Predicting sub-cellular location of proteins using Machine Learning

Yoga Advaith Veturi

University College London, UK

advaith.veturi.17@ucl.ac.uk,

Abstract -

Predicting the location of a protein within the cell can help in elucidating its function and deducing its involvement in certain biochemical pathways. In this study, machine learning models are investigated to classify amino acid sequences into five classes - cytosolic, mitochondrial, secreted, nuclear and "other". Two models are explored, which are support vector machines and neural networks. These models are trained and tested using a dataset of ≈ 11,000 protein sequences. From the results, it is seen that the test set accuracy is similar for both models (≈ 65-67%), but the support vector machine demonstrates the better performance, as indicated by its macro-F1 score (0.7066). Future steps of research to improve these results includes performing further feature analysis and exploring other sequence-based models such as recurrent neural networks and attention-based models.

1 Introduction

Proteins play a crucial role in various biological processes within the body. They are responsible for catalysing metabolic reactions, transporting molecules from one area of the body to another, mediating cell repair, and also form a part of our immune system. In order to elucidate a protein's function, one key piece of information is the location of the protein within the cell. Sub-cellular locations can be broadly classified into 5 different types:

1. Cytosolic proteins - these lie within the cell cytoplasm, but not in the organelles. They are important for regulating intracellular reactions. Examples include *G3BP1* which is involved in signal transduction pathways and the *MTHFS* enzyme which is required for metabolic reactions in the cytosol [1].

2. Extracellular/Secreted proteins - these are proteins produced from cells and play a role in cell signalling pathways. Examples of these include digestive enzymes such as amylase and peptidase which are secreted from the salivary glands and pancreas to break down proteins and complex sugars, facilitating their absorption into the blood stream.

3. Nuclear proteins - These proteins are found within the cell's nucleoplasm and are crucial for processes such as DNA replication, transcription, gene regulation and epigenomic activating/silencing of genes. Examples include *PDS5A* which keeps the sister chromatids in place during mitosis and *TP53BP1* which is regulates the DNA damage response [1].

4. Mitochondrial proteins - these proteins are situated within the mitochondria and participate in the processes that generate ATP in the cell. Examples include the *CS* protein involved in the citric acid cycle and *LRPPRC* which plays a role in mitochondrial gene transcription [1].

Determining the subcellular location of a protein through experimental methods can be time-consuming. Traditionally, this has been performed using immunofluorescence labelling where the protein of interest is tagged with a fluorescent protein like green-fluorescence protein (GFP) and using a fluorescence microscope, localization of the protein is studied [2]. However, over the past couple of years, due to the advances in high-throughput sequencing technology, the genomic and amino acid sequences have been determined for more than 3000 species, causing an increased population of bioinformatic databases. With this increased abundance of protein data, practioners have resorted to more computational approaches that can predict the protein localization based on patterns found in the data, which is more efficient compared to the experimental methods.

Various studies have been conducted to solve the protein subcellular location prediction task. The approaches used in these studies can be broadly classified into three types: (1) Amino acid composition methods (2) Sequence homology and motif analysis methods and (3) Hybrid methods [3]. Looking at the first method, in Cedano et al 1997 [4], it was found through correlation analysis that the amino acid composition can be used to determine subcellular location, from which they developed ProtLock, a system to classify the protein into based on the amino acid composition vector. CELLO is another composition-based tool which primarily predicts protein locations for gram-negative bacteria [5]. For the second method, examples include Mott et al 2002 [6] which used a homology-based strategy to deter-

mine the probability of proteins being secreted, nuclear or cytosolic proteins based on co-occurrence of SMART domains. Marcotte et al 2000 [7] investigated classification of proteins into mitochondrial or non-mitochondrial locations based on phylogenetic profiles. Some methods use hybrid models and machine learning methods - for example in Chia-Yu Su et al 2007 [8], two models were used to classify proteins from gram-negative bacteria into the cytoplasm, the inner membrane, periplasm, outer membrane and extracellular space - these include PSL101 which is a set of one-vs-one support vector machines and PSLsse which uses protein secondary structure homology alignment. Reinhardt and Hubbard 1998 [9] used deep neural networks to classify sequences from prokaryotic and eukaryotic cells into cytoplasmic, nuclear, mitochondrial and secreted. In fact, the PSORT tool 1999 [10], which is one of the first efforts to computationally determine subcellular location, is also a hybrid model which uses amino acid composition, motifs and N-terminal information (which acts as a "postal-code" for the amino acid sequence).

In this report, sub-cellular location prediction of proteins is performed using machine learning methods. Specifically, two models will be investigated for this task: a Support Vector Machine (SVM) and a Neural Network. While classical machine learning algorithms like SVMs have been greatly explored in the literature for this task, neural networks are relatively less explored. Furthermore, the flexibility neural networks offer in terms of model design is quite large, allowing for further development and research. Subsequent sections will discuss these methods in detail.

2 Methodology

2.1 Approaches

A brief outline on SVMs and Neural Networks is provided subsequently. Specific implementational details are provided in the **Model training**, **Cross-validation** and **Evaluation** sections.

2.1.1 Support Vector Machines (SVMs)

The SVM algorithm is a class of supervised learning algorithms that aims to find the optimal separating hyperplane

$$H_{\mathbf{w},b} = \mathbf{w}^T \mathbf{x} + b$$

that maximises the margin between two classes of data. $\mathbf{w}$ and b are the parameters that define this hyperplane. The margin is defined as the distance between the hyperplane and two closest points on either side of the surface. Specifically in our case, as the data may not be linearly separable, we implement a "soft-margin" SVM, where the objective function is of the form

$$\min \frac{1}{2}\mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{m} \xi_i$$

The first term represents the optimization of the margin itself while the second term represents the total misclassification error. The parameter C is crucial as it controls the trade-off between these two terms. Specifically, it controls how much we wish to reduce misclassifications versus how much we care about computing the optimal decision boundary. Another important component of the SVM is the kernel function, which allows us to map the data to higher-dimensional spaces, making it easier to find the separating hyperplane. In this report, the radial basis kernel function (RBF) is used, which has a parameter γ that controls the variance of the kernel. Lastly, as this is a multi-class classification task, the SVM is implemented using a "one-vs-rest" approach i.e a separate sub-classifier is trained for predicting "Secreted vs Not secreted", "Mitochondral vs Not mitochondral" etc and the final prediction is based on the most confident sub-classifier.

2.1.2 Neural Networks

Neural networks are popular algorithms in machine learning and have been used for various tasks such as image classification, anomaly detection, object detection and natural language processing. These are essentially complex non-linear functions for which the parameters are learned by optimizing a loss function measuring the difference between the true label and the predicted label.

Neural networks are highly flexible when it comes to hyperparameter tuning and searching for the best model. In general, these models can be tuned at two levels: first is the architecture of the network itself, where adding more layers will increase the number of learnable parameters and as a result, the model complexity. The second level is the optimization procedure which involves modifying hyperparameters such as the learning rate, the optimizer itself and the batch-size, in order to speed convergence during training. In order to prevent overfitting of the models, regularization methods including Dropout and L2 regularization can also be modified.

2.2 Dataset

The dataset contains totally 11,224 sequences in `fasta` format. Note that every sequence can be assumed to be non-homologous, and belongs to the set of classes

$$C = \{\text{Cytosolic}, \text{Mitochondrial}, \text{Nuclear}, \text{Other}, \text{Secreted}\}$$

The "Other" class contains prokaryotic sequences that can contaminate samples during the data collection process. These are just treated as a separate class in the problem, to ensure that our models also learn sequences

that do not come under the four main categories of protein sub-cellular location. The proportions of the classes is visualized in Figure 1. To apply our models on completely new data, an additional “blind" dataset is also provided, containing 20 unlabelled sequences.

