Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: XXXXXXX
Review

OXFORD

Sequence analysis

Random Forest Approach for Prediction of Eukaryotic Protein Subcellular Location

Udeepa Sashith Meepegama *

Department of Computer Science, University College London

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on 6 April 2020; revised on XXXXXXX; accepted on XXXXXXX

Abstract

Motivation: Investigations into protein function as well as therapeutic interventions both require an understanding of protein subcellular localisation. This topic has been well-studied in bioinformatics and has led to the development of several machine learning algorithms. More recently deep learning methods such as recurrent neural networks have been employed to handle these sequences. However, deep learning architectures continue to be treated as black-box function approximators. Thus it is desirable to have methods for predicting protein locations whilst maintaining a degree of interpretability.
Results: In this paper we train a Random Forest classifier on a selection of features derived from amino acid composition and physico-chemical properties. We benchmark our model against other traditional machine learning models, out-performing them with an accuracy of 67.74%. Additionally, to improve interpretability of our results we analyse the feature importance derived from our Random Forest classifier.
Availability: The code base is available at `https://github.com/udeepam/proteios`
Contact: udeepa.meepegama.19@ucl.ac.uk
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Proteins fulfil a wide variety of functions inside the morphologically and functionally distinct compartments of eukaryotic cells. The compartment/organelle where the protein is located influences the proteins function by controlling access to and availability of molecular interaction partners and the post-translational modification machinery Li *et al.* (2012a). It has be shown that abnormal protein subcellular localisation can affect the function of a protein and contributes to the pathogenesis of many human diseases, such as cancer as well as neurodegenerative and cardiovascular diseases Hung and Link (2011). Therefore, predicting the subcellular location of proteins is an essential task for obtaining insight into their functions as well as aiding emerging therapeutic strategies that target protein localisation.

The proteins location can be determined by biological experiments, but these methods are laborious, time-consuming and expensive. Additionally, these manual methods are not scalable to the large amounts of raw sequence data available. Hence, a fully automatic and reliable prediction system for protein subcellular localisation is highly desirable.

Many machine learning methods have been developed for this task. These methods largely fall into two categories: i) converting the protein sequence into a fixed length representation as input for a downstream classifier, or ii) processing the amino acid positions in the protein sequence sequentially.

In the first category, feature engineering is employed to extract a fixed number of features from the protein sequences as input to a classifier. These features can be derived from amino acid composition and related physico-chemical properties. Reinhardt and Hubbard (1998) uses the amino acid composition of the protein sequence to train a multi-layer perceptron (MLP) with an input dimension of twenty, one node for each amino acid fraction. Yu *et al.* (2006) present a multi-class support vector machine (SVM) classifier which predicts subcellular localisation based on amino acid composition, partitioned amino acid composition, di-peptide composition, and sequence composition based on the physico-chemical properties of amino acids.

The second category, utilises sequence-based models which can learn and make inference from varying length inputs. Sønderby *et al.* (2015) uses a deep learning architecture consisting of long-short term memory (LSTM) cells, attention units and convolutional layers. The LSTM sequentially reads the sequence into memory cells, the attention mechanism detects sorting signals and the convolutional layer extracts short motifs. Almagro Armenteros *et al.* (2017) build upon this deep learning architecture

and propose a hierarchical tree likelihood mimicking the biology of the sorting pathway. Although neural networks provide a means for powerful function approximation, a problem with these models is that they lack interpretability Gilpin *et al.* (2018).

Homology-based predictors are an alternative computational approach, which compare the localisation of unknown proteins with known proteins by using sequence similarity search methods such as BLAST Altschul *et al.* (1990). If a degree of similarity is found in the sequence, then it can be inferred that the subcellular location of the unknown protein is similar to known protein Wan *et al.* (2012). These method are appropriate for annotated proteins or proteins with annotated close homologues. However, they have a reduced chance of predicting the consequences of mutations affecting sorting signals. Hence, these methods are often integrated with other methods, Goldberg *et al.* (2014) introduce LocTree3 which combines a hierarchical system of SVMs with annotation transfer from close homologs with experimentally annotated localisation through PSI-BLAST Altschul *et al.* (1997).

In this paper we follow the methodology of the first class of machine learning methods, specifically training a Random Forest (RF) classifier Liaw *et al.* (2002) on 89 features derived from the amino acid compositions and related physico-chemical properties of the protein sequences. A RF classifier is chosen to balance classification accuracy with interpretability. The goals of this paper are twofold:

1. Evaluate the RF classifier against other traditional machine learning algorithms for predicting subcellular localisation,
2. Investigate and analyse the important features used by the RF classifier to aide in interpretability.

