Bioinformatics, 2021, 18–3
doi: 10.1093/bioinformatics/xxxxx
18 March 2021

Predicting Subcellular Localization of Proteins

Abstract

Motivation: The prediction of protein localization is important to understand both protein function and characteristics. Traditional methods in the laboratory can be time consuming, however with machine learning methods, protein localization can be predicted rapidly, and this should be explored as another method to elucidate protein localization.

Results: A dataset of 11,224 sequences of proteins were used, with both prokaryotic proteins and eukaryotic proteins with their localization. Features were derived from the sequences, including molecular weight, sequence length, amino acid composition, and the presence of specific targeting sequences. Four machine learning algorithms were considered, Random Forest, a Multi-Layer Perceptron, k-Nearest Neighbors and Support Vector Machines. Each of these algorithms were trained and validated using 5-fold cross validation on 90% of the dataset and the highest performing algorithm tested on 10% of the dataset. Support Vector Machines had the highest accuracy on the training dataset and achieved an average F1 Score, Precision and Recall of 0.67, and Matthew's Correlation Co-efficient of 0.57 on the test dataset. The results indicate that this model is a reasonable starting point for protein localization prediction. Future improvements could include the addition of other features and use of different machine learning approaches, such as transformer-based language models.

1 Introduction

Predicting protein localization in a cell is important to understand a protein's function and characteristics, which vary across cellular compartments (Liao, Pan et al. 2021). To date, studying protein localization has mostly relied on more traditional laboratory methods, such as using immunofluorescence techniques to fuse fluorescent proteins to the investigated protein, with subsequent visualization under a microscope (Liao, Pan et al. 2021). These techniques however can be laborious and time consuming (Gardy and Brinkman 2006), compared to computational approaches using machine learning (ML) methods, which have grown significantly in popularity in recent years in the field. Once a ML model has been developed with sufficient accuracy, the model can be used to predict protein localization for novel amino acid sequences at a significantly faster rate than analyzing sequences individually in the laboratory.

Various models have been developed for this purpose, for instance, Chen, Li et al. (2021) used Support Vector Machines (SVM), Random Forest, k-Nearest Neighbors (kNN), and Decision Trees to identify subcellular location of human proteins, achieving accuracies of 0.88, 0.85, 0.83 and 0.71 respectively. These models were built using both network features, such as protein-protein interactions, and functional features, such as Gene Ontology (GO) annotations (Chen, Li et al. 2021).

Gaussian Processes have also been used to build models, using amino acid sequences to make predictions (He, Gu et al. 2012). This model additionally considered correlations between multiple locations in the cell, which improved accuracy of multi-localized protein prediction (He, Gu et al. 2012).

As a result of advancements in proteomic deep learning techniques, it is now possible to treat sequences of amino acids similarly to how a sequence of words are treated in Natural Language Processing. Notably, Almagro Armenteros, Sønderby et al. (2017) used a recurrent neural network with an attention mechanism to predict subcellular localization based on sequences. An accuracy of 78% for 10 categories of proteins and 92% for membrane-bound or soluble proteins was achieved (Almagro Armenteros, Sønderby et al. 2017). Subsequently, Liao, Pan et al. (2021) used a bidirectional long short-term memory network to process the amino acid sequence, followed by a convolutional neural network to extract features from the sequence, achieving an average precision of 0.79.

In this study, protein subcellular localization will be predicted using features derived from the amino acid sequence. Rather than the approach taken by other researchers, which used additional information to predict the localization (Chen, Li et al. 2021), localization will be inferred based on the amino acid sequence alone. Features will be created with information on molecular weight, sequence length, amino acid composition, and presence of targeting sequences. Subsequently, models will be built using a variety of algorithms, including Random Forest, kNN and SVM, all of which have previously achieved high accuracies in predicting protein localization, as well as a neural network, the multi-layer perceptron (MLP). Two high-level approaches will be evaluated on training data using cross validation. Firstly, we use a 5-class model to classify proteins into one of five classes: cytosolic, mitochondrial, nucleic, secreted/extracellular, or prokaryotic proteins. Alternatively, we evaluate using a binary model to classify eukaryotic and prokaryotic proteins, followed by a 4-class model to classify proteins as one of cytosolic, mitochondrial, nucleic, or secreted/extracellular. The highest performing model will then be evaluated against test data.

2 Materials and Methods

2.1 Machine Learning Algorithms

2.1.1 Random Forest

Random Forest is an ensemble algorithm with multiple decision trees. Decision trees partition feature space into subsets, fitting a simple model in each subset (Hastie, Tibshirani et al. 2009). The feature space is partitioned recursively and in a binary manner, first splitting the feature space

into two subsets, then splitting each of those two subsets into two more, and so on, until a stopping criterion is reached. The stopping criteria can include a maximum depth (number of subsets) or until each node (subset) contains only one class. Fig. 1 illustrates this, the tree built splits this dataset into 5 leaves (L1 to L5) based on four splits (S1 to S4). If the datapoint is greater than S1, it will split down the right path. Then, if the datapoint is less than or equal to S3, it will land in leaf 3. The classification is the most popular class in leaf 3. The splits are decided based on the split that minimizes the Gini Impurity or Cross Entropy, as defined in Equations 2 and 3 (Hastie, Tibshirani et al. 2009), where m represents a node with leaf L_m and N_m observations. P_{mk} (Equation 1) represents the proportion of class k observations in node m.

$$p_{mk} = \frac{1}{N_m}\sum_{x_i \in L_m} I(y_i = k) \quad (1)$$

$$\text{Gini Impurity: } \sum_{k=1}^{K} p_{mk}(1 - p_{mk}) \quad (2)$$

$$\text{Cross Entropy: } -\sum_{k=1}^{K} p_{mk}\, log\, p_{mk} \quad (3)$$

Decision trees can be fast and efficient, as the datapoint can be classified using a series of binary decisions until arriving at a leaf node (Prince 2012). However, they have high variance, as different splits early on result in different trees (Hastie, Tibshirani et al. 2009). The Random Forest model extends the concept of decision trees while reducing noise, using bagging. In Random Forest, multiple trees are built, each on a subset of datapoints and features. Usually, building trees on even just one or two features gives optimal results (Breiman 2001). Subsequently, the test data is then classified using a majority vote amongst trees (Breiman 2001).