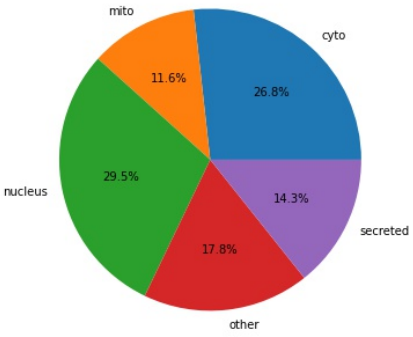

Figure 1: Proportion of the different classes within the whole dataset

2.3 Feature Selection

Feature selection was performed by analysing the sequences using the `ProtParam` Module in the BioPython library. Eight features were short-listed, for which a description is provided in Table 1. Note that the global and local amino acid composition is measured for each of the 20 amino acids, hence there are totally 46 short-listed features.

The correlation between the features was studied by computing the Pearson correlation coefficient r. The result is visualized as a heat map in Figure 4 in Appendix B. From these results, it is noticeable that the sequence length is strongly correlated ($r > 0.95$) with the molecular weight, hence the molecular weight is omitted from the feature set. For the rest of the features, the correlations appear to have moderate-weak correlation ($r < 0.70$), hence they are retained. Totally, there are 45 features which will be used as predictors for our models.

2.4 Data Pre-processing

The dataset was standardized in order to ensure that the features are all on a similar scale. The classes in C were represented using a numerical encoding $C_{\text{num}} = \{0, 1, 2, 3, 4, 5\}$.

During the feature selection method, it was also found that some sequences contained letters `"U"`, `"B"` and `"X"` which could not be interpreted by the `Protparam` module, hence these were ignored, resulting in a final dataset of shape (11160, 45).

2.5 Model Training and Cross Validation

2.5.1 Dataset Splitting

An 80:20 splitting was performed on the datset, where 80% of the data was used for training while the remaining 20% was used as a holdout test set to evaluate model performance on unseen sequences.

2.5.2 SVM model search

For the soft-margin SVM, the hyperparameters of interest were the C value which controls the “hardness" of the margin, and γ, the variance of the RBF kernel. The best values for these parameters were determined by performing a grid-search. The chosen values of C to experiment with were $[1, 10, 100]$ and the chosen range of γ values was $[2^{-7}, 2^{-8}, 2^{-9}, ..., 2^{-15}]$. For each of the hyperparameter combinations, five-fold cross validation was performed; the training data was divided into five groups, where one group was used as a "validation" set while the remaining 4 were used as a training set, and this was repeated for every grouping. Note also that the training set data was shuffled before performing the cross-validation, in order to eliminate bias. The best hyperparameters from the grid search were selected based on those which showed the smallest mean validation set accuracy. The final model was then re-trained with these best hyperparameters.

2.5.3 Neural Network model search

For the neural network model, the hyperparameters of interest were the layer architecture, describing the complexity of the model, and the amount of dropout regularization, which deals with overfitting. From initial experiments, it was found that network architectures having ≥ 1 hidden layers were prone to overfitting, hence only three-layered networks (including input and output layer) were investigated. To counter this effect of overfitting, dropout values of [0.0(No regularization), 0.1, 0.2, 0.3] were tested. This is summarized in Table 2 along with the values of other hyperparameters that were kept fixed. As in the SVM model, a grid search was performed over the Layer configurations and the dropout values. For each combination of hyperparameters, five-fold cross validation was performed. The best hyperparameters were those that demonstrated the smallest mean validation set loss. Using these best hyperparamters, the neural network architecture was again trained.

Feature	Description
Sequence length	Number of amino acids in sequence
Molecular Weight	Sum of the weights of all the amino acids in the sequence.
Isoelectric point	The pH at which the protein is neutral (no electric charge).
Global amino acid composition	Percentage of each of the 20 amino acids in the entire sequence
Local amino acid composition	Percentage of each of the 20 amino acids within a window of amino acids. This is chosen to be the first 50 amino acids.
Aromaticity	Represents the stability of the protein, as measured by the composition of aromatic rings.
Instability Index	Represents the stability of the protein in a test tube.
Gravy	Represents the hydrophobicity of the peptide, calculated by the averaging the hydropathy values of all amino acids.

Table 1: Selected Features from ProtParam library

Hyperparameter	Value
Layers	[45, 5], [45, 16, 5], [45, 32, 5]
Dropout	[0.0, 0.10, 0.20, 0.30]
Activation function (Hidden layer)	Rectified linear unit (ReLU)
Loss	Cross Entropy
Performance metric	Accuracy
Optimizer	Adam
Learning rate	10^{-4}
Batch size	128
Epochs	300

Table 2: Hyperparameters configurations for neural network. The Layers and Dropout are modified while rest are kept fixed.

2.6 Model Evaluation

2.6.1 Prediction on test data

The best SVM and neural network models from the grid search+cross-validation were evaluated by computing the predictions on the held out test dataset (20% of the data from the data splitting). Confusion matrices were created by comparing the true labels with the predicted labels. Using the confusion matrices, the following metrics were calculated:

1. Classification Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
2. Precision = $\frac{TP}{TP+FP}$
3. Recall = $\frac{TP}{TP+FN}$
4. F1-score = $2 * \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision+Recall}}$

TN, FN, TP, FP are the true negatives, false negatives, true positives and false positives respectively. Note that as this is a multi-class classification task, the metrics were calculated for each class separately. The precision for each class is computed by dividing the diagonal elements of the confusion matrix by the sums along their corresponding rows. For the recall, the diagonal elements are divided by the sum along their corresponding columns. The per-class F1-scores are then computed using the per-class precision and recall values. Taking the average of these three metrics over the classes gives the "macro"-precision, "macro"-recall and "macro"-F1 score. The macro-F1 score is especially important as it summarizes the overall model performance. The best model is defined to be the one with the greatest macro-F1 score.

2.6.2 Confidence Estimation on model predictions

When predicting on test examples, it is also helpful to understand the confidence level of the different classifiers for that particular training example. SVMs usually do not provide these class probabilities usually however this could be determined using an extended version of Platt scaling which essentially trains the SVM with a sigmoid function that maps the values to a probability [11]. This functionality is built-in to the Scikitlearn's SVM module.

For neural networks, the confidence level for a class can be described by the linear activation output, which intuitively represents a relative scoring for each class. The activations are converted into a probability distribution over the classes by applying the softmax function

$$\hat{\mathbf{y}} = \sigma(f(X))$$

$$\sigma(\mathbf{z}) = \frac{\exp z_i}{\sum_{j=1}^{K} \exp z_j}$$

where $\mathbf{z}$ is the vector containing activations z_i outputted by the neural network $f(X)$ for the K classes and $\sigma(\cdot)$ is the softmax function. $\hat{\mathbf{y}}$ is the vector of class distribution values. Typically when predicting on new input data, these class probabilities are converted into the final class prediction by taking the argmax of $\hat{\mathbf{y}}$.

2.6.3 Prediction on blind test data

Finally, the best model (chosen using the macro-F1 score) is applied on the dataset of 20 blind test sequences

to generate predictions along with their confidence level (see previous section).

