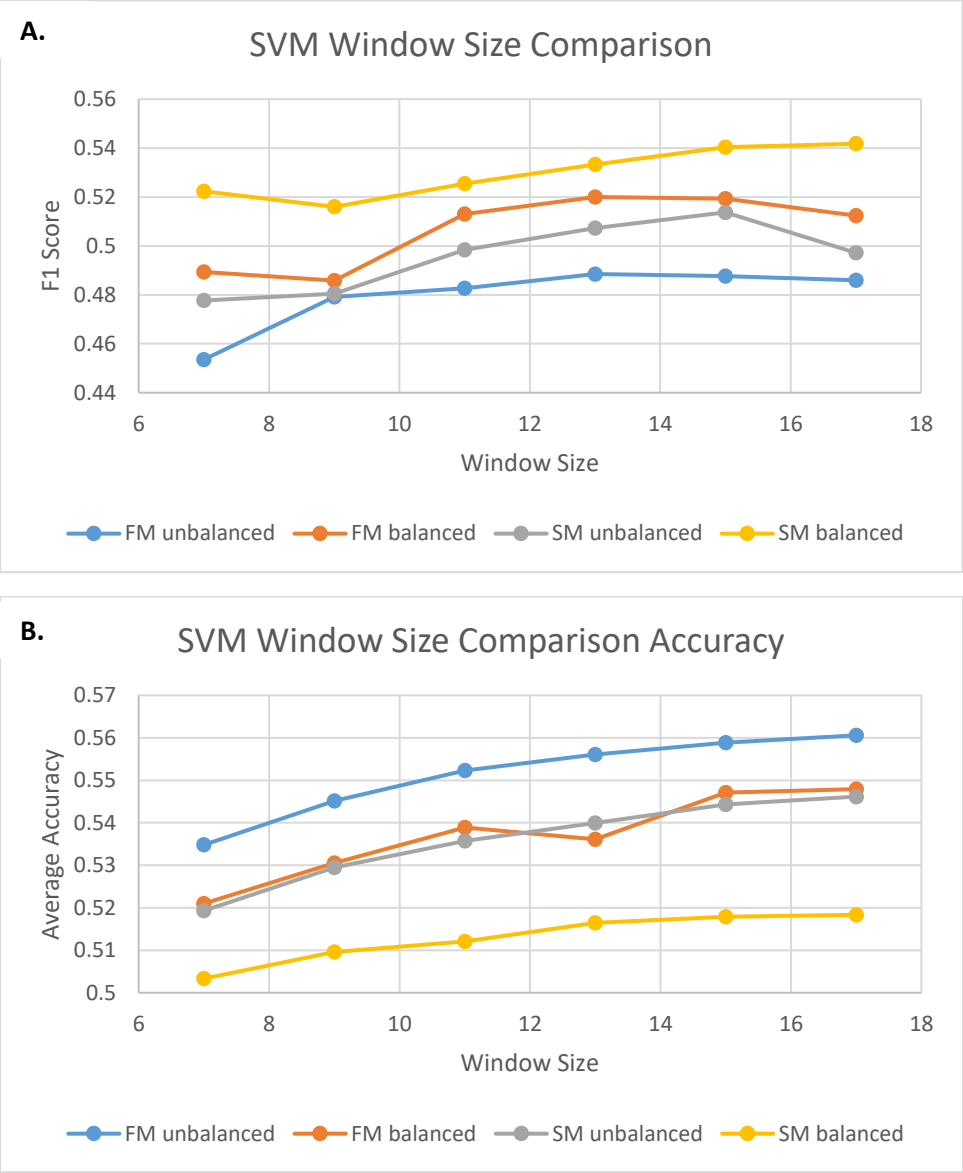


**Figure 2:** Comparison of SVM performance at varying window sizes, using SM or FM PSSM input, and balanced or default class weights for each feature. F1 scores in (A) and average accuracy in (B).

Figure 3 summarizes the performance of SVM, Decision Tree, and Random Forests based prediction models. Both the F1 and Accuracy scores increased by between 0.05-0.10 using the decision tree algorithm, and by between 0.10-0.20 using the random forests algorithm, relative to the SVM algorithm. Additionally, the time it took to build models with the RF and DT algorithms was significantly shorter than with the SVM algorithm. Taken together, these results indicate that DT and RF are both more accurate, as well as faster, than SVM for 8-state secondary structure prediction.