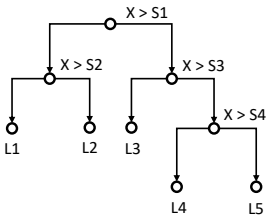


Fig. 1. Decision Tree Splits. Adapted from Hastie, Tibshirani et al. (2009).

2.1.2 Multi-Layer Perceptron (MLP)

The MLP is a supervised learning algorithm that learns a non-linear function (Gardner and Dorling 1998), which can be used for classification. Fig. 2 shows the structure of the MLP, consisting of input, output and one or more non-linear layers of variable sizes, known as hidden layers. The nodes in an MLP are connected by weights and outputs from the previous layer. The input layer is comprised of the features of the dataset. Then, each hidden layer transforms the values from the previous layer using a weighted summation, followed by the application of a non-linear activation function (Bishop 1995, Gardner and Dorling 1998). The output layer transforms the values from the previous hidden layer into the output. MLPs are fully connected, with each node connected to every node in the previous and subsequent layers (Gardner and Dorling 1998). During training, data propagates through the network and if the output does not equal the ground truth, the magnitude of this error will feed back into the network and the weights are accordingly adjusted to minimize the error (Gardner and Dorling 1998).

MLP's learn non-linear models well. However, there are many hyperparameters to tune, including number of layers and the regularization parameter, which reduces overfitting (Scikit Learn 2021). The MLP has a non-convex loss function and may converge to local minima. Additionally, the MLP is sensitive to feature scaling and therefore it is recommended to standardize the data (Scikit Learn 2021).

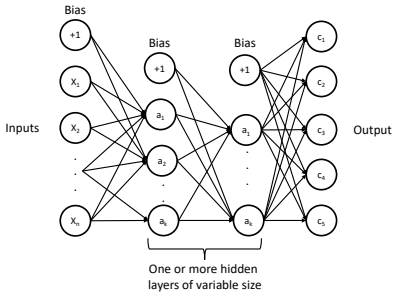


Fig. 2. A two-layer MLP. Adapted from Scikit Learn (2021).

2.1.3 k Nearest Neighbors (kNN)

The kNN classifier does not require learning, it is instead memory based. A test point is classified by finding the k nearest training points, or neighbors, and classifying the test point based on a majority vote between the neighbors (Hastie, Tibshirani et al. 2009). A distance metric is used to calculate the distance between the test point and all the training points to find the k nearest points. A commonly used and simple metric is the squared Euclidian distance (Equation 4) (Barber 2012), where the distance between the i^{th} training point, x_i, and the test point, x_0, is calculated. However, the Euclidian distance does not take data distribution into account and if features have different scales, the feature with the largest scale is considered the most in distance calculations (Barber 2012). Therefore, it is important to standardize the data before running kNN.

$$\text{Squared Euclidian Distance: } (x_i - x_0)^2 \quad (4)$$

kNN has been successful in a variety of classification tasks, including handwritten digit classification and classifying EKG patterns (Hastie, Tibshirani et al. 2009). kNN is best for smaller datasets as the entire dataset must be used to classify and if the dataset is very high dimensional, the distance computation will be expensive (Barber 2012).

2.1.4 Support Vector Machines (SVM)

SVMs are classifiers that find a boundary, known as a hyperplane, between two classes in a dataset (Hastie, Tibshirani et al. 2009). If a point is close to the decision boundary, a small change will lead to a potential misclassification. Therefore, for robustness, the decision boundary should be separated from the data by a margin (Barber 2012). The best hyperplane will maximize this margin (Bishop and Nasrabadi 2006). In the case of overlapping classes in feature space, the goal is still to maximize the margin, but allow for some misclassifications. To do this, the hyperparameter C is used. This is the tradeoff between penalizing misclassifications and maximizing the margin (Bishop and Nasrabadi 2006). The closest data points to the hyperplane are called the support vectors.

In the simplest case, a linear decision boundary is found between the classes. This can also be extended to find more complex and non-linear decision boundaries. To find more flexible boundaries, additional dimensions are added, and the feature space is allowed to get very large. The idea is that in higher dimensions, the data could become separable. A kernel function is used to do this, this projects the data to a higher dimension and with the right kernel, the data could become separable (Noble 2006). When the resulting boundary is projected back into the lower, 2-dimensional space, a non-linear boundary is seen (Noble 2006). As distances are computed in SVM, it is recommended to standardize features before running the model, for the same reasoning as in kNN (Section 2.1.3). An example kernel is the radial basis function kernel (Equation 5) (Hastie, Tibshirani et al. 2009), where $\boldsymbol{\gamma}$ is a hyperparameter.

$$K(x, x') = \exp\left(-\gamma \|x - x'\|^2\right) \quad (5)$$

The binary classifier can be extended to classify multiple classes using a one-verses-one or one-verses-rest approach (Bishop and Nasrabadi 2006). This splits the multi-class problem into multiple binary classifications. The one-verses-one approach forms multiple binary classification problems for one class, establishing a classification for that class against each other class. This can be quite slow as many classifiers need to be trained (Bishop and Nasrabadi 2006). The one-verses-rest approach forms one binary classification problem per class, one class in the dataset against all other classes. This is faster than the one-verses-one approach but can suffer from issues of class imbalance as the size of the "rest" of the data is likely much larger than one class (Bishop and Nasrabadi 2006).

2.2 Dataset

11,224 sequences of amino acids were provided. Within this dataset, there were 3,004 cytosolic sequences, 1,299 mitochondrial sequences, 3,314 nucleic sequences, 1,605 secreted or extracellular sequences and 2,002 prokaryotic sequences. The prokaryotic sequences were added as they can contaminate sequencing samples. The proteins have variable length, with 98% of the proteins less than 2000 amino acids in length and a mean length of 526 amino acids. A histogram of sequence length distribution is seen in Fig. 3. In addition to this, a challenge dataset was provided with sequences of 20 proteins to make predictions for.