2.7 Code

This study was implemented in an IPython Notebooks. An in-depth code briefing is provided in Appendix A.

3 Results

3.1 Best models from Grid Search + Cross-Validation

The results for the grid search over the various hyperparameter combinations and cross-validation experiments showed that the best SVM model had $C = 10$ and $\gamma = 2^{-7}$ for the RBF kernel. For the neural network implementation, the best performing model had an architecture [45, 32, 5] and a dropout=0.0. Note the remaining hyperparameters are kept the same as listed in Table 2. The validation accuracies for each fold of cross-validation on the SVM is presented in Table 3, along with the mean and standard deviation. For the neural network, the validation loss and accuracy computed on each fold of cross-validation is presented in Table 4, along with the mean and standard deviation. Raw data for all other model variants is presented in Appendix C.

Fold	Val Accuracy
1	0.6618
2	0.6540
3	0.6691
4	0.6711
5	0.6706
Mean ± STD	**0.66532 ±0.00657**

Table 3: Cross Validation results for best SVM model

Fold	Val Loss	Val Accuracy
1	0.8702	0.632
2	0.7998	0.665
3	0.7493	0.681
4	0.7348	0.691
5	0.7675	0.711
Mean ± STD	**0.7843 ±0.04812**	**0.6760 ±0.02658**

Table 4: Cross Validation results for best Neural Network model

3.2 Performance of best SVM and Neural Network

The best models from the grid search and cross validation experiments were retrained on the full training set and performance was measured on the holdout test set. The results for both models are presented in Tables 5. Predictions on a few holdout test sequences are also presented in Appendix D, along with their confidence estimates.

Model	Train Loss	Train Accuracy	Test Loss	Test Accuracy
SVM	-	0.8094	-	0.6783
NN	0.7487	0.694	0.8405	0.6532

Table 5: Model Performance on training and test sets

3.3 Confusion Matrices

The confusion matrices for the best SVM model and the Neural Network model from the grid search+cross-validation are presented in Figures 2 and 3.

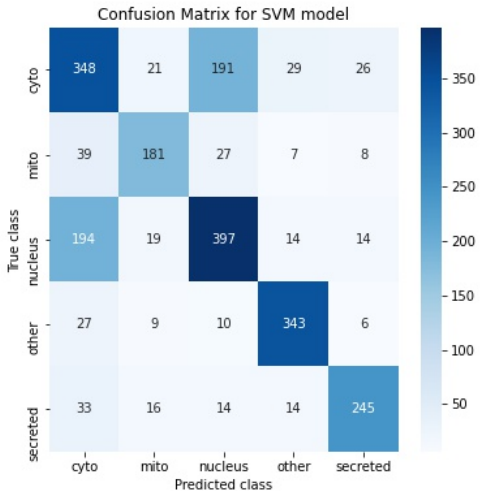

Figure 2: Confusion matrix for SVM model

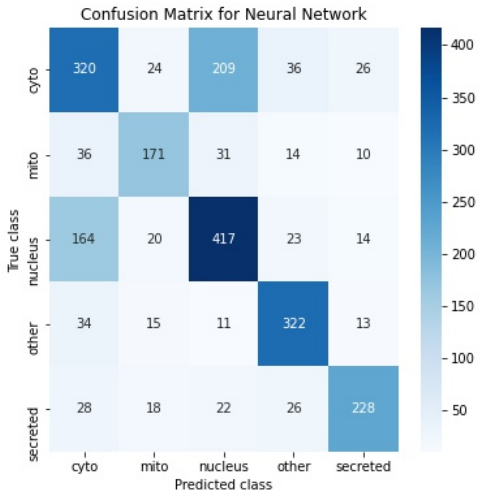

Figure 3: Confusion matrix for Neural Network model

3.3.1 Metrics calculated from Confusion matrices

The per-class Precision, Recall and F1 metrics and their "macro-" values (mean) for the SVM and Neural Network models are presented in Tables 6 and 7.

	Precision	Recall	F1
Cyto	0.543	0.566	0.554
Mito	0.736	0.691	0.713
Nucleus	0.621	0.622	0.622
Other	0.843	0.868	0.855
Secreted	0.819	0.761	0.789
Mean	0.7124	0.7016	**0.7066**

Table 6: Precision, Recall and F1 metrics for SVM

	Precision	Recall	F1
Cyto	0.550	0.520	0.535
Mito	0.690	0.653	0.671
Nucleus	0.604	0.654	0.628
Other	0.765	0.815	0.789
Secreted	0.784	0.708	0.744
Mean	0.6784	0.6699	**0.6733**

Table 7: Precision, Recall and F1 metrics for Neural Network

3.4 Performance on blind test sequences

The class predictions for the blind test sequences, along with their confidence (represented as High/Medium/Low where High = ≥ 0.70, Medium = between 0.5-0.7 and Low = < 0.5), are presented in Table 8. These were predicted using the best model (SVM, macro-F1 score = 0.7066).

4 Discussion

Over the course of this study, two machine learning models, namely an SVM and a neural network were trained and analysed for their performance on the protein subcellular location prediction task. SVMs were the first area of focus due to the benefit of using kernel tricks which could expand the dimensionality of the dataset and help to find a decision boundary that separates all five classes. Neural networks were chosen due to their ability to learn highly complex functions that could represent the relationship between the inputs sequences and output locations.

From the present findings, we can conclude that the SVM demonstrates a better performance. This is indicated by the higher macro-F1 score of the SVM (=0.7066) versus the score for the neural network (=0.6733) computed on the holdout test set. Although, it is surprising to see from the cross-validation results in Tables 3 and 4 that the mean validation set accuracy is very similar for both models. This actually makes the SVM a much more attractive choice of model, because it requires relatively lesser hyperparameter tuning to identify the best model and can achieve just as good performance.

Setting aside differences between the models, the general performance on this task is quite poor, with both models achieving around 65-67% accuracy. One major contributor to this result is the fact that the models tend to misclassify cytoplasmic and nuclear protein sequences. This is evident from the confusion matrices in Figures 2 and 3, where the off-diagonal values are largest for the cytoplasmic-nuclear and nuclear-cytoplasmic cells. This suggests that the models tend to confuse cytoplasmic sequences with nuclear sequences. Furthermore, it is seen that the models misclassify cytoplasmic sequences worse than nuclear sequences. This is reflected in the metrics from Table 6 and 7, where the precision of the cytoplasmic sub-model is the lowest of all five sub-models.

Further analysis into this problem also revealed that within the test set, there were 136 unique cytoplasmic proteins that both models classified as nuclear, and 117 nuclear sequences that both models classified as cytoplasmic. Observing the confidences of the predictions on these sequences, it was noticeable that for the cytoplasmic sequences predicted as nuclear, the second largest confidence was placed on the "Cyto" class while for the 117 nuclear sequences predicted as cytoplasmic, the second largest confidence was placed on the "nuclear" class, which validates how the model struggles to classify between cytoplasmic and nuclear sequences. To understand why this was happening, summary statistics were computed for the features of the 136 cytoplasmic and 117 nuclear sequences, for which results are presented in Tables 11 and 12 of Appendix E. All features have similar the mean values which could be a possible explanation for why the predictions were incorrect. Looking at the predictions and precision scores for the secreted and mitochondrial protein classes on the other hand, we see that both models generally perform well on classifying these sequences.