2 Materials and methods

2.1 Random Forest

Here we describe the basic ideas behind the Random Forest (RF) classifier for multiclass classification.

Decision Tree

Given input vectors and corresponding labels $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)\} \in (\mathbb{R}^n, \{0, \ldots, K-1\})^m$, a decision tree recursively partitions the input space into a set of hyper-rectangles such that samples with the same labels are grouped together.

For a parent region R_b (i.e. at node b), each candidate split has a feature j and threshold $s_b \in \mathbb{R}$. We partition the data into two child regions $R_1(j, s_b)$ and $R_2(j, s_b)$ given by,

$$R_1(j, s_b) = \{\mathbf{x} \mid x_j \leq s_b, \mathbf{x} \in R_b\}, \tag{1a}$$

$$R_2(j, s_b) = \{\mathbf{x} \mid x_j > s_b, \mathbf{x} \in R_b\}. \tag{1b}$$

The impurity at node b is computed as,

$$G(R_b, j, s_b) = \frac{n_1}{N_b} H\Big(R_1(j, s_b)\Big) + \frac{n_2}{N_b} H\Big(R_2(j, s_b)\Big), \tag{2}$$

where $H(\cdot)$ is the impurity function, N_b is the number of training examples in parent region R_b and n_1 is the number of training examples in the child region $R_1(j, s_b)$. We then select the parameters that minimises the impurity,

$$j^*, s_b^* = \underset{j, s_b}{\operatorname{argmin}}\, G(R_b, j, s_b), \tag{3}$$

where for each split the possible values j can take is chosen from a random subset of size $\sqrt{n}$ features. In order to build the decision tree, we recursively assign training examples to the child regions and repeat the above steps for each region.

Impurity Function

For node b that represents a region R_b with N_b training examples, the empirical class probability is given by,

$$p_{bk} = \frac{1}{N_b} \sum_{x_i \in R_b} \mathbb{I}[y_i = k], \tag{4}$$

i.e. the proportion training examples in region R_b that are of class k. A common measure of impurity that we use is the `Gini` impurity measure,

$$H(R_b) = \sum_{k=0}^{K-1} p_{bk}(1 - p_{bk}). \tag{5}$$

Stopping Criterion for Decision Tree

We now require a stopping criteria to determine when to halt the growth of a tree to prevent overfitting to the training data. In our case the tree nodes expand until all leaves are pure (i.e. $N_b = 1$) or until all leaves contain less than the minimum number of samples for an internal node to split. Thus these criterion's are equivalent. This approach will lead to overfitting of the training data however our classifier is not a single decision tree but a Random Forest and we will see that by introducing randomness in the feature selection we are able to mitigate overfitting.

Prediction for Decision Tree

We classify a test point $\mathbf{x}_{test}$ which falls into the region b into the class with maximum probability,

$$\hat{y}(\mathbf{x}_{test}) = \underset{k}{\operatorname{argmax}} \sum_{b=1}^{B} p_{nk} \mathbb{I}[\mathbf{x}_{test} \in R_b]. \tag{6}$$

Random Forest

A Random Forest is an ensemble of decision trees, where each tree is built using bootstrapping (samples drawn with replacement) from the training set. Additionally, we use a random subset of features when splitting the nodes (i.e. in Eq.(3)). We use these two principles because by training many decision trees on different samples using random features, each tree may have high variance w.r.t. a particular set of the training data, however, the entire forest will have lower variance but not at the cost of increasing the bias.

2.2 Dataset

We are provided the protein datasets in `FASTA` format and so use the `BioPython` package to read in the sequences. The training data consists of 9222 sequences which are mapped to 4 main subcellular locations; cytosolic (3004), secreted (1605), nuclear (3314) and mitochondrial (1299). As can be seen, there exists a class imbalances in our dataset. Every sequence in the training set is assumed to be unique (non-homologous). This is a necessary criteria Hobohm *et al.* (1992), as measured test performance should be a true measure of the predictive performance of a new protein and not a measure of the ability of a method to find homologues with the same subcellular location. The 'blind' test dataset consists of 20 mutated protein sequences.

Subcellular location	Number of proteins
Cytosolic	2924
Secreted	1586
Nuclear	3248
Mitochondrial	1297
Total	9055

Table 1. Number of proteins in each of the 4 subcellular locations in the training dataset after preprocessing.