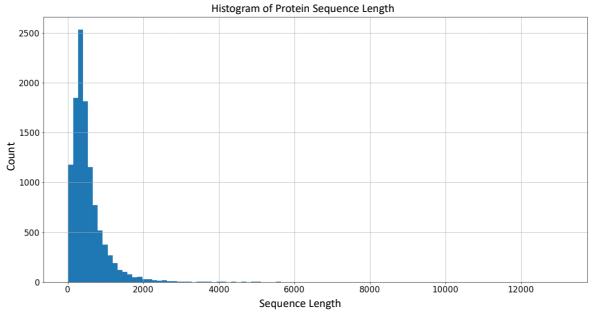


Fig. 3. Histogram to show the distribution of protein sequence length.

3 Experiments and Results

3.1 Data Analysis and Processing

To process this data, the sequences were screened for codes that code for more than one amino acid, such as B for aspartate/asparagine, Z for glutamate/glutamine or X for any amino acid. In addition, codes that do not code for an amino acid were screened for, such as a "*" for translation stop or a "-" for a gap (U.S. National Library of Medicine). There were 55 sequences with a X, accounting for 0.5% of the dataset. Replacing the X's would be challenging, as they could mean any amino acid with equal probability, so sequences containing an X were removed. One sequence contained a B, this was replaced by a D for aspartate and has a 50% probability of being correct. No sequences were found to contain a Z, a translation stop signal or a gap.

3.2 Training and Test Data Split

It was assumed that the sequences in this dataset were non-homologous, so the dataset can be shuffled and split into training and test datasets randomly. 90% of the data was used for training and validation and the rest of the data was used for testing and evaluating the chosen model. The percentages of each class were checked in the training and test datasets to ensure that the proportions of each class were the same.

3.3 Feature Engineering

Feature engineering was then carried out to derive features from the sequences in the dataset. This was first performed on the training dataset during model training, and a similar approach was then conducted on the test and challenge datasets before testing. The following features were derived from the sequences, calculated using the BioPython library (Biopython 2021):

(1) Sequence length.

(2) Amino acid composition across the entire sequence. There are known associations between amino acid composition and localization. For example, proteins with many charged amino acids, glutamine or asparagine residues are associated with the nucleus and proteins with many proline residues are associated with the cytosol (Cascarina and Ross 2018).

(3) Amino acid composition for 50 amino acids at the N-terminus, where many targeting sequences are predominantly located.

(4) Isoelectric point.

(5) Molecular weight.

(6) Aromaticity value. This is the relative frequency of Phenylalanine, Tryptophan and Tyrosine and was calculated using the method described by Lobry and Gautier (1994).

In addition to the above, specific targeting sequences were searched for and a binary feature was created to indicate their presence. For nucleic proteins the following sequences were searched for at any point in the sequence (Lu, Wu et al. 2021):

(1) Lysine (K), Lysine or Arginine (R), any amino acid (X), Lysine or Arginine, then a Lysine. In summary: K (K/R) X (K/R) K.

(2) Proline (P) followed by four amino acids containing three basic residues.

(3) Arginine or Lysine, 10 to 12 amino acids, Lysine, Arginine, any amino acid and then Lysine. In summary: (R/K) $X_{10\text{-}12}$ K R X K.

For the secretory pathway, proteins generally contain a sequence of 5-15 hydrophobic amino acids near the N-terminus (Kapp, Schrempf et al. 2009). Therefore, a binary feature was created to search for 5 hydrophobic amino acids (Betts and Russell 2003) near the N-terminus. Mitochondrial proteins have been found to contain a sequence of 15-70 alternating hydrophobic and positively charged (Betts and Russell 2003) amino acids at the N-terminus to form an amphipathic helix (Bacman, Gammage et al. 2020). Therefore, a pattern of 15 alternating hydrophobic and positively charged amino acids at the N-terminus was searched for.

After the binary features indicating targeting sequences were added, each of these was ensured to contain a match to at least one sequence. The mitochondrial targeting sequence was not found in any sequence and therefore that feature was removed from the training dataset.

Once features were derived for every sequence, the dataset was edited to remove the sequence and only the derived features were used to predict protein location. After features were added, there were 48 features in total. Additional feature engineering was conducted to standardize any non-categorical features (everything apart from the binary targeting features). This was done to ensure all features were on the same scale.

Furthermore, the dataset is slightly imbalanced, with approximately 26% of the dataset containing cytosolic proteins, 12% mitochondrial proteins, 30% nucleic proteins, 14% secreted proteins and 18% prokaryotic

proteins. Balancing the dataset through resampling strategies, such as the Synthetic Minority Oversampling Technique (SMOTE) (Imbalanced Learn 2022), was considered, however it was decided not to resample the training data due to the possibility of overfitting.

3.4 Feature Selection

Feature selection was conducted to remove redundant features that do not improve model performance, thus increasing computational efficiency. The best 30, 35, 40 and 45 features were selected. The best features were then tested using a range of algorithms, including the MLP and Random Forest. An accuracy reduction of 1-3% was noted with all feature subsets. This is not a very large reduction and if the computational time to run the model with all features was high, it would have been considered to use a subset. However, as all features did contribute to model accuracy and the models ran with all features in a relatively short time period, it was decided to retain all of the features.

3.5 Model Approaches

Two approaches were considered to build this model. Firstly, the problem could be treated as a multi-class classification problem, where one of five classes must be predicted (Prokaryotic, Cytosolic, Nucleic, Mitochondrial or Secreted). Alternatively, the problem could be treated as a combination of two approaches, a binary classification problem to classify proteins as prokaryotic or eukaryotic, followed by a multi-class classification of eukaryotic proteins as one of the four remaining classes (Cytosolic, Nucleic, Mitochondrial or Secreted). Both approaches were considered and are discussed in Sections 3.6 and 3.7 respectively.

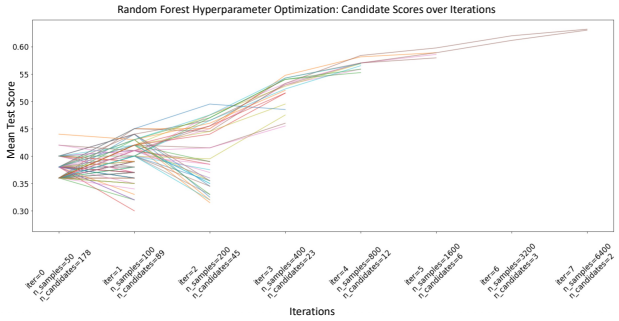
3.6 Multi-Class Model

To train a multi-class model, four algorithms were considered, Random Forest, MLP, kNN and SVM. For each of these algorithms hyperparameter optimization was firstly carried out, followed by model training and subsequent validation on the optimized hyperparameters.