5 Conclusion and Next Steps

We conclude from our findings that the SVM demonstrates the best performance at the protein sub-cellular location task. While the overall performance is generally sub-optimal (65-67% range for accuracy), there are still many areas of improvement. A potential next step in research to improve our results is to focus is on feature analysis, particularly identifying unique features for cytoplasmic and nuclear protein sequences. This could potentially increase the accuracy of the models to the 85-90% range. At the same time, it must be considered that a drawback of using models like SVMs and deep neural networks with manual feature selection methods is that the process can be somewhat subjective (based on how

CHALLENGE SEQ01 OTHR CONFIDENCE HIGH
CHALLENGE SEQ02 MITO CONFIDENCE MEDIUM
CHALLENGE SEQ03 MITO CONFIDENCE MEDIUM
CHALLENGE SEQ04 NUCL CONFIDENCE HIGH
CHALLENGE SEQ05 CYTO CONFIDENCE MEDIUM
CHALLENGE SEQ06 MITO CONFIDENCE MEDIUM
CHALLENGE SEQ07 EXTR CONFIDENCE HIGH
CHALLENGE SEQ08 MITO CONFIDENCE LOW
CHALLENGE SEQ09 EXTR CONFIDENCE HIGH
CHALLENGE SEQ10 MITO CONFIDENCE LOW
CHALLENGE SEQ11 NUCL CONFIDENCE MEDIUM
CHALLENGE SEQ12 NUCL CONFIDENCE HIGH
CHALLENGE SEQ13 NUCL CONFIDENCE MEDIUM
CHALLENGE SEQ14 NUCL CONFIDENCE MEDIUM
CHALLENGE SEQ15 OTHR CONFIDENCE HIGH
CHALLENGE SEQ16 NUCL CONFIDENCE HIGH
CHALLENGE SEQ17 OTHR CONFIDENCE LOW
CHALLENGE SEQ18 CYTO CONFIDENCE LOW
CHALLENGE SEQ19 OTHR CONFIDENCE HIGH
CHALLENGE SEQ20 CYTO CONFIDENCE LOW

Table 8: Predictions on blind test dataset

we interpret the correlation coefficients), hence the chosen features may not be truly relevant. Therefore, another area of focus would be investigating sequence-based models such as recurrent neural networks, attention models and other such techniques derived from the natural language processing field. These models have the power to analyse sequences at the amino acid level itself, which could eliminate the loss of information when transforming data into a new feature space, potentially delivering better performance than SVMs and basic neural networks.

References

[1] Peter J. Thul, Lovisa Åkesson, Mikaela Wiking, Diana Mahdessian, Aikaterini Geladaki, Hammou Ait Blal, Tove Alm, Anna Asplund, Lars Björk, Lisa M. Breckels, Anna Bäckström, Frida Danielsson, Linn Fagerberg, Jenny Fall, Laurent Gatto, Christian Gnann, Sophia Hober, Martin Hjelmare, Fredric Johansson, Sunjae Lee, Cecilia Lindskog, Jan Mulder, Claire M. Mulvey, Peter Nilsson, Per Oksvold, Johan Rockberg, Rutger Schutten, Jochen M. Schwenk, Åsa Sivertsson, Evelina Sjöstedt, Marie Skogs, Charlotte Stadler, Devin P. Sullivan, Hanna Tegel, Casper Winsnes, Cheng Zhang, Martin Zwahlen, Adil Mardinoglu, Fredrik Pontén, Kalle von Feilitzen, Kathryn S. Lilley, Mathias Uhlén, and Emma Lundberg. A subcellular map of the human proteome. *Science*, 356(6340), 2017. ISSN 0036-8075. doi:10.1126/science.aal3321. URL https://science.sciencemag.org/content/356/6340/eaal3321.

[2] Kenneth E. Sawin and Paul Nurse. Identification of fission yeast nuclear markers using random polypeptide fusions with green fluorescent protein. *Proceedings of the National Academy of Sciences*, 93(26):15146–15151, 1996. ISSN 0027-8424. doi:10.1073/pnas.93.26.15146. URL https://www.pnas.org/content/93/26/15146.

[3] Pierre Dönnes and Annette Höglund. Predicting protein subcellular localization: Past, present, and future. *Genomics, Proteomics Bioinformatics*, 2(4):209–215, 2004. ISSN 1672-0229. doi:https://doi.org/10.1016/S1672-0229(04)02027-3. URL https://www.sciencedirect.com/science/article/pii/S1672022904020273.

[4] Juan Cedano, Patrick Aloy, Josep A. Pérez-Pons, and Enrique Querol. Relation between amino acid composition and cellular location of proteins11edited by f. e. cohen. *Journal of Molecular Biology*, 266(3):594–600, 1997. ISSN 0022-2836. doi:https://doi.org/10.1006/jmbi.1996.0804. URL https://www.sciencedirect.com/science/article/pii/S0022283696908049.

[5] Chin-Sheng Yu, Chih-Jen Lin, and Jenn-Kang Hwang. Predicting subcellular localization of proteins for gram-negative bacteria by support vector machines based on n-peptide compositions. *Protein Science*, 13(5):1402–1406, 2004. doi:https://doi.org/10.1110/ps.03479604. URL https://onlinelibrary.wiley.com/doi/abs/10.1110/ps.03479604.

[6] Richard Mott, Jörg Schultz, Peer Bork, and Chris P Ponting. Predicting protein cellular localization using a domain projection method. *Genome research*, 12(8):1168–1174, 2002. ISSN 1088-9051.

[7] Edward M. Marcotte, Ioannis Xenarios, Alexander M. Van der Bliek, and David Eisenberg. Localizing

proteins in the cell from their phylogenetic profiles. *Proceedings of the National Academy of Sciences - PNAS*, 97(22):12115–12120, 2000. ISSN 0027-8424.

[8] Emily Chia-Yu Su, Hua-Sheng Chiu, Allan Lo, Jenn-Kang Hwang, Ting-Yi Sung, and Wen-Lian Hsu. Protein subcellular localization prediction based on compartment-specific features and structure conservation. *BMC bioinformatics*, 8(1):330–330, 2007. ISSN 1471-2105.

[9] A. Reinhardt and T. Hubbard. Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Research*, 26(9):2230–2236, 05 1998. ISSN 0305-1048. doi:10.1093/nar/26.9.2230. URL `https://doi.org/10.1093/nar/26.9.2230`.

[10] Kenta Nakai and Paul Horton. Psort: a program for detecting sorting signals in proteins and predicting their subcellular localization. *Trends in biochemical sciences (Amsterdam. Regular ed.)*, 24(1):34–35, 1999. ISSN 0968-0004.

[11] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.

6 Appendices

6.1 A: Code briefing

1. Libraries required: BioPython, NumPy, Pandas, os, re, Scikitlearn, Seaborn, Matplotlib

2. Feature transformation: A class `Data(raw_data, features, lbl_format, scale)` was created to perform the feature transformation and preprocessing. `raw_data` is a dictionary of the sequences and other meta-data such as the identifiers. `features` is a list of the desired features to include, for example it can be `['seqlen', 'mol_wt', 'aromaticity']` to include just the sequence length, molecular weight, aromaticity in the feature space or `'all'` to include all features. `scale` processes the features by standardization or normalization. `lbl_format` specifies the encoding of labels, as one-hot encoded form or plain numerical encoding.

3. Feature analysis: Pandas function `.corr()` was used to compute correlation coefficients for each of the features and seaborn was used to visualize the correlations as heatmaps.

4. Model Development:

   (a) SVM: scikitlearn's SVM module was used.