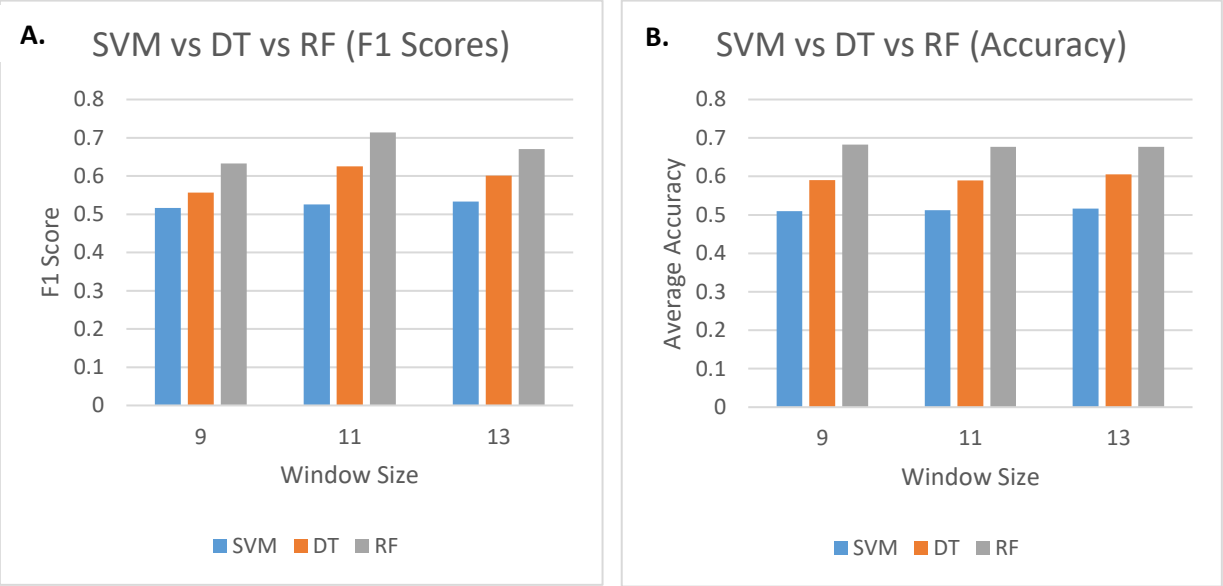


**Figure 3:** Comparison of SVM, decision tree, and random forest machine learning algorithms using SM data and balanced class weights. F1 score in (A) and average accuracy in (B).

To better understand why DT and RF algorithms improve so drastically upon the model performance when compared to the SVM-based model, we compared recall, precision, and F1 scores for all 8 states across models from each of the 3 algorithms in Figure 4. Confusion matrices for the performance of each machine learning algorithm can be found in supplementary Figure S1. From glancing at Figure 4A, the SVM-based predictor performs almost as well as the DT-based predictor for the most common classes: H, E, and C. However, a definite difference between algorithms can be seen for the remaining 5 less common classes, which is most noticeable in G, I, B, and S. Figure 4B confirms this, showing that precision, recall, and F1 scores using the SVM method are comparable to the DT method for H and E, and to a lesser extent C. Similarly, the improvements in precision, recall, and F1 scores in the less common classes when using either of the two non-SVM-based models are shown in Figure 4C. A noteworthy finding here is that, on average, RF-based models achieve much improved precision scores in predicting the 5 less common classes relative to DT-based models, while their recall scores are comparable. Possible explanations for the observed performance differences can be found in the discussion below.