To tune the hyperparameters of the model, a halving random search was carried out (Scikit Learn 2021). This is an iterative approach; in the first iteration a small amount of data is used on all hyperparameter candidates. Over successive iterations, the number of candidates is halved, and the amount of data used doubled. This is repeated until only two candidates remain and the best one is selected. This has shown to be much faster than an exhaustive grid search and therefore was employed in this study. The result of a halving random search is demonstrated in Fig. 4, using Random Forest as an example.

Following hyperparameter optimization, the model was trained and evaluated using 5-fold cross validation. For each fold, the Precision (Equation 6), Recall (Equation 7) and F1 score (Equation 8) were computed for each class and overall, as well as an Area Under Receiver Operating Curve (AUC) score and Matthew's Correlation Co-efficient (MCC) (Equation 9, where TP are the True Positives, TN are the True Negatives, FP are the False Positives and FN are the False Negatives). The results are shown for all models in Table 1. The weighted averages and micro averages were used to report these results as they take the class weightings into account, which is useful here as the classes are not perfectly balanced. In addition, a confusion matrix, and Receiver Operating Characteristic (ROC) curve were plotted (Fig. 5 and 6). The ROC curve shows both a micro and macro average, where the macro average treats each class the same and the micro average is better for imbalanced classes.

Fig. 4. Random Forest Hyperparameter Optimization. As iterations continue, one candidate is found to have the highest mean test score.



$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (6)$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (7)$$

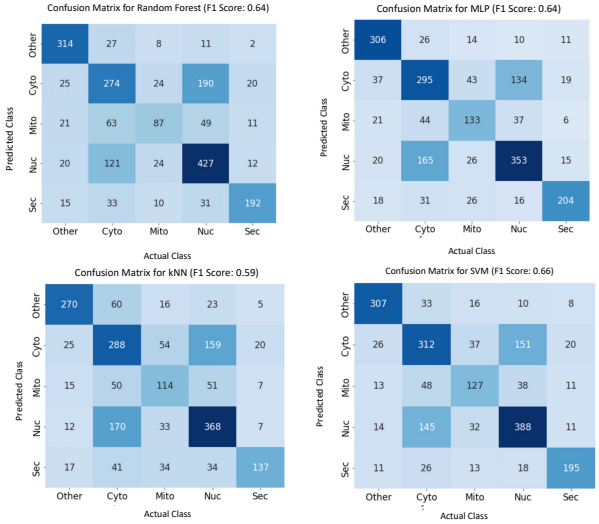
$$F1\ Score = \frac{2(Precision \times Recall)}{Precision + Recall} \quad (8)$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (9)$$

Table 1. Results of the 5-class models. These are the weighted or micro averages averaged over 5 folds. The results in **bold** are the highest values across models. In the table, RF refers to the Random Forest model.

Model	Precision (weighted average)	Recall (weighted average)	F1 Score (weighted average)	AUC (micro-average)	MCC (mean)
RF	0.64	0.64	0.64	0.77	0.54
MLP	0.64	0.64	0.64	0.77	0.53
kNN	0.61	0.59	0.59	0.74	0.47
SVM	**0.66**	**0.66**	**0.66**	**0.79**	**0.56**

Fig. 5. Confusion Matrices for the four models, with the corresponding weighted average F1 scores. The "Other" class corresponds to prokaryotic proteins. "Cyto" refers to the cytosol, "Mito" refers to the mitochondria, "Nuc" refers to the nucleus and "Sec" refers to secreted or extracellular proteins.



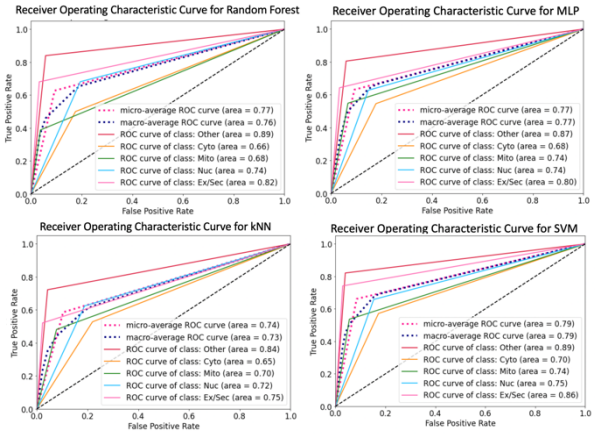


Fig. 6. ROC curves for the four models. The "Other" class corresponds to prokaryotic proteins. "Cyto" refers to the cytosol, "Mito" refers to the mitochondria, "Nuc" refers to the nucleus and "Ex/Sec" refers to secreted or extracellular proteins.

The highest-ranking model was the SVM model, achieving the highest Precision (0.66), Recall (0.66), F1 Score (0.66), AUC (0.79) and MCC (0.56) across all four models (Table 1). When analyzing the confusion matrices (Fig. 5), the SVM model correctly predicted the most cytosolic proteins across the four models. However, the MLP performed best at mitochondrial and secreted/extracellular protein prediction and Random Forest performed best at prokaryotic and nucleic protein prediction. The ROC curves (Fig. 6) showed that the SVM model achieved the highest AUC across all five classes, with scores of 0.89, 0.70, 0.74, 0.75 and 0.86 for the prokaryotic, cytosolic, mitochondrial, nucleic, and secreted classes respectively. Random Forest achieved the same AUC as SVM for the prokaryotic class. The MLP additionally achieved the same AUC as SVM in the mitochondrial class.

3.7 Binary Model followed by Multi-Class Model

For this approach, first a binary model was trained to distinguish prokaryotic proteins from eukaryotic proteins, followed by a multi-class model to classify where in the cell the eukaryotic protein resides. To assess whether this approach is better than the 5-class model trained in Section 3.6, the accuracies of the multi-class models were compared. Supposing that the accuracy of the binary model is high, if the removal of prokaryotic proteins from the dataset does not result in a better multi-class model than the 5-class model or results in similar model performance, the 5-class model would be preferred as it is simpler to train and run inference against.