   (b) Neural Network: Neural networks were implemented using Pytorch.

      i. `seqData` : a python class inherited from Pytorch `Dataset` class. This accepts the data matrix and true labels, which it will convert into PyTorch tensors.

      ii. `MLP(Layers, p)` : The deep neural network class which inherits from `nn.Module`. Can take any architecture as specified in `Layers`, a list containing number of nodes in each layer of network. The dropout to apply on network can be specified through `p`.

5. Model training: SVM

   (a) Data splitting: This was performed using Scikitlearn `train_test_split` function.

   (b) Grid search + Cross validation : For performing cross-validation, a function `crossval(data, model, nFolds, shuffle)` was created. `data` is the feature data. `model` is the model used, `nFolds` is the number of folds that will be performed, and `shuffle` accepts a boolean to shuffle/not shuffle the data before cross-validation. The grid search over

hyperparameters is performed using simple for-loops.

6. Model training: Neural network

   (a) Data splitting : This was performed using Scikitlearn `train_test_split` function.

   (b) `train()` : This function performs model training and returns a dictionary `history` which contains the model loss and accuracies per epoch. It inputs the model to be trained (which will be an instance of `MLP()`), a tuple of the training and validation datasets (which are instances of `seqData`), and other hyperparameters such as the `epochs`, `batch_size`, `optimizer`, `lr` (learning rate). Hyperparameter settings are provided in Table 2.

   (c) `evaluate()` : Evaluates the model on data by computing the loss and accuracy. Accepts variable `X` and `y` which are the inputs and true outputs, the loss function `nn.CrossEntropy()` and the model instance itself.

   (d) Grid-search + Cross validation: The cross-validation function from "Model training : SVM" was used, modified to deal with pytorch models. Grid search was performed using for-loops over range of hyperparameters.

7. Model Evaluation

   (a) Performance on test set with confidence estimation: For the SVM, this is performed by `model.score()` which accepts the test data and its true labels to compute accuracy. For confidence estimation, the `model.predict_proba()` function is used which implements Platt scaling. This takes the test data and creates a $(n_{data}, 5)$ matrix of the class probabilities for each data point. For the neural network, the `model` is applied on the test data and then probabilities are created using `nn.Softmax()`. These probabilities are converted to final predictions using `np.argmax()`.

   (b) Confusion matrices : created using scikitlearn's `confusion_matrix` function

   (c) Metrics: Precision, recall and F1 score are computed using scikitlearn's `precision_score`, `recall_score` and `f1_score` functions.

6.2 B: Correlation matrix of feature set

The correlation matrix for all 46 features is presented in Figure 4.

6.3 C: Cross-validation raw data for SVM and Neural Network

Below are the cross-validation results for each variant of the SVM. The results are in the form of a (5, 2) matrix where the first column stores the training set accuracy and the second column stores the validation set accuracies and each of the five rows corresponds to the folds of cross-validation.

```
C = 1, gamma = 2^-7.0
[[0.69714366 0.65845465]
 [0.69784374 0.63829787]
 [0.6925231  0.66069429]
 [0.69634607 0.65546218]
 [0.69872603 0.64313725]]
C = 1, gamma = 2^-8.0
 [[0.65527863 0.63941769]
 [0.65905909 0.62038074]
 [0.65359843 0.64221725]
 [0.65854683 0.64257703]
 [0.65966681 0.62128852]]
C = 1, gamma = 2^-9.0
 [[0.62797536 0.6237402 ]
 [0.63651638 0.61478163]
 [0.63035564 0.6181411 ]
 [0.62956741 0.61848739]
 [0.63558729 0.60448179]]
C = 1, gamma = 2^-10.0
 [[0.60795295 0.59966405]
 [0.60963316 0.60022396]
 [0.60739289 0.59406495]
 [0.60828783 0.59719888]
 [0.6102478  0.58263305]]
C = 1, gamma = 2^-11.0
 [[0.58429011 0.57894737]
 [0.5875105  0.57950728]
 [0.5841501  0.57782755]
 [0.58868823 0.57591036]
 [0.58672827 0.56526611]]
C = 1, gamma = 2^-12.0
 [[0.52646318 0.52239642]
 [0.53626435 0.52967525]
 [0.52758331 0.52967525]
 [0.5349293  0.52661064]
 [0.53772925 0.51820728]]
C = 1, gamma = 2^-13.0
 [[0.47535704 0.48152296]
 [0.48179782 0.45968645]
 [0.47619714 0.48488242]
 [0.47529049 0.47619048]
 [0.47753045 0.47002801]]
C = 1, gamma = 2^-14.0
 [[0.37538505 0.37737962]
 [0.38364604 0.38913774]
 [0.37174461 0.38073908]
 [0.37365253 0.38151261]
 [0.38723226 0.36246499]]
```

```
C = 10, gamma = 2^-7.0
 [[0.81545785 0.66181411]
 [0.81489779 0.65397536]
 [0.81489779 0.66909295]
 [0.81562369 0.67114846]
 [0.81296374 0.67058824]]
C = 10, gamma = 2^-8.0
 [[0.74096892 0.6556551 ]
 [0.74040885 0.64109742]
 [0.73564828 0.66629339]
 [0.73764525 0.66610644]
 [0.73582528 0.66834734]]
C = 10, gamma = 2^-9.0
 [[0.68636236 0.65229563]
 [0.69154299 0.63213886]
 [0.68370204 0.65509518]
 [0.68724626 0.65770308]
 [0.68724626 0.63697479]]
C = 10, gamma = 2^-10.0
 [[0.65065808 0.63381859]
 [0.65541865 0.62038074]
 [0.64939793 0.63885778]
 [0.65252695 0.63137255]
 [0.65392692 0.61904762]]
C = 10, gamma = 2^-11.0
 [[0.63091571 0.61366181]
 [0.63525623 0.61422172]
 [0.62881546 0.61982083]
 [0.62900742 0.61904762]
 [0.63250735 0.60392157]]
C = 10, gamma = 2^-12.0
 [[0.6157939  0.60470325]
 [0.61761411 0.6019037 ]
 [0.61397368 0.60526316]
 [0.61332773 0.60392157]
 [0.61962761 0.59271709]]
C = 10, gamma = 2^-13.0
 [[0.6024923  0.59406495]
 [0.60725287 0.5912654 ]
 [0.60193223 0.59966405]
 [0.60632787 0.59327731]
 [0.60758785 0.57983193]]
C = 10, gamma = 2^-14.0
 [[0.5875105  0.5806271 ]
 [0.58877065 0.57950728]
 [0.58611033 0.57614782]
 [0.59106818 0.57927171]
 [0.59064819 0.57142857]]
C = 100, gamma = 2^-7.0
 [[0.96863624 0.64389698]
 [0.95715486 0.6237402 ]
 [0.96359563 0.63829787]
 [0.96836063 0.63809524]
 [0.96234075 0.62072829]]
C = 100, gamma = 2^-8.0
 [[0.85634276 0.6450168 ]
 [0.86110333 0.62653975]
 [0.85872305 0.64893617]
 [0.86070279 0.65098039]
 [0.86196276 0.64761905]]
C = 100, gamma = 2^-9.0
 [[0.77975357 0.65397536]
 [0.77961355 0.63829787]
 [0.774993   0.66349384]
 [0.77852443 0.65938375]
 [0.77740445 0.65938375]]
C = 100, gamma = 2^-10.0
 [[0.72220666 0.65397536]
 [0.72234668 0.64445689]
 [0.71268552 0.66517357]
 [0.71692566 0.65658263]
 [0.71636567 0.66106443]]
C = 100, gamma = 2^-11.0
 [[0.67292075 0.65285554]
 [0.67684122 0.62765957]
 [0.66984038 0.65341545]
 [0.6750665  0.64593838]
 [0.67562649 0.62857143]]
C = 100, gamma = 2^-12.0
 [[0.64295715 0.63325868]
 [0.64631756 0.61926092]
 [0.63875665 0.63605823]
 [0.64412712 0.62913165]
 [0.64706706 0.61288515]]
C = 100, gamma = 2^-13.0
 [[0.62629516 0.61030235]
 [0.63049566 0.6075028 ]
 [0.62517502 0.62430011]
 [0.62914742 0.61512605]
 [0.63166737 0.60560224]]
C = 100, gamma = 2^-14.0
 [[0.6183142  0.5968645 ]
 [0.6208345  0.60134378]
 [0.61355363 0.61534155]
 [0.61878762 0.60504202]
 [0.62200756 0.58711485]]
```