Before extracting features from the sequences, we employ a preprocessing step, where we remove protein sequences with lengths greater than 2000 amino acids. By analysing the sequence lengths we found the range of the distribution to be $10 - 16,000$ amino acids, the bulk of the distribution lying between $10 - 2000$ amino acids with a mean of 300 amino acids. Thus by removing protein sequences with lengths greater than 2000 we remove anomalous lengths that would heavily scale down the sequence length feature when normalising. This results in 167 proteins sequences being eliminated from the original 9222.

Additionally, the amino acid codes accepted by the `BioPython` package cannot be ambiguous. Thus we replace each occurrence of the ambiguous code 'X' by drawing from a list of non-ambiguous codes with uniform probability. For each occurrence of the ambiguous code 'B' we replace with either aspartate 'D' or asparagine 'N' with equal probability. An issue with the `BioPython` package is that it has not been updated to analyse selenocysteine 'U'. Hence we replace every occurrence of selenocysteine with cysteine 'C' as these amino acids share a similar structure. This preprocessing step is applied to both the training dataset and the 'blind' test dataset. The mapped subcellular locations and the number of proteins in each main location after preprocessing is summarised in Table (1).

2.3 Extraction of Features

For feature extraction we use the `ProtParam` module from the `BioPython` package, which provides us with a range of methods for computing properties from amino acid compositions. All the features used are listed in Table (2).

The features with dimensions greater than 1 are the percentage count of the standard 20 amino acids found in genetic code. The secondary structure fraction corresponds to the fraction of amino acids in the helix, turn and sheet, and the molar extinction coefficient is calculated twice assuming first cysteines (reduced) and second cystines residues (Cys-Cys-bond).

We note that for subcellular localisation most of the information lies in the beginning (N-terminus) and end (C-terminus) of the sequence. Thus we calculate all the features listed below globally and a subset on the first 50 and last 50 amino acids in the sequence.

Features	Dimension	Global	Local
Amino acid percentage	20	✓	✓
Secondary structure fraction	3	✓	
Molar extinction coefficient	2	✓	
Sequence length	1	✓	
Molecular weight	1	✓	
Isoelectric point	1	✓	
Aromaticity	1	✓	
Instability index	1	✓	
GRAVY	1	✓	
Charge at pH 1	1	✓	
Charge at pH 7	1	✓	
Charge at pH 12	1	✓	
Hydrophobicity	1	✓	✓
Hydrophilicity	1	✓	✓
Flexibility	1	✓	✓
Emini surface accessibility	1	✓	✓
Janin surface accessibility	1	✓	✓

Table 2. Features extracted from the protein sequences. We include the dimension of the feature as well as if it is extracted for the whole amino acid or for the first 50 and last 50 amino acids in the sequence.

The final five features listed in Table (2) are computed by taking the mean of the produced profile using the amino acid scales; hydrophobicity Kyte and Doolittle (1982), hydrophilicity Hopp and Woods (1981), flexibility Vihinen *et al.* (1994), Emini surface fractional probability Kaptein *et al.* (1997) and Janin interior to surface transfer energy scale Kaptein *et al.* (1997). These profiles are calculated globally over the entire amino acid, as well as locally for the first 50 and last 50 amino acids.

In total we obtain 89 features. The features that are not percentages are normalised using the standard procedure of subtracting the mean and dividing by the standard deviation, such that the mean of each feature is 0 and the variance is 1. These transformation obtained from the training dataset are saved and applied to the 'blind' test data. The features that are percentages are scaled up by a constant (in our case 5) such that the values of all features are on the same scale. We do this so as to map the features to a common scale, without distorting differences in the ranges of values Guyon and Elisseeff (2006). This is particularly important for algorithms sensitive to scaling such as SVMs.

2.4 Baseline models

We compare our model, the RF classifier against other traditional classifiers that are used in machine learning: MLP Pal and Mitra (1992), SVM using a radial basis function (RBF) kernel Hsu *et al.* (2003), Gaussian Naïve Bayes (GNB) John and Langley (2013), linear discriminant analysis (LDA) Balakrishnama and Ganapathiraju (1998) and a k-nearest neighbour classifier (KNN) Guo *et al.* (2003). Below we outline briefly rhe assumptions made by the baseline models.

A MLP is a non-parametric, deterministic model consisting of at least three layers of neurons: an input and an output layer with one or more hidden layers. MLPs are a class of fully connected neural networks trained using backpropagation.

