

III. Model documentation and write-up

Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

1. Who are you (mini-bio) and what do you do professionally?

Grigorii Karmatskii - Beginner data scientist.

If you are on a team, please complete this block for each member of the team.

2. What motivated you to compete in this challenge?

Great way to get familiar with bioinformatics and latest approaches in the field. Opportunity to win prizes. Nice addition to my CV.

3. High level summary of your approach: what did you do and why?

I'm using a simple MLP model, trained on k-mer counts and phenotypic features. Shows high accuracy scores, it is quick to train and doesn't require a high-end PC.

4. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

Extending the sequences with its reverse complements:

```
from Bio.Seq import Seq
X_train['sequence_rc'] = X_train['sequence'].apply(lambda x:
                                                    str(Seq(x).reverse_complement()))

# then extend regular string with rev. comp.
```

Improved quality a lot and reduced the amount of features after dropping dupl. columns.

Counting all possible 6mers:

```
def get_ngram_features(sequence, subsequences):
    """Generates counts for each subsequence.
```

Args:

data (DataFrame): The data you want to create features from. Must include a "sequence" column.

subsequences (list): A list of subsequences to count.

Returns:

DataFrame: A DataFrame with one column for each subsequence.

.....

```
features = pd.DataFrame(index=sequence.index)
for subseq in tqdm(subsequences):
    features[subseq] = sequence.str.count(subseq).astype('float32')
return features
```

Adding all the phenotypic features and its polynoms.

```
# Polynoms of one-hot encoded features
poly = PolynomialFeatures(interaction_only=True)
```

```
train_poly = pd.DataFrame(poly.fit_transform(X_train[one_hot_columns]))
```

5. Please provide the machine specs and time you used to run your model.
 - CPU (model): Intel Core i5
 - GPU (model or N/A): nVidia GeForce GTX 1060 6 gb
 - Memory (GB): 32 gb
 - OS: Ubuntu
 - Train duration: ~ 20 min per seed (5 seeds)
 - Inference duration: ~ 1 minute per seed
6. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

Translation into proteins (features based on proteins), statistics for tokens(BPE) instead of k-mers, Tree-based models(random forest, gradient boosting), RNN and LSTM models.
Using tf-idf instead of counters.
7. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

no, full EDA is in the 'notebooks' folder.
8. How did you evaluate performance of the model other than the provided metric, if at all?

Average accuracy per class. It helps to score rare classes.
9. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

At least 16 gb of RAM and 6 gb for GPU. Feature preparation could be way faster if multi thread support was implemented (get_ngram_features function).
10. Do you have any useful charts, graphs, or visualizations from the process?

no
11. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

I'd try transformer models pre-trained on big amounts of DNA sequences using self-learning (predicting random subsequence).