Next, we present the raw cross-validation results for each variant of the Neural Network. The results are in the form of a matrix where the columns are the train loss, validation loss, train accuracy and validation accuracy. The rows correspond to each cross-validation fold.

```
Layers = [45, 5], Dropout = 0.0
[[0.989 1.004 0.605 0.580]
 [0.974 1.024 0.606 0.596]
 [0.984 0.980 0.604 0.607]
 [0.987 0.970 0.608 0.592]
 [0.971 1.034 0.607 0.583]]
Layers = [45, 5], Dropout = 0.1
 [[0.988 1.003 0.607 0.583]
 [0.974 1.023 0.605 0.597]
 [0.984 0.980 0.603 0.606]
 [0.987 0.970 0.608 0.591]
 [0.971 1.034 0.607 0.583]]
Layers = [45, 5], Dropout = 0.2
 [[0.988 1.003 0.607 0.583]
 [0.974 1.023 0.605 0.597]
 [0.984 0.980 0.603 0.606]
 [0.987 0.970 0.608 0.591]
 [0.971 1.034 0.607 0.583]]
Layers = [45, 5], Dropout = 0.3
 [[0.988 1.003 0.607 0.583]
 [0.974 1.023 0.605 0.597]
 [0.984 0.980 0.603 0.606]
 [0.987 0.970 0.608 0.591]
 [0.971 1.034 0.607 0.583]]
Layers = [45, 16, 5], Dropout = 0.0
 [[0.819 0.899 0.661 0.610]
 [0.780 0.852 0.678 0.630]
```

```
 [0.766 0.809 0.681 0.671]
 [0.757 0.789 0.686 0.668]
 [0.739 0.827 0.688 0.667]]
Layers = [45, 16, 5], Dropout = 0.1
 [[0.880 0.940 0.633 0.591]
 [0.853 0.916 0.644 0.610]
 [0.854 0.871 0.641 0.629]
 [0.845 0.861 0.644 0.639]
 [0.823 0.922 0.652 0.643]]
Layers = [45, 16, 5], Dropout = 0.2
 [[0.931 0.981 0.610 0.569]
 [0.909 0.958 0.616 0.584]
 [0.912 0.928 0.616 0.597]
 [0.896 0.916 0.626 0.620]
 [0.875 0.991 0.628 0.607]]
Layers = [45, 16, 5], Dropout = 0.3
 [[0.981 1.016 0.599 0.574]
 [0.957 0.994 0.599 0.562]
 [0.962 0.968 0.598 0.586]
 [0.945 0.971 0.603 0.593]
 [0.929 1.020 0.609 0.583]]
Layers = [45, 32, 5], Dropout = 0.0
 [[0.750 0.870 0.688 0.632]
 [0.696 0.800 0.719 0.665]
 [0.673 0.749 0.730 0.681]
 [0.657 0.735 0.736 0.691]
 [0.635 0.767 0.738 0.711]]
Layers = [45, 32, 5], Dropout = 0.1
 [[0.812 0.903 0.665 0.613]
 [0.775 0.866 0.682 0.629]
 [0.758 0.813 0.682 0.656]
 [0.751 0.838 0.692 0.655]
 [0.727 0.851 0.701 0.661]]
Layers = [45, 32, 5], Dropout = 0.2
 [[0.853 0.933 0.656 0.593]
 [0.814 0.904 0.658 0.610]
 [0.804 0.863 0.668 0.639]
 [0.798 0.865 0.666 0.662]
 [0.768 0.892 0.674 0.645]]
Layers = [45, 32, 5], Dropout = 0.3
 [[0.893 0.962 0.643 0.578]
 [0.856 0.942 0.644 0.601]
 [0.845 0.892 0.646 0.626]
 [0.837 0.892 0.653 0.621]
 [0.805 0.919 0.660 0.641]]
```

6.4 D: Sample predictions of SVM and neural network on test set

The predictions of the SVM and Neural Network models on 20 sample sequences from the test set are presented in Tables 9 and 10. The final prediction is the class with the greatest probability, highlighted in yellow.

6.5 E: Summary statistic for commonly misclassified cytoplasmic and nuclear proteins

To compare the cytoplasmic and nuclear sequences that were "confused" in both models, summary statistics were computed for the 45 features. These are presented in Tables 11 and 12.

	cyto	mito	nucleus	other	secreted
0	0.424113	0.311285	0.143910	0.085788	0.034904
1	0.051494	0.049330	0.017454	0.251358	0.630364
2	0.808928	0.050235	0.096800	0.023309	0.020728
3	0.074751	0.086184	0.040559	0.771037	0.027469
4	0.453267	0.113294	0.309718	0.038467	0.085253
5	0.228516	0.023876	0.084569	0.016099	0.646940
6	0.422464	0.110218	0.010777	0.008328	0.448213
7	0.325229	0.362526	0.264230	0.032175	0.015840
8	0.585444	0.018281	0.378213	0.002149	0.015912
9	0.554836	0.004509	0.439640	0.000767	0.000247
10	0.258752	0.022560	0.105534	0.600820	0.012333
11	0.416950	0.010575	0.571795	0.000298	0.000382
12	0.007947	0.000300	0.000258	0.000258	0.991237
13	0.005288	0.001181	0.000871	0.992558	0.000101
14	0.215573	0.568407	0.090456	0.051155	0.074409
15	0.204174	0.001619	0.790447	0.001773	0.001988
16	0.566811	0.015208	0.080246	0.023838	0.313896
17	0.645000	0.030483	0.205075	0.109641	0.009802
18	0.001301	0.001893	0.002138	0.000005	0.994663
19	0.583731	0.025450	0.347193	0.038849	0.004778