A SVM is a non-parametric, deterministic model which projects the original training data to a higher dimensional feature space via a Mercer kernel operator, to then compute the maximal margin hyperplane. The choice of kernel function leads to more complex decision boundaries in the original space. The kernel function we choose is the RBF kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\big(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\big)$, where σ is the bandwidth.

A GNB is a parametric, generative model which makes the "naïve" assumption that the features are conditionally independent given the class label.

A LDA is a parametric, generative model that makes two assumptions. Firstly, the conditional probability density functions are all normally distributed. Secondly, homoscedasticity is assumed with the covariance matrices having full rank.

A KNN is a non-parametric, discriminative model that simply searches for the k nearest data points in the training set to the test input. It counts how many members of each class are in this set and returns the empirical fraction as an estimate.

2.5 Experiments

Two sets of experiments were carried out. The first experiment is a grid-search over the possible hyperparameter values for each model using 5-fold cross-validation. The data is first shuffled and stratified folds are used, such that each of the 5 splits are made by preserving the percentage of samples for each class. This is important as the models will then be trained with the same class imbalances present in the validation set. We found that in literature the Jackknife cross-validation test Li *et al.* (2012b) is used instead, a variant of the leave-one-out cross-validation method. However, due to computational constraints we were not able to follow this procedure.

The grid-search for the RF classifier is run over the following hyperparameters; number of trees in the forest $[50, 100, 400, 800, 1000,$

Model	Accuracy (%)	Precision	Recall	F1-Score	MCC
RF	$\mathbf{67.74 \pm 0.96}$	0.6948 ± 0.0105	$\mathbf{0.6994 \pm 0.0050}$	$\mathbf{0.6970 \pm 0.0067}$	$\mathbf{0.5449 \pm 0.0131}$
MLP	65.09 ± 0.74	0.6752 ± 0.0093	0.6723 ± 0.0079	0.6729 ± 0.0082	0.5125 ± 0.0096
SVM	67.41 ± 1.02	$\mathbf{0.6995 \pm 0.0084}$	0.6931 ± 0.0072	0.6959 ± 0.0078	0.5445 ± 0.0137
GNB	54.10 ± 0.40	0.5628 ± 0.0051	0.5817 ± 0.0030	0.5550 ± 0.0027	0.3847 ± 0.0070
LDA	61.21 ± 1.02	0.6366 ± 0.0066	0.6267 ± 0.0113	0.6301 ± 0.0086	0.4571 ± 0.0155
KNN	61.26 ± 0.60	0.6451 ± 0.0057	0.6219 ± 0.0048	0.6287 ± 0.0048	0.4596 ± 0.0083

Table 3. Averaged results from 5-fold cross-validation, comparing the performance of our Random Forest classifier against the baseline models.

$1500, 2000]$, minimum number of samples required to split an internal node $[2, 4, 10, 12, 16]$ and the minimum number of samples required to be at a leaf node $[1, 5, 10]$. The hyperparameter values that produce the highest averaged classification accuracy is used for the final model. The final test results presented is the averaged performance on the validation sets of the best model during 5-fold cross validation.

For the second experiment, we retrain our RF classifier with the best parameter values on the full training dataset. We then test on the 'blind' test dataset, such that the output is a measure of confidence of the assigned class. For the RF classifier, the predicted class probabilities of an input sample is computed as the mean predicted class probabilities of the trees in the forest. The class probability of a single tree is the fraction of samples of the same class in a leaf.

All the models are implemented in Python 3.6.8 using the machine learning library `Scikit-learn` 0.22.2.

2.6 Assessment of prediction performance

The performance of each model is evaluated by the following set of metrics; recall, precision, F1-score, accuracy, and multiclass Matthew Correlation Coefficient (MCC) Gorodkin (2004). The first three are defined in the binary classification setting as,

$$Recall = \frac{TP}{TP + FN}, \tag{7}$$

$$Precision = \frac{TP}{TP + FP}, \tag{8}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}, \tag{9}$$

respectively. True positives (TP) denotes the number of times the method correctly predicted the label as positive. Whilst, false positives (FP) and false negatives (FN) denote the number of misclassified positive and negative instances, respectively. In the multiclass setting, these metrics are calculated per class and their unweighted mean is taken. The multiclass MCC is defined as,

$$MCC = \frac{\sum_{k,l,m} C_{kk}C_{lm} - CklC_{mk}}{\sqrt{\sum_{k}(\sum_{l} C_{kl})(\sum_{k' \neq l,l'} C_{k'l'})}\sqrt{\sum_{k}(\sum_{l} C_{lk})(\sum_{k' \neq l,l'} C_{l'k'})}} \tag{10}$$

where C is the confusion matrix.