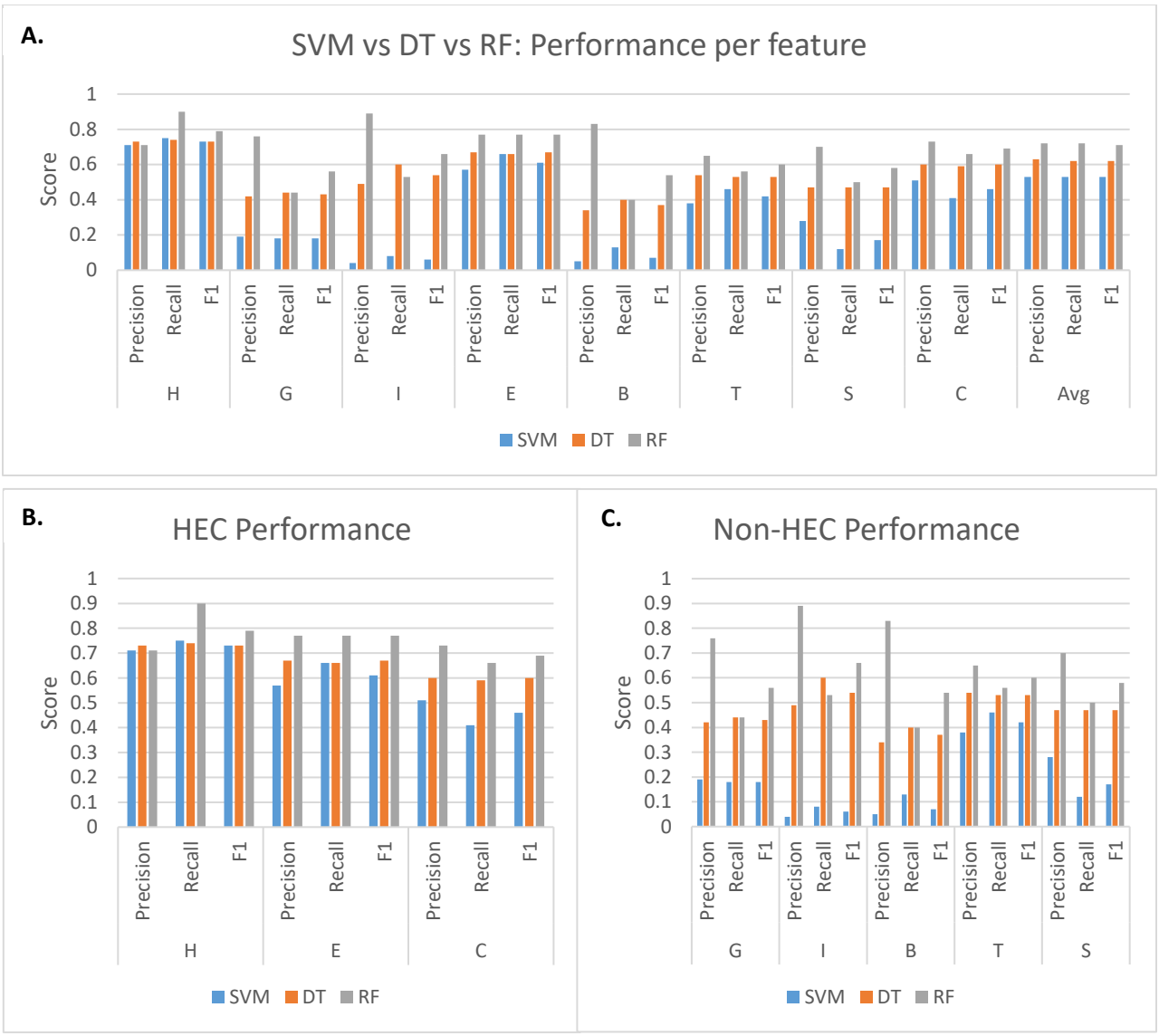


**Figure 4:** Feature-wise precision, recall, and F1 performance for SVM, DT, and RF algorithms. Window size is 11, SM data was used as input, and balanced class weights. (A) All 8 features and averages. (B) Performance on the 3 most common features: H, E, C. (C) Performance on less common features: G, I, B, T, S.

After optimizing our predictor as described above, the performance using each of the 3 machine learning algorithms was compared to 4 currently used state-of-the-art predictors: SSpro, SSpro with template, RaptorX-SS8, and DeepCNF-SS. The metrics for these predictors was taken from a recent paper by Wang et al. (7). The average of each predictor's performance on 5 different reference datasets is plotted, in Figure 5, alongside performances of the 3 predictors we built evaluated using our 718 protein dataset. Q8 accuracy for SSpro, SSpro with template, RaptorX-SS8, and DeepCNF-SS is 64.82, 76.66, 66.14, and 71.94 percent respectively. Our predictors: 8SS SVM, 8SS DT, and 8SS RF have accuracies of 55.23, 61.79, and 68.74 percent respectively. From these data, it appears that while our SVM and DT-based predictors perform slightly worse than currently used predictors, the 8SS RF predictor actually performs quite well, falling directly in the middle of the 4 listed predictors.