3.7.1 Prokaryotic Binary Model

As the binary model is of less importance than the multi-class model, an initial model was trained and validated using 5-fold cross validation with the highest performing model from Section 3.6, SVM. This would provide a preliminary result, which could be further refined if the 4-class model performs better than the 5-class model. The results of this model are seen in Table 2 and the corresponding confusion matrix and ROC curve in Fig. 7. The model has performed well, with an overall F1 score of 0.94.

3.7.2 Eukaryotic Multi-Class Model

To train this model, a similar approach to Section 3.6 was used. Four algorithms were considered, Random Forest, MLP, kNN and SVM. Hyperparameter tuning was first carried out using a halving random search, followed by model training and validation using 5-fold cross validation. The results of this are shown in Table 3, as well as the confusion matrices and ROC curves in Fig. 8 and 9.

The results are similar to the 5-class model, where the model using SVM has achieved the best accuracy as quantified by the Precision (0.65), Recall (0.64), F1 Score (0.64), AUC (0.76) and MCC (0.49). Analysis of the confusion matrices in Fig. 8, revealed that SVM is best at classifying cytosolic and mitochondrial proteins. However, the MLP and Random Forest models were better at classifying nucleic and extracellular/secreted proteins. The ROC curves (Fig. 9) showed that SVM had the highest accuracy at classifying cytosolic, mitochondrial, nucleic, and secreted/extracellular proteins, with AUC scores of 0.67, 0.77, 0.72 and 0.86 respectively. The MLP was equally good at classifying cytosolic and nucleic proteins. Overall, the SVM model was decided to be the highest performing model based on the overall metrics in Table 3.

Table 2. Results of the binary model. These are the weighted or micro averages averaged over the 5 folds.

Model	Precision (weighted average)	Recall (weighted average)	F1 Score (weighted average)	AUC (micro-average)	MCC (mean)
SVM	0.94	0.94	0.94	0.87	0.79

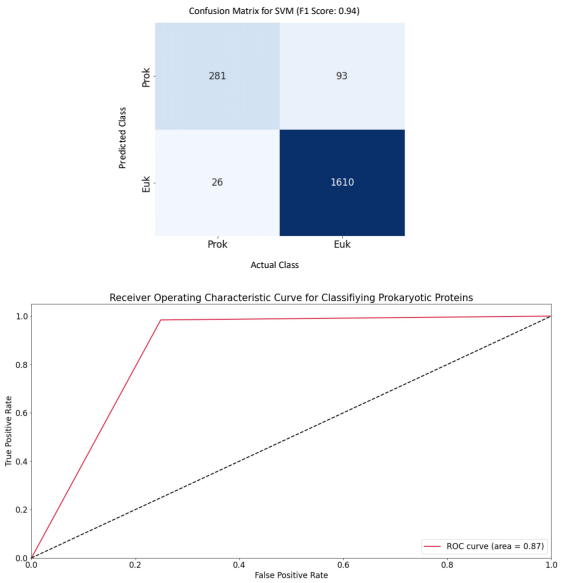


Fig. 7. Confusion Matrix and ROC curve for the SVM binary classification model built to classify sequences as prokaryotic ("Prok") or eukaryotic ("Euk").

Table 3. Results of the 4-class models. These are the weighted or micro averages averaged over 5 folds. The results in **bold** are the highest values across models. RF refers to the Random Forest model.

Model	Precision (weighted average)	Recall (weighted average)	F1 Score (weighted average)	AUC (micro-average)	MCC (mean)
RF	0.63	0.62	0.62	0.74	0.46
MLP	0.63	0.63	0.63	0.74	0.48
kNN	0.61	0.58	0.58	0.72	0.40
SVM	**0.65**	**0.64**	**0.64**	**0.76**	**0.49**

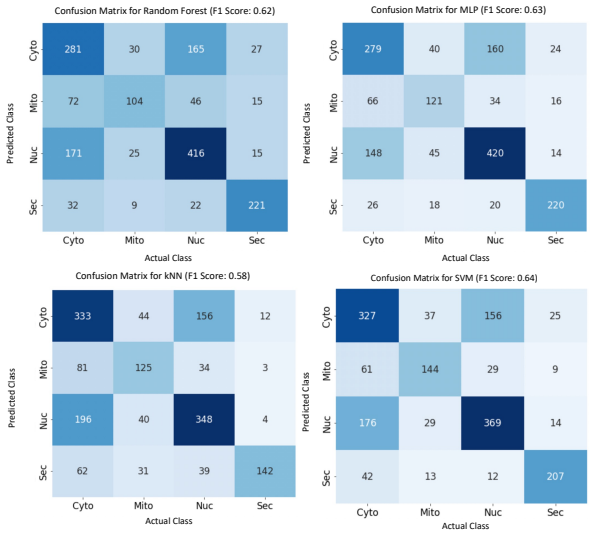


Fig. 8. Confusion Matrices for the four models to classify eukaryotic proteins, with the corresponding weighted average F1 scores. "Cyto" refers to the cytosol, "Mito" refers to the mitochondria, "Nuc" refers to the nucleus and "Sec" refers to secreted or extracellular proteins.

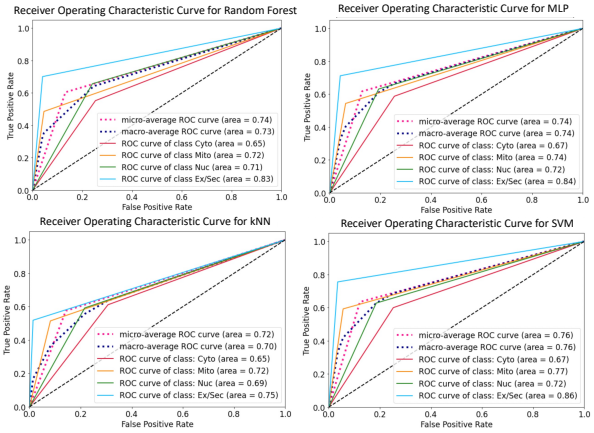


Fig. 9. ROC curves for four models to classify eukaryotic proteins. "Cyto" refers to the cytosol, "Mito" refers to the mitochondria, "Nuc" refers to the nucleus and "Ex/Sec" refers to secreted or extracellular proteins.