3 Results

3.1 Comparison to baseline models

In Table (3), we compare the performance of our RF classifier against the baseline models. From the grid search our final models hyperparameters are,

1. 1000 trees in the forest,
2. 2 samples required to split an internal node,
3. 1 sample required to be a leaf node.

We see that RF classifier achieves the highest performance in predicting the subcellular localisation with an accuracy and MCC score of 67.74% and 0.5449, respectively. This can be attributed to the fact that the RF classifier performs well with high dimensional data as each tree works with a subset. The next closest model is the SVM which achieves comparable performances in accuracy and MCC score, and in fact beats our RF classier in precision with a score of 0.6995. The level of performance demonstrated by the SVM is to be expected as the RBF kernel is able to create a flexible decision boundary in the projected high dimensional space.

One the other hand, we observe that the other baseline models perform comparatively worse, with the GNB obtaining the lowest accuracy of 54.10%. A possible reason for this could the naïve assumption made by the GNB, that the features are conditionally independent given the class label, does not hold with this dataset. The performances of the LDA and KNN classifiers are comparable, achieving accuracies of 61.21% and 61.26%, respectively. Additionally we note that the standard deviations of the results presented in Table (3) are small. This could be a result of the stratified folding such that the training and validation set, during 5-fold cross-validation, have class distributions representative of the true distribution of the whole dataset.

3.2 Random Forest

From Table (3) we identified the RF classifier as the best performing model in terms of accuracy and MCC score. We now further investigate the performance of our model in the above experiment by analysing the evaluation metrics on a class-level basis, as seen in Table (4).

Class	Precision	Recall	F1-Score
Cytosolic	0.5819 ± 0.0256	0.5677 ± 0.0192	0.5745 ± 0.0186
Secreted	0.8031 ± 0.0149	0.8165 ± 0.0108	0.8097 ± 0.0101
Mitochondrial	0.7418 ± 0.0372	0.6368 ± 0.0218	0.6845 ± 0.0171
Nuclear	0.6522 ± 0.0042	0.6967 ± 0.0277	0.6736 ± 0.0146

Table 4. Averaged performance for the Random Forest classifier on a class-level basis.

We first note the observation that the F1-score for the cytosolic class is significantly lower than the secreted and mitchondrial classes. This is surprising as from Table (1) we see that there is a class imbalance in our dataset, such that there is twice as many cytosolic protein sequences as there are secreted and mitochondrial. With more data of one class it should be easier for the classifier to learn a better decision boundary. However, this is not the case and could be attributed to the selection of features. We can infer that the features chosen are not completely informative for identifying protein sequences in the cytosolic class, resulting in a lower precision and recall score. On the other hand, it would appear that the feature engineering process has produced very informative features for

identifying the secreted class, as the RF classifier achieves a F1-score of 0.8097, which is significantly higher than the other classes.

Another interesting observation, this time regarding the mitcochondrial class, is the large difference in precision and recall score, 0.7418 and 0.6368 respectively. This indicates that there are fewer false positives than false negatives, i.e. the RF classifier more frequently labels a protein sequence that is in the mitochondrial class incorrectly.

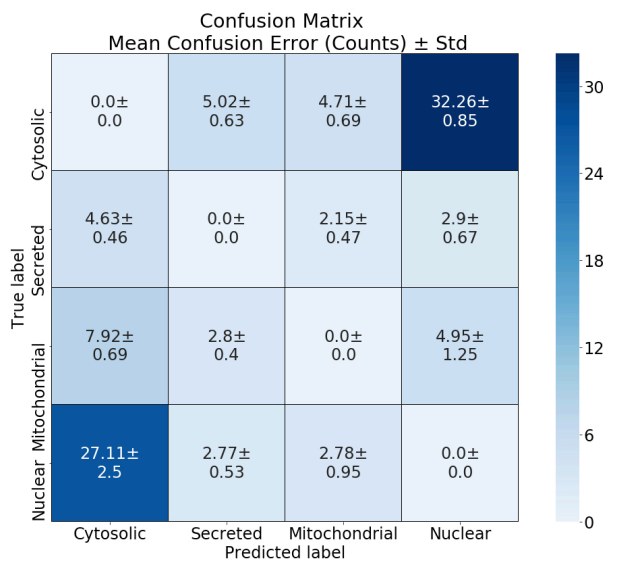

Fig. 1. Confusion matrix for the Random Forest classifier.