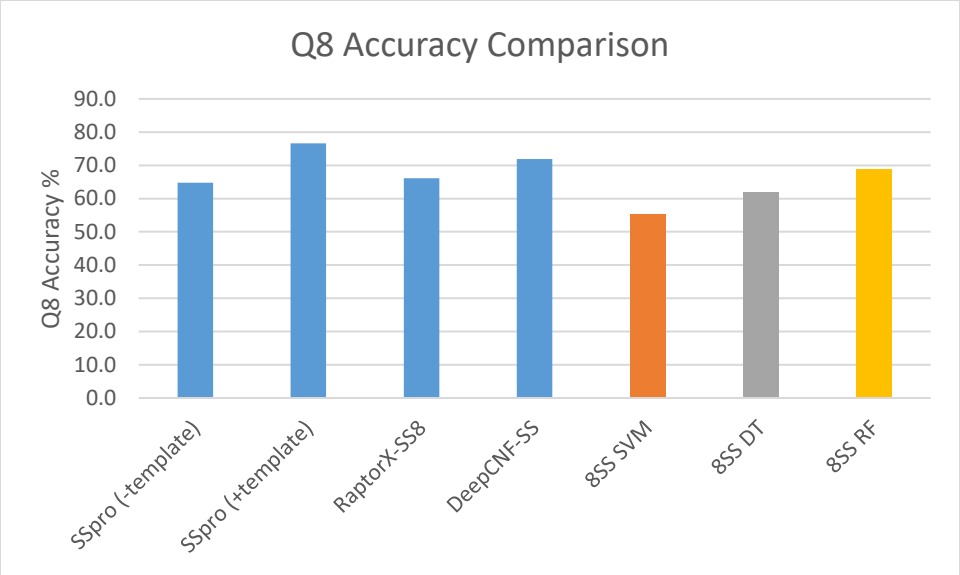


**Figure 5:** Comparison of Q8 accuracy between current state-of-the-art predictors and 8SS using SVM, DT, or RF algorithms.

**Discussion/Future Directions**

During the optimization process for our 8-state secondary structure predictor, 8SS, various parameters were varied to try to improve performance, albeit with varying degrees of success. As was done with other commonly used predictors such as PSI-PRED, we decided to test window sizes between 7-17 amino acid residues to capture the full length of longer alpha-helices, which have an average length of around 10 residues (7). We found there to be a clear trade-off here between performance and time to build the SVM model, with larger window sizes tending to provide better performance but also requiring significantly longer to process. This time increase can be explained by the fact that each additional residue added to the window size adds 20 additional X-value data-points to each window-feature pair. This means that for a dataset of only 200 proteins with an average length of 500 residues, 200*500*20 or 200,000 more data points must be considered.

To address the problem of underrepresented classes in the data such as G or I, a balanced class weight parameter was added to the machine learning algorithm (10). However, though this change did result in improvements to classification of these classes, it also decreased prediction accuracy for the more common classes such as helices, and thus only provided minimal improvements in overall accuracy.

Finally, after initially using normalized frequency matrix data from PSI-BLAST PSSMs to build the SVM-based predictor, we were curious to see if and how performance would be affected with normalized substitution matrix data, as is used by other secondary structure predictors (6). Interestingly, results were mixed as F1 scores increased but accuracy scores decreased on average. It is difficult to explain this seemingly paradoxical result. However, since substitution matrices more accurately reflect the impact of certain amino acid substitutions on the ability to form given secondary structures in a region, one may have expected SM data to improve both accuracy and F1 scores for the predictor. In this case, it appears that the use of SM instead of FM data improves prediction for less common classes while worsening prediction of the most common classes. This would lead to an increased average F1 score across all 8 classes but a decreased average accuracy, as it is calculated per the overall instance of right and wrong

predictions on the complete test set, most of which will be predictions for the common classes H, E, and C.