3.8 Model Selection

The best performing model was then selected. The two high-level approaches were compared, the 5-class classification model (Section 3.6) against the binary model to classify prokaryotic and eukaryotic proteins followed by a 4-class classification model (Section 3.7).

As the focus is on eukaryotic protein location classification, the classification accuracies of both the 4-class and 5-class models were assessed. The two top performing models were compared, which both used SVM to classify proteins. The 5-class model using SVM has higher Precision, Recall, F1 Score, AUC and MCC metrics (Table 1). The differences between the Precision, Recall and F1 Scores of the 5-class (Table 1) and 4-class (Table 3) models were relatively minor, with a difference of 0.01 to 0.02. However, the AUC was 0.03 higher and the MCC 0.07 higher in the 5-class model. In addition, the ROC curves (Fig. 6 and 9) were compared. The 5-class model performs better when classifying cytosolic and nucleic proteins and the 4-class model performs better in classifying mitochondrial proteins. Both models perform equally well on extracellular/secreted proteins. Overall, since the performance of the 5-class SVM model is slightly better and it is a simpler model to train and run inference against, this model was chosen as the best performing model. The results for this model on the test dataset are detailed in Section 3.9.

3.9 Results for Test Dataset

The model was then evaluated against the test dataset. Before evaluating, feature engineering was carried out. Additional features were derived, and the numerical features were standardized, as outlined in Section 3.3. The results for evaluating the SVM model against the test dataset are in Table 4. The table contains the metrics for each class, as well as averages of the Precision, Recall, F1 Score, AUC and MCC metrics for the overall result. The confusion matrix and ROC curve is shown in Fig. 10 and 11.

In Table 4, the overall metrics are very similar to the results of training and validation, implying that the model was not overfitted to the training data. The overall metrics are not very high for this model, with an average Precision, Recall and F1 Score of 0.67, AUC of 0.79 and MCC of 0.57. However, other studies have achieved higher results (Almagro Armenteros, Sønderby et al. 2017, Chen, Li et al. 2021, Liao, Pan et al. 2021), indicating that this model is a good starting point, but there is potential room for improvement through additional features and use of different algorithms.

The metrics for each of the five classes (Table 4, Fig. 10 and 11) show that the model performs best in classifying prokaryotic (F1 Score: 0.82) and extracellular/secreted proteins (F1 Score: 0.73), performs moderately well in classifying nucleic proteins (F1 Score: 0.68) and performs worst on cytosolic (F1 Score: 0.58) and mitochondrial proteins (F1 Score: 0.58).

Table 4. Results of evaluating the 5-class SVM model on the test dataset. The first 5 rows contain the results for each class and the last row contains the overall averaged evaluation results.

Class	Precision	Recall	F1 Score	AUC	
Prok	0.84	0.80	0.82	0.89	
Cyto	0.56	0.61	0.58	0.71	
Mito	0.57	0.58	0.58	0.77	
Nuc	0.68	0.67	0.68	0.77	
Ex/Sec	0.78	0.68	0.73	0.83	
Model	Precision (weighted average)	Recall (weighted average)	F1 Score (weighted average)	AUC (micro-average)	MCC
Overall	0.68	0.67	0.67	0.79	0.57

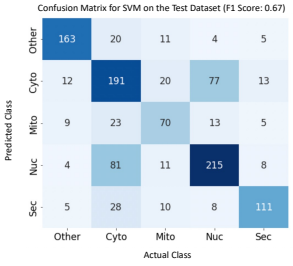


Fig. 10. Confusion Matrix for the SVM model applied to the test dataset, with the corresponding weighted average F1 score. The "Other" class corresponds to prokaryotic proteins. "Cyto" refers to the cytosol, "Mito" refers to the mitochondria, "Nuc" refers to the nucleus and "Sec" refers to secreted or extracellular proteins.

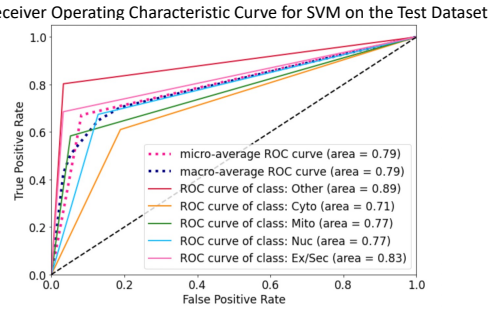


Fig. 11. ROC curve for the SVM model applied to the test dataset. The "Other" class corresponds to prokaryotic proteins. "Cyto" refers to the cytosol, "Mito" refers to the mitochondria, "Nuc" refers to the nucleus and "Ex/Sec" refers to secreted or extracellular proteins.

4 Discussion

As prokaryotic classification accuracy is quite high (Table 4) and is not the main goal, we will focus on how the results for the eukaryotic proteins could be improved. The accuracy obtained for the extracellular/secreted proteins was the highest out of the four classes of eukaryotic proteins, with an F1 Score of 0.73 (Table 4). This indicates that this class of proteins likely has some clear distinguishing characteristics that the model was able to learn, for example the targeting sequence at the N-terminus.

However, the other classes were not predicted as accurately. For nucleic proteins, the F1 Score obtained was 0.68 (Table 4). 30% of the dataset contained nucleic proteins, therefore it is unlikely that data imbalance and model bias towards the more frequent classes contributed to a lower accuracy. Some nucleic targeting sequences were added as binary features, although additional features could be added. For example, Chen, Li et al. (2021) found that some functional features, such as GOs, could assist with classification of nucleic proteins. This included the RNA surveillance (GO:0071025) annotation, which was found to be positively correlated with nucleic proteins (Chen, Li et al. 2021) as RNA surveillance occurs in the nucleus (Hernandez-Verdun, Roussel et al. 2010). Additionally, GO:0045596 accounts for negative regulation for cell differentiation and was also positively correlated with nucleic proteins (Chen, Li et al. 2021).

