November 29, 2023

CMSC 435: Introduction to Data Science

Assignment: Final Report

It Depends:

Saifuding Daniyaer

Brian Dao

Adam Harms

Ian Jaffe

Danielle Joan Mapili

Descriptions of Our Design:

Initially, our plan was to get a basic iteration out to gain a feel for the process and have some numbers to improve upon. We started by generating a set of features to use in RapidMiner. Below is a list of our comprehensive features and description:

- Composition Percent of each protein
    - For a given sequence, the percent composition of each amino acid was calculated.
- Length
    - The length of each sequence was generated.
- Hydrophobicity
    - Each amino acid is given a hydrophobicity, the sequence is summed by each amino acid's hydrophobicity. The higher the value the more hydrophobic. The lower the value the more hydrophilic.
- Net Charge
    - In a similar fashion to hydrophobicity, each amino acid is given a 1,0, or -1 to indicate the charge. This is then summed for all aminos in the sequence.
- Dipeptide composition
    - Dipeptide composition is a method for encoding protein sequences into a numerical form that reflects the local sequential properties of amino acids.

Dipeptide composition was added for the second iteration of the project. We believed that all of these features could be linked to individual or multiple labels hoping they would help us when we received the blind test dataset. We decided to keep all features for the final iteration to try and receive the highest MCC value possible.

For the algorithm choice we picked a simple and fast algorithm to get our initial version out as soon as possible. Below is our list of algorithm choices and a description of each:

- Decision Tree
    - Decision Tree was our simple and fast algorithm for the initial version. This was performed with default parameters as we knew this wouldn't be our main algorithm as it doesnt like continuous data.
- Knn
    - This algorithm was chosen because we read Knn was useful for numerical datasets which our dataset was heavily reliant upon. The K value was adjusted to 7 to try and get more accuracy with more neighbors with little success. Next

measure types and numerical measures were adjusted to Numerical Measures and Manhattan Distance respectively. There were some gains to accuracy but mostly losses to the MCC values (and lower avg) so we decided to scrap Knn for our last algorithm.

- Deep Learning
    - Deep learning is renowned for its ability to detect complex, non-linear patterns in large datasets. Protein sequences often exhibit such complex patterns that are crucial for determining their function or type. Deep learning models, especially those with multiple layers (deep architectures), are adept at uncovering these intricate patterns. The activation parameter was changed to Maxout to test and it proved to be better than the other parameters we tried, so we stuck with that for the final solution.

We also considered using clustering to try and map out the labels in N-D space and potentially be able to label the clusters, but we decided it was too time intensive and it would work better for an unsupervised dataset. This led us to Deep Learning which was our last algorithm and provided us with the highest MCC values.

Results:

Table 1. Summary of results based on the 5-fold cross validation on the training dataset.

Outcome	Quality measure	Baseline result	Design 1	Design 2	Design 3	Best Design
DNA	*Sensitivity*	6.9	16.1	21.7	18.9	18.9
	Specificity	99.3	97.7	97.7	98.3	98.3
	Accuracy	95.2	94.1	94.3	94.8	94.8
	MCC	**0.132**	**0.170**	**0.227**	**0.232**	**0.232**
RNA	*Sensitivity*	39.6	35.6	41.9	42.4	42.4
	Specificity	98.9	98.4	97.9	98.3	98.3
	Accuracy	95.3	50.0	50.0	50.0	50.0
	MCC	**0.501**	**0.427**	**0.458**	**0.487**	**0.487**
DRNA	*Sensitivity*	4.5	0	9.1	9.1	9.1
	Specificity	100.0	100.0	100.0	99.9	99.9
	Accuracy	99.7	99.7	99.7	99.7	99.7
	MCC	**0.122**	**-0.001**	**0.190**	**0.16**	**0.16**
nonDRNA	*Sensitivity*	98.6	96.7	96.2	97.1	97.1
	Specificity	29.8	34.3	39.7	37.4	37.4
	Accuracy	91.3	90.1	90.2	90.7	90.7
	MCC	**0.428**	**0.386**	**0.416**	**0.43**	**0.43**
averageMCC		**0.296**	**0.246**	**0.323**	**0.327**	**0.327**
accuracy4labels		90.8	89.3	89.4	90.13	90.13

Table 2. Confusion matrix for our best model (Deep Learning).