From the performance metrics displayed in Figures 3 and 4, either decision tree or random forests machine learning algorithms seem to be more suitable for 8-state secondary structure prediction than support vector machines. Figure 3 indicates that overall F1 score and accuracy is increased by 0.05-0.10 when using decision trees, and another 0.05-0.10 when using random forests. After breaking this improvement down into single-class prediction results, we learned from Figure 4 that these two algorithms fare significantly better than SVM in providing accurate predictions of less common secondary structure elements.

SVM algorithms work by clustering training samples in a multi-dimensional space, building decision surfaces to separate each cluster, and then classifying new samples according to which side of these decision surfaces they fall on (11). Therefore, it makes sense that SVMs may struggle to separate similar secondary structure types such as the 3 different types of helices, as they are likely to cluster quite closely together in the same region of the multi-dimensional space. On the other hand, both decision tree and random forest algorithms are based on building classification trees, like phylogenetic trees in biology (12). They classify data along a complex branched structure, with leaves corresponding to classes and branches represent decisions based on the values of certain elements within the input data (12). The main difference is that random forests take the average of multiple decision trees, each of which is built based on a randomly selected subset of information from the training set (13). In this way, random forests are said to reduce overfitting and the resultant high variance that is common to simple decision trees (14).

Due to their differences in approach for multi-class classification, we suggest that one reason for the improved classification performance when using tree-based machine learning algorithms is that this architecture allows for a sort of step-wise separation of the 8-states which is not possible using a single SVM. This means that on the classification tree, each amino acid on the test sequence can be classified as helical or non-helical at an early branch junction, and subsequently all amino acids said to be helical can be branched into one of the 3 helical secondary structures. Furthermore, random forests seem to suffer less from overfitting than decision trees. Across the 5 cross-validation sets evaluated for each model, accuracy score variance was higher in decision tree based models relative to random forest based models (standard deviation of 0.028 compared to 0.015 for window size 9, FM, unbalanced). Taken together, this explains why the different machine learning algorithms rank in the following order, from best to worst: random forests, decision tree, support vector machines.

The performance of our best predictor, using random forests, is quite competitive compared to the other 8-state secondary structure predictors in Figure 5. It is, however, difficult to account for the differences and similarities in performance between our method and the others, as we use random forests and others use a variety of neural networks based approaches to solve the prediction problem (7, 15, 16). An aspect worth comparing in the future is the speed of prediction between random forests and neural networks approaches, especially for longer protein sequences, but no information on prediction speed is currently available for the listed methods. Although the machine learning algorithms are different, methods such as SSpro8 also use 3 iterations of PSI-BLAST on each sequence to obtain evolutionary information about each position prior to prediction (16). As such, using PSI-BLAST appears to be an effective way to improve performance and is an important part of how we have achieved an accuracy of

around 70%. After testing various window sizes, the optimum in terms of accuracy and speed was found to be at a size of 11 residues, which matches the window size used in the DeepCNF method (7).

Interestingly, all of the 4 neural network based methods that are listed in Figure 5 also use additional input information alongside the PSI-BLAST PSSM data. SSpro8 with template uses sequence-based structural similarity, by searching the PDB for any known structures with similar sequences, to help with secondary structure prediction, which may account for it having the highest Q8 accuracy (16). In contrast, RaptorX-SS8 incorporates both physio-chemical amino acid properties and propensity for residues to appear as the end residue of a secondary structure feature to aid prediction (15). Lastly, Deep-CNF input contains both a PSSM input vector, as well as a sparse encoded vector corresponding to the actual amino acid at that position (7). In the future, inclusion of such additional input data alongside our current procedure should be considered to improve 8-state prediction accuracy.

Though the performance of our predictor on the given dataset is comparable among the best currently available 8-state secondary structure predictors, further steps must be taken before this performance is validated and the predictor is ready for public use.

From looking at the complete dataset of 718 proteins used to build and test our predictor, one may question whether the dataset is both big enough and representative of the global protein landscape. This is an important concern to address, as a dataset that is too small and/or biased will not only result in a poor predictor for new protein sequences, but also prohibit an accurate evaluation of predictor performance. To answer this question, we must examine the prediction accuracy of our predictor using several commonly used reference datasets that were used to evaluate the performance of other state-of-the-art predictors, such as CullPDB and CASP10-11 (7). To get the best possible predictor, training should be redone using a proven representative dataset, for example one that contains one protein with known secondary structure from each family of databases such as SCOP or CATH. Another option would be to use datasets previously curated for training of secondary structure prediction, such as the 5000+ protein set used for SSpro8 (16). This will allow the classification model to best predict the structure of any new protein sequence. Taking these additional steps will allow for improvement in prediction, as well as a more accurate comparison of performance between our predictor and others.