To further investigate the classification error produced by our RF classifier we analyse the resulting confusion matrix, where each element is a count of the number of times the true label was mistakenly predicted as another label. We produce a confusion matrix for each of the 5 folds and then average to obtain a 4×4 matrix where each cell contains a mean confusion error and its standard deviation as shown in Figure (1). From this figure we see that the highest confusion error comes from incorrectly classifying a protein sequence as nuclear when it is cytosolic, a confusion error of 32.26. Whilst the second highest confusion error comes from the opposite, incorrectly classifying a protein sequence as cytosolic when it is nuclear, a confusion error of 27.11. This again is surprising as both classes are represented more in the training dataset. But again this could be due to the selection of features which are not completely informative for distinguishing between sequences in these classes. In comparison the mean confusion errors for the secreted and mitochondrial classes are relatively low.

Sequence Identifier	Class	Confidence
SEQ677	Secr	Confidence 40%
SEQ231	Secr	Confidence 57%
SEQ871	Secr	Confidence 55%
SEQ388	Nucl	Confidence 71%
SEQ122	Nucl	Confidence 69%
SEQ758	Nucl	Confidence 80%
SEQ333	Cyto	Confidence 43%
SEQ937	Cyto	Confidence 67%
SEQ351	Cyto	Confidence 45%
SEQ202	Mito	Confidence 73%
SEQ608	Mito	Confidence 79%
SEQ402	Mito	Confidence 69%
SEQ433	Nucl	Confidence 30%
SEQ821	Secr	Confidence 85%
SEQ322	Nucl	Confidence 90%
SEQ982	Nucl	Confidence 84%
SEQ951	Nucl	Confidence 46%
SEQ173	Nucl	Confidence 54%
SEQ862	Mito	Confidence 67%
SEQ224	Cyto	Confidence 37%

Table 5. Results table for our Random Forest classifier predicting on the 'blind' test dataset. Formatted as required.

Finally, in Table 5 we present our results from testing our RF classifier on the 'blind' test data. For each test sequence, our model outputs a probability estimate for each class. The class label with the maximum probability (i.e. maximum confidence) is presented. It is worth noting that an uncertain estimate would place a uniform probability over all the classes resulting in a confidence of 25%. From the table we see that our RF classifier is not overly confident in its predictions. This is encouraging as the model achieves an accuracy of 67.74% during the 5-fold cross-validation, and so it would be questionable if it was predicting every class labels with high confidence. Thus our model can be considered as well-calibrated, as it predicts class labels that it is certain of with high confidence and those that it is uncertain of with low confidence.

4 Discussion

In this paper we have evaluated and compared the performance of our RF classifier against other traditional machine learning classifiers, for the task of classifying protein sequences to their subcellular locations. From Table 4 and Figure 1 we provided an explanation for the large classification error of the cytosolic and mitochondrial classes, as being the result of the selection of features. We now analyse the 89 features used in our study.

One of the advantages of working with a RF classifier is that we can easily obtain feature importance by computing the mean decrease impurity. Every node in the decision trees are a condition on a single feature, designed to split the dataset, so that similar values of the dependent variable end up in the same set. The optimal condition is chosen based on the impurity measure. When training a decision tree, we compute how much each feature decreases the weighted impurity in the tree. For a ensemble of trees, the impurity decrease from each feature can be averaged and the features are ranked according to this measure. This can be interpreted as a measure of feature importance.

In Figure 2 we rank the 89 features according to their computed importance, where the importance sums to 1. From this figure we see that the most important features for the subcellular localisation of proteins using the RF classifier are the average profiles computed using protein scales for the first 50 amino acids, the sequence length, molecular weight as well