Table 9: Sample predictions of SVM model on test set

	cyto	mito	nucleus	other	secreted
0	0.3676787	0.3601055	0.1791812	0.0644130	0.0286216
1	0.0929130	0.0810931	0.0235273	0.5714332	0.2310334
2	0.7489110	0.0283294	0.1048155	0.0883761	0.0295680
3	0.0528309	0.0522961	0.0201097	0.8578848	0.0168787
4	0.2268762	0.0854167	0.5721152	0.0196546	0.0959372
5	0.1939268	0.0332159	0.1475430	0.0222720	0.6030424
6	0.1760980	0.1638564	0.0007894	0.0339804	0.6252757
7	0.3732996	0.3865525	0.1895542	0.0106981	0.0398957
8	0.3573385	0.0194676	0.5796671	0.0006266	0.0429002
9	0.5259020	0.0057225	0.4678053	0.0002207	0.0003494
10	0.1680456	0.0221073	0.0735017	0.7184054	0.0179401
11	0.4027032	0.0052567	0.5913814	0.0001643	0.0004944
12	0.0473804	0.0021528	0.0008644	0.0005605	0.9490419
13	0.0032484	0.0026602	0.0004791	0.9934690	0.0001433
14	0.2200546	0.5239266	0.1396801	0.0255986	0.0907402
15	0.2478486	0.0074273	0.7432650	0.0001480	0.0013110
16	0.4152693	0.0176836	0.0522589	0.0050615	0.5097267
17	0.5710899	0.0549596	0.2333052	0.1176442	0.0230011
18	0.0031681	0.0001558	0.0076256	0.0000002	0.9890504
19	0.4968877	0.0281772	0.4305609	0.0390130	0.0053612

Table 10: Sample predictions of Neural Network model on test set

	count	mean	std	min	25%	50%	75%	max
seqlen	136.0	634.7206	386.0173	113.0000	355.0000	561.0000	838.5000	2542.0000
isoe_pt	136.0	7.1110	1.7588	4.1261	5.3810	6.9156	8.8331	10.1360
global_A	136.0	0.0611	0.0234	0.0223	0.0454	0.0591	0.0716	0.1582
global_C	136.0	0.0146	0.0103	0.0000	0.0075	0.0130	0.0204	0.0615
global_D	136.0	0.0562	0.0167	0.0190	0.0447	0.0542	0.0657	0.1010
global_E	136.0	0.0786	0.0262	0.0304	0.0618	0.0738	0.0901	0.1643
global_F	136.0	0.0330	0.0126	0.0000	0.0236	0.0329	0.0429	0.0663
global_G	136.0	0.0521	0.0212	0.0099	0.0388	0.0501	0.0635	0.1231
global_H	136.0	0.0252	0.0106	0.0056	0.0180	0.0244	0.0313	0.0596
global_I	136.0	0.0450	0.0161	0.0035	0.0343	0.0427	0.0534	0.0882
global_K	136.0	0.0716	0.0273	0.0122	0.0537	0.0692	0.0863	0.1544
global_L	136.0	0.0844	0.0211	0.0220	0.0682	0.0868	0.0996	0.1283
global_M	136.0	0.0211	0.0075	0.0035	0.0162	0.0204	0.0260	0.0417
global_N	136.0	0.0512	0.0185	0.0063	0.0406	0.0517	0.0639	0.1182
global_P	136.0	0.0603	0.0265	0.0197	0.0416	0.0570	0.0698	0.2053
global_Q	136.0	0.0474	0.0201	0.0063	0.0359	0.0439	0.0562	0.1605
global_R	136.0	0.0536	0.0187	0.0108	0.0419	0.0538	0.0635	0.1266
global_S	136.0	0.1011	0.0255	0.0442	0.0810	0.1003	0.1195	0.1758
global_T	136.0	0.0577	0.0135	0.0249	0.0497	0.0573	0.0647	0.1002
global_V	136.0	0.0516	0.0119	0.0207	0.0445	0.0506	0.0592	0.0956
global_W	136.0	0.0091	0.0061	0.0000	0.0045	0.0079	0.0123	0.0334
global_Y	136.0	0.0252	0.0114	0.0000	0.0176	0.0248	0.0333	0.0537
local_A	136.0	0.0628	0.0500	0.0000	0.0200	0.0600	0.0800	0.2200
local_C	136.0	0.0113	0.0199	0.0000	0.0000	0.0000	0.0200	0.1000
local_D	136.0	0.0634	0.0436	0.0000	0.0400	0.0600	0.0800	0.2000
local_E	136.0	0.0815	0.0509	0.0000	0.0400	0.0800	0.1000	0.3200
local_F	136.0	0.0249	0.0244	0.0000	0.0000	0.0200	0.0400	0.1000
local_G	136.0	0.0609	0.0579	0.0000	0.0200	0.0500	0.0800	0.2800
local_H	136.0	0.0234	0.0267	0.0000	0.0000	0.0200	0.0400	0.1400
local_I	136.0	0.0315	0.0277	0.0000	0.0200	0.0200	0.0400	0.1200
local_K	136.0	0.0674	0.0518	0.0000	0.0350	0.0600	0.1000	0.2200
local_L	136.0	0.0754	0.0461	0.0000	0.0400	0.0700	0.1000	0.2600
local_M	136.0	0.0365	0.0220	0.0200	0.0200	0.0200	0.0400	0.1200
local_N	136.0	0.0500	0.0454	0.0000	0.0200	0.0400	0.0800	0.2000
local_P	136.0	0.0688	0.0523	0.0000	0.0400	0.0600	0.1000	0.3000
local_Q	136.0	0.0488	0.0428	0.0000	0.0200	0.0400	0.0650	0.2800
local_R	136.0	0.0566	0.0426	0.0000	0.0200	0.0600	0.0800	0.2600
local_S	136.0	0.1065	0.0583	0.0000	0.0600	0.1000	0.1400	0.3400
local_T	136.0	0.0541	0.0415	0.0000	0.0200	0.0400	0.0800	0.2600
local_V	136.0	0.0493	0.0366	0.0000	0.0200	0.0400	0.0600	0.1800
local_W	136.0	0.0057	0.0124	0.0000	0.0000	0.0000	0.0000	0.0800
local_Y	136.0	0.0213	0.0238	0.0000	0.0000	0.0200	0.0400	0.1200
aromacity	136.0	0.0673	0.0218	0.0190	0.0545	0.0668	0.0820	0.1207
instability_index	136.0	53.8492	10.8688	32.0025	47.2632	53.3583	59.3084	94.6690
gravy	136.0	-0.6785	0.2599	-1.3730	-0.8353	-0.6617	-0.4772	-0.1764

Table 11: Summary statistics for misclassified cytoplasmic proteins in SVM and Neural Network models