Furthermore, it may be possible to integrate further sequence information or use step wise classification models to achieve better 8-state secondary structure prediction. Amino acids have different properties with regards to hydrophobicity, size, charge, and polarity. It has been shown that based on these properties, amino acids are more or less likely to appear in different secondary structure features (17). Therefore, including information about these properties alongside the respective PSSM data for each position may serve to further improve prediction performance. There have also been examples where predictors include the PSSM data along with a second sparse matrix representing the actual amino acid in that position for each input position (7).

Another possibility would be to include a step wise classification scheme, with layered machines, for difficult to distinguish related secondary structures such as the 3 helix types. Step-wise machine learning has been shown to be effective for similar structure prediction methods, such as in SPINE X (18). Such a predictor would first classify into 3 states: H, E, and C, and then proceed to classify the helices into H, G, and I. This scheme may help to better resolve classification conflicts that arise when trying to differentiate between helix types because the differences between helix types would likely become more defined when only residues from helices are considered.

Following further optimization of the method parameters and validation using representative datasets, the predictor presented here may prove to become a promising new approach to the challenging problem of 8-state secondary structure prediction.

**References**


1. Anfinsen, C.B. (1972). The formation and stabilization of protein structure. *Biochemical Journal*, 128(4): 737-749.
2. Chou, P. Y. & Fasman, G. D. (1974). Conformational parameters for amino acids in helical, -sheet, and random coil regions calculated from proteins. *Biochemistry*, 13: 211-222.
3. Garnier, J., Osguthorpe, D. J. & Robson, B. (1978). Analysis and implications of simple methods for predicting the secondary structure of globular proteins. *Journal of Molecular Biology*, 120: 97-120.
4. Kabsch, W. & Sander, C. (1983). A dictionary of protein secondary structure. *Biopolymers*, 22: 2577-2637.
5. Rost, B. & Sander, C. (1993). Prediction of protein secondary structure at better than 70 % accuracy. *Journal of Molecular Biology*, 232: 584-599.
6. Jones D. T (1999). Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292: 195–202.
7. Wang, S., Peng, J., Ma, J., & Xu, J. (2016). Protein Secondary Structure Prediction Using Deep Convolutional Neural Fields. *Scientific Reports*, 6: 18962.
8. Altschul, S. F. et al. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, *25*(17):3389–3402.
9. Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 2(12): 1137–1143.
10. Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825-2830.
11. Cortes, C., Vapnik, V. (1995). Support-vector Networks. *Machine Learning*, 20(3): 273-297.
12. Quinlan, J.R. (1986). Induction of Decision Trees. *Machine Learning*, 1: 81-106.
13. Ho, T.K. (1995). Random Decision Forests. *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, 278-282.
14. Kleinberg, E. (1996). An Overtraining-Resistant Stochastic Modeling Method for Pattern Recognition. *Annals of Statistics*, 24(6): 2319-2349.
15. Wang, Z., Zhao, F., Peng, J., & Xu, J. (2011). Protein 8-class secondary structure prediction using conditional neural fields. *Proteomics*, *11*(19): 3786–3792.
16. Magnan, C. N., & Baldi, P. (2014). SSpro/ACCpro 5: almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity. *Bioinformatics*, *30*(18): 2592–2597.
17. Costantini, S., Colonna, G., Facchiano, A.M. (2006). Amino Acid Propensities for Secondary Structures are Influenced by the Protein Structural Class. *Biochemical and Biophysical Research Communications*, 342(2): 441-451.
18. Faraggi, E., Zhang, T., Yang, Y., Kurgan, L., & Zhou, Y. (2012). SPINE X: Improving protein secondary structure prediction by multi-step learning coupled with prediction of solvent accessible surface area and backbone torsion angles. *Journal of Computational Chemistry*, *33*(3): 259–267.