	True nonDRNA	True RNA	True DNA	True DRNA	**Class Precision**
pred. nonDRNA	7629	273	294	19	**92.87%**
pred. RNA	116	222	22	0	**61.67%**
pred. DNA	110	28	74	1	**34.74%**
pred. DRNA	4	0	1	2	**28.57%**
Class Recall	**97.07%**	**42.45%**	**18.93%**	**9.09%**	

In our model, we added an aggregate operator so it outputs a count summary of each predicted classification on the blind test dataset. Our prediction summary:

- DNA: 233
- DRNA: 3
- RNA: 366
- nonDRNA: 8192

Table 3. Confusion matrix of our initial model (Decision Tree)

	true nonDRNA	true RNA	true DNA	true DRNA	class precision
pred. nonDRNA	7601	290	305	20	92.51%
pred. RNA	112	186	23	1	57.76%
pred. DNA	143	47	63	1	24.80%
pred. DRNA	3	0	0	0	0.00%
class recall	96.72%	35.56%	16.11%	0.00%	

Comparing our initial matrix to our best matrix shows us some interesting results. Our Decision Tree model was not able to correctly predict any DRNA, which significantly lowered our MCC to below baseline. Our deep learning model was able to correctly predict some, but not all though still significantly improving our MCC. Additionally our initial model was less accurate than our best model across the board, which shows significant improvement. One thing to note is that even though the MCC for our initial was around .1 less than our best solution,

accuracy for label was generally close to our best model (around .8 off) which means Decision Tree could potentially perform well given optimal parameters.

Conclusion:

On a broader scale, our best design's MCC value is greater than the baseline which generally reflects a better overall predictive performance. However, using Table 1, we can take a deeper look at each outcome comparing them and its quality.

- DNA: Our MCC is higher compared to the baseline because, even though our specificity and accuracy are a bit lower, our sensitivity is three times greater.
- DRNA: Our MCC is higher compared to the baseline because our model has a higher sensitivity (2x more). There is little difference in our model's accuracy and specificity compared to the baseline.
- RNA: Our MCC is lower compared to the baseline because our model is less than half of the accuracy of the baseline. The other values are similar in nature.
- nonDRNA: Our MCC is higher compared to the baseline because even though our sensitivity and accuracy are a bit lower, our specificity is greater.

Overall for each outcome, one of the measures of quality was significantly different, differing our MCC values from the baseline, and three times out of four resulting in a higher MCC value, thus potentially making our deep learning model better.

We think our model is close but generally a little bit worse based on the accuracy for label section (90.8 vs 90.13) which could be seen as a disadvantage versus the baseline, but our MCC is about 1.5x higher than baseline which is a clear advantage of our method. The parameters of Deep Learning are generally simple (less options) compared to the parameters of an algorithm like SVM which was beneficial to us. Deep Learning was CPU intensive, so for larger data sets it could pose a problem but for our project it seemed like a good choice. We think because of our iterative approach we were able to gain a better solution overall because we were able to finely tune our method to optimize one thing at a time though it was a slower process.

To talk about our experience with the project, it was nice to have the opportunity to play around with the whole process by having the progress report due. By the progress report, we were familiar with what we had to do, making the rest of the project after it just exploration and documentation. We achieved this because we used the progress report to put out an initial version and generate our main features. Generally, our team had the mindset of 'everyone works on everything' in the sense that because it’s a straightforward project, it was hard to divide up and delegate small bits of work, so everyone would just hop in and do some work when and where they could. The idea of getting a test set with labels and a final set with no labels also made the project challenging, but also interesting since our initial predictions could be completely blindsided by this new data set. Factoring that in with noise in the data makes this a problem with many good solutions which we found fun experimenting with to get the best possible results. The same could be said for feature generation as the possibilities are truly endless and each could be meaningful or meaningless. Overall based on the information we gathered from the table/experimentation we think our best solution generally is better than baseline on all fronts except accuracy for labels though they are still relatively close.