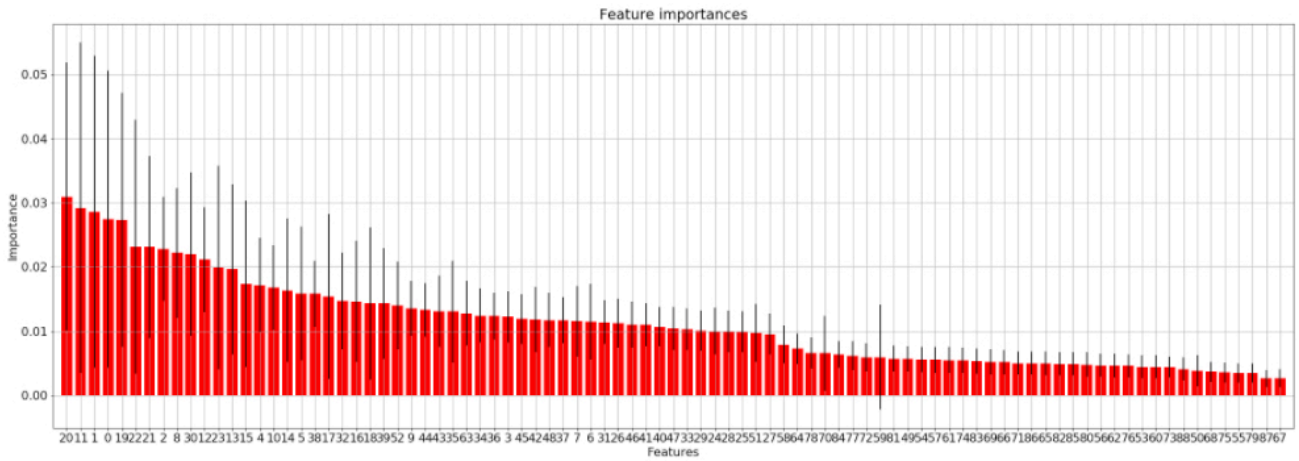

Fig. 2. Feature importance with standard deviation calculated from inter-tree variability. The mapping from index to feature can be found in Appendix 1.

as the charge at pH 1. The physico-chemical properties of the N-terminus being important is to be expected as this is where most of the information of a protein sequence resides. However, it is interesting that the physico-chemical properties of the C-terminus do not appear near the top of the list. The most important of these physico-chemical properties are the structural flexibility, which enables the proteins to move and is associated with various biological processes Yaseen *et al.* (2016). As well as the hydrophobicity of the amino acids. It is also interesting that the charge at pH 1 is an important feature used by the RF classifier for distinguishing between the classes, eluding to a connection between subcellular localisation and the charges of proteins at specific pH's.

We also observe that the features the RF classifier deem less important are the features regarding amino acid percentages, both locally and globally. This may be because most of the information provided by the amino acid percentages are condensed by the physico-chemical features.

5 Conclusion

In this paper, a RF classifier is investigated to predict subcellular location of eukaryotic proteins. Our model is well-calibrated and out-performs other traditional machine learning algorithms, achieving an accuracy of 67.74%. Furthermore, we analysed the features utilised by the RF classifier and found that interestingly, high importance is given to the charge at pH 1 feature. In contrast the C-terminus physico-chemical features are deemed less important.

References

Almagro Armenteros, J. J. *et al.* (2017). Deeploc: prediction of protein subcellular localization using deep learning. *Bioinformatics*, **33**(21), 3387–3395.

Altschul, S. F. *et al.* (1990). Basic local alignment search tool. *Journal of molecular biology*, **215**(3), 403–410.

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.

Balakrishnama, S. and Ganapathiraju, A. (1998). Linear discriminant analysis-a brief tutorial. *Institute for Signal and information Processing*, **18**, 1–8.

Gilpin, L. H. *et al.* (2018). Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE.

Goldberg, T. *et al.* (2014). Loctree3 prediction of localization. *Nucleic acids research*, **42**(W1), W350–W355.

Gorodkin, J. (2004). Comparing two k-category assignments by a k-category correlation coefficient. *Computational biology and chemistry*, **28**(5-6), 367–374.

Guo, G. *et al.* (2003). Knn model-based approach in classification. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 986–996. Springer.

Guyon, I. and Elisseeff, A. (2006). An introduction to feature extraction. In *Feature extraction*, pages 1–25. Springer.

Hobohm, U. *et al.* (1992). Selection of representative protein data sets. *Protein Science*, **1**(3), 409–417.

Hopp, T. P. and Woods, K. R. (1981). Prediction of protein antigenic determinants from amino acid sequences. *Proceedings of the National Academy of Sciences*, **78**(6), 3824–3828.

Hsu, C.-W. *et al.* (2003). A practical guide to support vector classification.

Hung, M.-C. and Link, W. (2011). Protein localization in disease and therapy. *Journal of cell science*, **124**(20), 3381–3392.