**Supplementary Data**

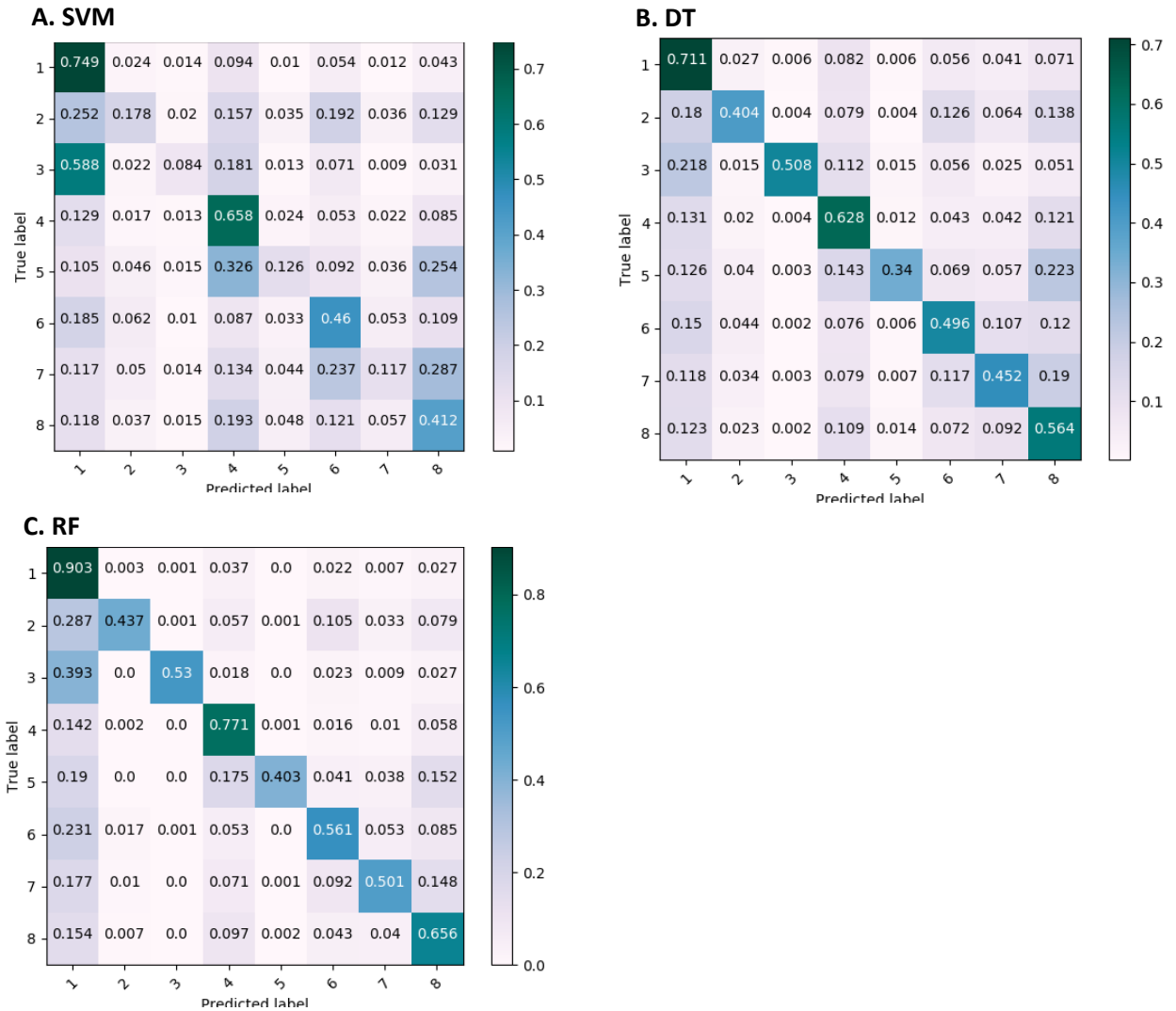


**Figure S1:** Confusion matrices for predictions from the 3 different machine learning algorithms used. Window size 11, SM data as input, balanced class weights. Features are as follows: 1:H, 2:G, 3:I, 4:E, 5:B, 6:T, 7:S, 8:C. (A) is the SVM-based predictor. (B) is the DT-based predictor. (C) is the RF-based predictor.