	count	mean	std	min	25%	50%	75%	max
seqlen	117.0	783.3077	648.2847	81.0000	361.0000	594.0000	1056.0000	4717.0000
isoe_pt	117.0	6.2663	1.2292	4.5226	5.4183	6.0155	6.7291	10.3107
global_A	117.0	0.0677	0.0169	0.0275	0.0571	0.0660	0.0783	0.1175
global_C	117.0	0.0177	0.0104	0.0000	0.0119	0.0168	0.0230	0.0544
global_D	117.0	0.0525	0.0121	0.0167	0.0450	0.0514	0.0608	0.0838
global_E	117.0	0.0813	0.0240	0.0255	0.0665	0.0775	0.0947	0.1479
global_F	117.0	0.0363	0.0137	0.0000	0.0280	0.0344	0.0447	0.0735
global_G	117.0	0.0565	0.0192	0.0186	0.0445	0.0539	0.0676	0.1111
global_H	117.0	0.0236	0.0079	0.0075	0.0175	0.0232	0.0285	0.0442
global_I	117.0	0.0497	0.0176	0.0130	0.0371	0.0487	0.0616	0.0988
global_K	117.0	0.0623	0.0180	0.0231	0.0510	0.0619	0.0724	0.1111
global_L	117.0	0.1012	0.0249	0.0395	0.0848	0.1001	0.1160	0.1922
global_M	117.0	0.0225	0.0074	0.0090	0.0172	0.0211	0.0275	0.0415
global_N	117.0	0.0399	0.0160	0.0045	0.0307	0.0387	0.0479	0.1032
global_P	117.0	0.0537	0.0222	0.0174	0.0383	0.0497	0.0659	0.1710
global_Q	117.0	0.0489	0.0165	0.0095	0.0388	0.0464	0.0571	0.1122
global_R	117.0	0.0551	0.0155	0.0112	0.0449	0.0549	0.0645	0.1169
global_S	117.0	0.0821	0.0223	0.0225	0.0649	0.0805	0.0971	0.1618
global_T	117.0	0.0506	0.0132	0.0257	0.0433	0.0493	0.0574	0.0931
global_V	117.0	0.0592	0.0157	0.0255	0.0480	0.0589	0.0701	0.1111
global_W	117.0	0.0112	0.0075	0.0000	0.0060	0.0102	0.0141	0.0415
global_Y	117.0	0.0279	0.0101	0.0000	0.0218	0.0268	0.0325	0.0665
local_A	117.0	0.0694	0.0432	0.0000	0.0400	0.0600	0.1000	0.2200
local_C	117.0	0.0169	0.0206	0.0000	0.0000	0.0200	0.0200	0.1000
local_D	117.0	0.0526	0.0340	0.0000	0.0200	0.0400	0.0800	0.2200
local_E	117.0	0.0807	0.0454	0.0000	0.0400	0.0800	0.1200	0.2200
local_F	117.0	0.0338	0.0278	0.0000	0.0200	0.0400	0.0400	0.1000
local_G	117.0	0.0631	0.0462	0.0000	0.0200	0.0600	0.0800	0.2400
local_H	117.0	0.0207	0.0200	0.0000	0.0000	0.0200	0.0400	0.0800
local_I	117.0	0.0506	0.0358	0.0000	0.0200	0.0400	0.0800	0.1600
local_K	117.0	0.0609	0.0409	0.0000	0.0400	0.0600	0.0800	0.1600
local_L	117.0	0.0880	0.0433	0.0000	0.0600	0.0800	0.1200	0.2200
local_M	117.0	0.0349	0.0184	0.0200	0.0200	0.0400	0.0400	0.1000
local_N	117.0	0.0369	0.0280	0.0000	0.0200	0.0400	0.0600	0.1200
local_P	117.0	0.0540	0.0351	0.0000	0.0200	0.0400	0.0800	0.1800
local_Q	117.0	0.0451	0.0298	0.0000	0.0200	0.0400	0.0600	0.1200
local_R	117.0	0.0583	0.0386	0.0000	0.0200	0.0400	0.0800	0.1800
local_S	117.0	0.0879	0.0505	0.0000	0.0400	0.0800	0.1200	0.2800
local_T	117.0	0.0523	0.0330	0.0000	0.0400	0.0400	0.0800	0.1600
local_V	117.0	0.0571	0.0368	0.0000	0.0400	0.0600	0.0800	0.1600
local_W	117.0	0.0111	0.0154	0.0000	0.0000	0.0000	0.0200	0.0600
local_Y	117.0	0.0256	0.0260	0.0000	0.0000	0.0200	0.0400	0.1400
aromacity	117.0	0.0753	0.0210	0.0247	0.0628	0.0743	0.0889	0.1344
instability_index	117.0	48.4544	10.4867	8.4775	42.3365	48.9728	53.9233	81.6200
gravy	117.0	-0.4346	0.2318	-1.0911	-0.6110	-0.4068	-0.2603	0.1829

Table 12: Summary statistics for misclassified nuclear proteins in both SVM and Neural Network models

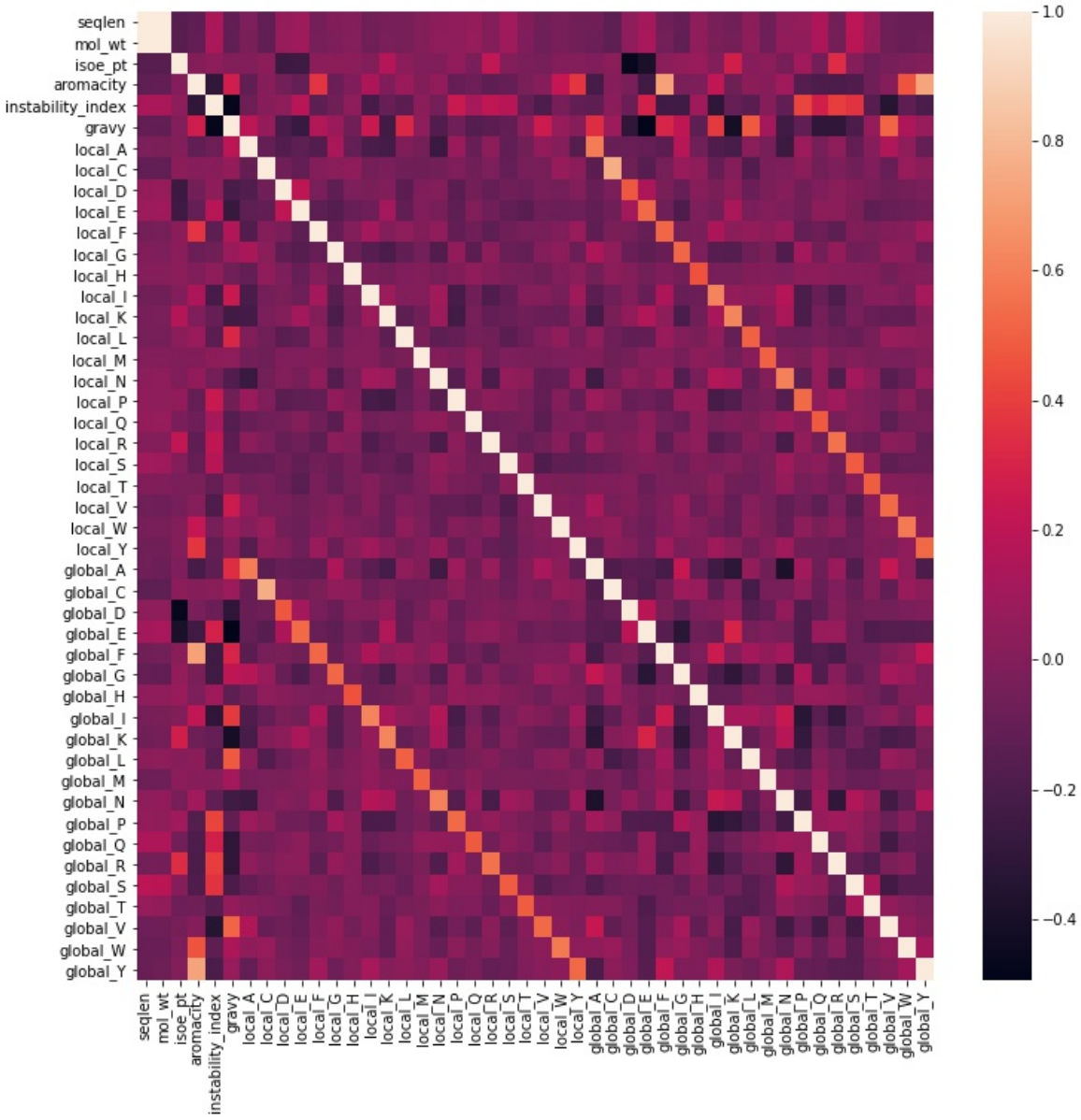

Figure 4: Pearson Correlation coefficient measured between the features and visualized as a heat map