The F1 score for mitochondrial proteins was 0.58 (Table 4), below the weighted average for the dataset. 12% of the dataset was mitochondrial, therefore additional sequences should be added to reduce model bias towards the more frequent classes. A mitochondrial targeting sequence was searched for, however returned no matches. Therefore, other features could be incorporated in future work. This may include annotations for the GO term for the membrane envelope structures possessed by mitochondria (GO:0031975), and indeed studies have found this to be positively correlated with the mitochondrion (Peabody, Laird et al. 2016, Chen, Li et al. 2021).

Finally, cytosolic proteins also had a low classification accuracy, with an F1 score of 0.58 (Table 4). 26% of the dataset were cytosolic proteins, such that our findings are unlikely to be attributed to issues relating to class imbalance. However, no additional features were added to relate to cytosolic proteins, such as targeting sequences. Therefore, to improve the model additional features could be incorporated in future work. For instance, the cytosol is involved in wound healing (Jeon and Jeon 1975) and therefore the GO annotation for wound healing (GO:0042060) could act as a marker for cytosolic proteins (Chen, Li et al. 2021).

In addition, future models should also consider that proteins may have multiple subcellular localizations. For instance, Liu, Jin et al. (2022) proposed a novel method, ML-locMLFE, which can predict multiple labels for a protein, achieving an overall accuracy of 72.73% on a SARS-CoV-2 dataset.

Finally, the model could be enhanced further through the application of other ML techniques. Neural networks have shown promise in this task, for instance using recurrent neural networks with an attention mechanism (Almagro Armenteros, Sønderby et al. 2017). As the amino acid sequence could be treated like a sequence of words, it would be interesting to apply other neural networks that have been successful in natural language modelling. BERT (Bidirectional Encoder Representations from Transformers) (Devlin, Chang et al. 2018), a transformer-based architecture, has previously achieved state of the art results on many natural language tasks and can be extended for use with proteins. ProtBERT, a model trained on protein sequences (Elnaggar, Heinzinger et al. 2020) has been developed and it would therefore be interesting to apply these to protein localization classification.

5 Conclusion

In this study it has been shown that protein subcellular location can be predicted using an SVM with features derived directly from the amino acid sequence. The model classifies extracellular/secreted proteins well, classifies nucleic proteins moderately well, but has a low classification accuracy on mitochondrial or cytosolic proteins. This model acts as a reasonable starting point for protein localization classification however, there is scope for further improvement. Future work could focus on the addition of features, building multi-label classification models or utilizing alternative models that have achieved state of the art results on natural language processing tasks.

References

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

Bacman, S. R., et al. (2020). Manipulation of mitochondrial genes and mtDNA heteroplasmy. Methods in cell biology, Elsevier. **155:** 441-487.

Barber, D. (2012). Bayesian reasoning and machine learning, Cambridge University Press.

Betts, M. and R. Russell (2003). Amino acid properties and consequences of substitutions In Barnes MR, & Gray IC (Eds.), Bioinformatics for geneticists (pp. 289–316). Hoboken, NJ: John Wiley & Sons.

Biopython (2021). "Analyzing protein sequences with the ProtParam module.". Retrieved 23/02/2022, from https://biopython.org/wiki/ProtParam.

Bishop, C. M. (1995). Neural networks for pattern recognition, Oxford university press.

Bishop, C. M. and N. M. Nasrabadi (2006). Pattern recognition and machine learning, Springer.

Breiman, L. (2001). "Random forests." Machine learning **45**(1): 5-32.

Cascarina, S. M. and E. D. Ross (2018). "Proteome-scale relationships between local amino acid composition and protein fates and functions." PLoS computational biology **14**(9): e1006256.

Chen, L., et al. (2021). "Predicting Human Protein Subcellular Locations by Using a Combination of Network and Function Features." Frontiers in Genetics: 2229.

Devlin, J., et al. (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Elnaggar, A., et al. (2020). "ProtTrans: towards cracking the language of Life's code through self-supervised deep learning and high performance computing." arXiv preprint arXiv:2007.06225.

Gardner, M. W. and S. Dorling (1998). "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences." Atmospheric environment **32**(14-15): 2627-2636.

Gardy, J. L. and F. S. Brinkman (2006). "Methods for predicting bacterial protein subcellular localization." Nature Reviews Microbiology **4**(10): 741-751.

Hastie, T., et al. (2009). The elements of statistical learning: data mining, inference, and prediction, Springer.

He, J., et al. (2012). "Imbalanced multi-modal multi-label learning for subcellular localization prediction of human proteins with both single and multiple sites." PloS one **7**(6): e37155.

Hernandez-Verdun, D., et al. (2010). "The nucleolus: structure/function relationship in RNA metabolism." Wiley Interdisciplinary Reviews: RNA **1**(3): 415-431.

Imbalanced Learn (2022). "Over-sampling methods." Retrieved 23/02/2022, from https://imbalanced-learn.org/stable/references/over_sampling.html.

Jeon, K. and M. Jeon (1975). "Cytoplasmic filaments and cellular wound healing in Amoeba proteus." The Journal of cell biology **67**(1): 243-249.

Kapp, K., et al. (2009). "Post-targeting functions of signal peptides." Protein transport into the endoplasmic reticulum: 1-16.

Liao, Z., et al. (2021). "Predicting subcellular location of protein with evolution information and sequence-based deep learning." BMC bioinformatics **22**(10): 1-23.

Liu, Y., et al. (2022). "Predicting the multi-label protein subcellular localization through multi-information fusion and MLSI dimensionality reduction based on MLFE classifier." Bioinformatics **38**(5): 1223-1230.

Lobry, J. and C. Gautier (1994). "Hydrophobicity, expressivity and aromaticity are the major trends of amino-acid usage in 999 Escherichia coli chromosome-encoded genes." Nucleic acids research **22**(15): 3174-3180.

Lu, J., et al. (2021). "Types of nuclear localization signals and mechanisms of protein import into the nucleus." Cell communication and signaling **19**(1): 1-10.

Noble, W. S. (2006). "What is a support vector machine?" Nature biotechnology **24**(12): 1565-1567.

Peabody, M. A., et al. (2016). "PSORTdb: expanding the bacteria and archaea protein subcellular localization database to better reflect diversity in cell envelope structures." Nucleic acids research **44**(D1): D663-D668.

Prince, S. J. (2012). Computer vision: models, learning, and inference, Cambridge University Press.