John, G. H. and Langley, P. (2013). Estimating continuous distributions in bayesian classifiers. *arXiv preprint arXiv:1302.4964*.

Kaptein, R. *et al.* (1997). *Biomolecular Structure and Dynamics: Recent Experimental and Theoretical Advances*, pages 189–209. Springer Netherlands, Dordrecht.

Kyte, J. and Doolittle, R. F. (1982). A simple method for displaying the hydropathic character of a protein. *Journal of molecular biology*, **157**(1), 105–132.

Li, G.-Z. *et al.* (2012a). Multilabel learning for protein subcellular location prediction. *IEEE transactions on Nanobioscience*, **11**(3), 237–243.

Li, L. *et al.* (2012b). An ensemble classifier for eukaryotic protein subcellular location prediction using gene ontology categories and amino acid hydrophobicity. *PLoS One*, **7**(1).

Liaw, A. *et al.* (2002). Classification and regression by randomforest. *R news*, **2**(3), 18–22.

Pal, S. K. and Mitra, S. (1992). Multilayer perceptron, fuzzy sets, classifiaction.

Reinhardt, A. and Hubbard, T. (1998). Using neural networks for prediction of the subcellular location of proteins. *Nucleic acids research*, **26**(9), 2230–2236.

Sønderby, S. K. *et al.* (2015). Convolutional lstm networks for subcellular localization of proteins. In *International Conference on Algorithms for Computational Biology*, pages 68–80. Springer.

Vihinen, M. *et al.* (1994). Accuracy of protein flexibility predictions. *Proteins: Structure, Function, and Bioinformatics*, **19**(2), 141–149.

Wan, S. *et al.* (2012). Goasvm: Protein subcellular localization prediction based on gene ontology annotation and svm. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2229–2232. IEEE.

Yaseen, A. *et al.* (2016). Flexc: protein flexibility prediction using context-based statistics, predicted structural features, and sequence information. *BMC bioinformatics*, **17**(8), 281.

Yu, C.-S. *et al.* (2006). Prediction of protein subcellular localization. *Proteins: Structure, Function, and Bioinformatics*, **64**(3), 643–651.

1. We use the `BioPython` package and specifically the `Seq` module to read in the protein sequences in FASTA format.
2. We implement a simple function `preprocess` which removes sequences of length greater than 2000 amino acids and replaces amino acids 'X', 'B' and 'U' as described in Section 2.2.
3. We use the methods from the `ProtParam` module of the `BioPython` package to extract features from the preprocessed sequences.
4. We then use the `scikit-learn` library to implement the models, `RandomForestClassifier`, `MLPClassifier`, `SVC`, `GaussianNB`, `LinearDiscriminantAnalysis` and `KNeighborsClassifier`.
5. To do hyperparameter search as well as5-fold cross-validation we use the functions `StratifiedKFold`, `GridSearchCV` from the `scikit-learn` library.
6. To evaluate the models we use the `metrics` module of the `scikit-learn` library.

1 Feature Labels

Here we present the mapping from index to feature as used in Figure (2),

- 0: sequence length,
- 1: molecular weight,
- 2: isoelectric point,
- 3: aromaticity,
- 4: instability index,
- 5: GRAVY,
- 6: molecular extinction coefficient cysteines (reduced),
- 7: molecular extinction coefficient cysteines residues
- 8: secondary structure fraction (helix),
- 9: secondary structure fraction (turn),
- 10: secondary structure fraction (sheet),
- 11: charge at pH 1,
- 12: charge at pH 7,
- 13: charge at pH 12,
- 14: global hydrophobicity,
- 15: global flexibility,
- 16: global hydrophilicity,
- 17: global Emini surface accessibility,
- 18: global Janin surface accessibility,
- 19: first 50 amino acids hydrophobicity,
- 20: first 50 amino acids flexibility,
- 21: first 50 amino acids hydrophilicity,
- 22: first 50 amino acids Emini surface accessibility,
- 23: first 50 amino acids Janin surface accessibility,
- 24: last 50 amino acids hydrophobicity,
- 25: last 50 amino acids flexibility,
- 26: last 50 amino acids hydrophilicity,
- 27: last 50 amino acids Emini surface accessibility,
- 28: last 50 amino acids Janin surface accessibility,
- $29 - 48$: global amino acid composition,
- $49 - 68$: first 50 amino acid composition,
- $69 - 88$: last 50 amino acid composition.

2 Description of the Code

Below we briefly outline the pipeline and the code base which is written in Python 3.6.8,