Scikit Learn (2021). "Neural network models (supervised)." Retrieved 23/02/2022, from https://scikit-learn.org/stable/modules/neural_networks_supervised.html.

Scikit Learn (2021). "sklearn.model_selection.HalvingRandomSearchCV." Retrieved 23/02/2022, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.HalvingRandomSearchCV.html.

U.S. National Library of Medicine. "BLAST: Query Input and database selection." Retrieved 29/02/2022, from https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=BlastHelp.

Appendix

A.1 Code Overview

This section contains an overview of the code that was written to build this model.

A.1.1 Required Libraries

The following libraries are recommended to create the code for the model:

1. NumPy
2. Pandas
3. Matplotlib
4. Seaborn
5. SciPy
6. BioPython
7. Imbalanced Learn
8. Scikit Learn

A.1.2 Data Loading and Exploration

1. Load the data into a Pandas DataFrame with the sequence, sequence length, and the target class. The SeqIO interface in BioPython may be helpful to do this.
2. Remove the 55 sequences with an X.
3. For the single sequence containing a B, replace this with a D.

A.1.3 Training and Test Data Split

1. Create an X and y dataset, where X contains all the features and y has the relevant target class.
2. Split the X and y datasets into training and test datasets, with 10% of the data being used for testing. The train test split function from Scikit Learn can be used for this.

A.1.4 Feature Engineering

1. Using the ProParam module from BioPython, for each sequence calculate the molecular weight, aromaticity, isoelectric point, and percentage amino acids for both the whole sequence

and the 50 amino acids at the N terminus. Add these to the X DataFrame or array.

2. Create binary features for each of the targeting sequences being searched for, there are five in total (three nucleic sequences, one mitochondrial sequence and one secretion/extracellular sequence). Check that each of these have at least one sequence match, the mitochondrial sequence was found to have no matches in this dataset so was not used. Add the four binary features (three nucleic sequences and one secretion/extracellular sequence) to X.
3. Remove the sequence feature itself.
4. In total, there should be 48 columns. 20 for the amino acid percentage over the whole sequence, another 20 for amino acid percentage over the 50 amino acids at the N terminus, 4 binary features for the targeting sequences, molecular weight, aromaticity, isoelectric point, and sequence length.
5. Standardize the non-categorical features of X. The Standard Scaler from Scikit Learn can be used here.

A.1.5 Feature Selection

1. Feature selection was carried out using the Select k-best method from Scikit Learn. This was used to select the best *k* features.
2. The best *k* features can then be used to build a model and the accuracy computed. The process used to build a model is outlined in Appendix A.1.6. In this case, all features were found to contribute to the model, so all were used.

A.1.6 Multi-Class Model

The following process was followed for each of the four models trained, namely Random Forest, MLP, kNN and SVM. The Scikit Learn implementations of these algorithms were used.

1. First conduct hyperparameter optimization using the halving random search method from Scikit Learn.
2. Train and validate the classifier using 5-fold cross validation with the best hyperparameters found. Cross validation can be conducted using the k-Fold method from Scikit Learn. For each fold, compute metrics using the classification report method from Scikit Learn (this includes Precision, Recall and F1 Score), the confusion matrix, Area Under the ROC Curve score and Matthew's Correlation Co-efficient.
3. The confusion matrix can be visualized using a Seaborn heatmap and the ROC curve can be drawn for the multi-class case as documented on this page: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html.

After hyperparameter optimization, the following hyperparameters were found to work best for each model:

1. Random Forest:
    a. 'bootstrap': False
    b. 'criterion': 'entropy'
    c. 'max_depth': None
    d. 'max_features': 3
    e. 'min_samples_split': 8
    f. 'n_estimators': 150
2. MLP:
    a. 'solver': 'adam'
    b. 'max_iter': 300
    c. 'learning_rate_init': 0.001
    d. 'learning_rate': 'adaptive'
    e. 'early_stopping': False
    f. 'alpha': 1.05
    g. 'activation': 'tanh'
3. kNN:
    a. 'weights': 'distance'
    b. 'p': 1
    c. 'n_neighbors': 5
    d. 'algorithm': 'ball_tree'
4. SVM:
    a. 'kernel': 'rbf'
    b. 'gamma': 'scale'
    c. 'decision_function_shape': 'ovo'
    d. 'C': 1.5

A.1.7 Binary Model followed by Multi-Class Model

1. For the binary model, the y dataset created was edited to contain a binary label for prokaryotic or eukaryotic proteins.
2. A SVM model was trained and validated using 5-fold cross validation, and the metrics (including Precision, Recall and F1 Score), confusion matrix, AUC score and Matthew's Correlation Co-efficient computed. The confusion matrix and ROC curve can then be visualized. The SVM model used the hyperparameters listed in Appendix A.1.6.
3. For the 4-class model, the X and y dataset were modified to remove the prokaryotic class.
4. Then, the same process as described in section A.2.6 was followed, carrying out hyperparameter optimization followed by model training and validation for each of the four models tested (Random Forest, MLP, kNN and SVM).

After hyperparameter optimization, the following hyperparameters were found for each model:

1. Random Forest:
    a. 'bootstrap': True
    b. 'criterion': 'gini'
    c. 'max_depth': None
    d. 'max_features': 3
    e. 'min_samples_split': 5
    f. 'n_estimators': 100
2. MLP:
    a. 'solver': 'sgd'
    b. 'max_iter': 300
    c. 'learning_rate_init': 0.001
    d. 'learning_rate': 'constant'
    e. 'early_stopping': False
    f. 'alpha': 0.1
    g. 'activation': 'relu'
3. kNN:
    a. 'weights': 'distance'
    b. 'p': 1
    c. 'n_neighbors': 10
    d. 'algorithm': 'brute'
4. SVM:
    a. 'kernel': 'rbf'
    b. 'gamma': 'scale'

c. 'decision_function_shape': 'ovo'
d. 'C': 1.5

A.1.8 Evaluation on Test Data

1. Carry out feature engineering, as outlined in Appendix A.1.4, adding features and standardizing the numerical features.
2. Make predictions on the test dataset using the 5-class SVM model.
3. Evaluate the accuracy using the metrics (including Precision, Recall and F1 Score), confusion matrix, AUC score and Matthew's Correlation Co-efficient. The confusion matrix and ROC curve can then be visualized.