Supplementary Document

Supplementary document adding details to each part when relevant. In general, data and other files that may be relevant can be found at https://github.com/yoann-ba/ColabFold_PepPI_files.

Table of Contents

1 Introduction

Throughout its pipeline, AF2 updates and jointly embeds a representation of the MSA of (n_similar_seqs, n_AA, n_channels) shape and a pairwise representation of the amino acids of (n_AA, n_AA, n_channels) shape.

At input these tensors are respectively the true MSA and an aggregation (depth-wise, over the similar sequences found) of some inter-amino acids features. At its last recycle step though, before the last call to the structure generation module and the output of the last 3D structure that serves as the main prediction, these two neural embeddings contain (at least) all the information necessary to generate the output structure.

2 Methods, Materials

2.1 Dataset

This dataset has the advantages of :

- having many different types of proteins while staying small, allowing for a computationally quick overview of the protein landscape, and
- having the negative pairs being computed from pairs of different families rather than purely random.

Notable issues and areas for improvements are as follows.

- Only the binary/boolean nature of the label, “pair interacts or not”, is preserved for this work as a prediction target, and no further nuance such as the AA inter-object distances.
- PepBDB defines peptides as being shorter than 50 AA, and an interaction as a complex possessing at least one inter-object AA distance of less than 5A. No other meaning of “interaction” that may be relevant to biology or chemistry is involved.
- The negative examples are generated which, while usual for this task, remains unsatisfactory for representing real-world phenomena.
- There is only one complex per ECOD cluster, meaning this work can not speak as to the intra-cluster variation of the performance of the method or of its confidence measure.
- The last update date of PepBDB is Mars 2020, meaning parts of its dataset may be present in the training of AF-Mv2 (stops at April 2018) and AF-Mv3 (stops at September 2021). Even though only a small portion of PepBDB ends up being used in this work, and the task evaluated is not the structure prediction but a downstream task, this is still not ideal. (see https://github.com/deepmind/alphafold/releases)

2.2 ColabFold Run + Feature extraction

ColabFold run parameters :

- no templates or amber relaxation,
- MSAs computed by MMseqs2 on the UniRef + Environmental database in “unpaired” mode,
- AF-M v3,
- all 5 AF-M models,
- only 2 RNG seeds per model (for MSA subsampling and for inference dropout),
- maximum 5 recycles with a recycle tolerance of 0.5,
- early stopping threshold of 0.9 tracking the multimer score (0.8*iptm + 0.2* ptm),
- a 10% re-compile padding,
- no cluster profiles,
- sequences read from Google Drive, script running in Google Colab, data saved back to Google Drive (nothing local).

The ColabFold personal branch used is https://github.com/yoann-ba/ColabFold_light.

For the confidence scores, for each rank :

- The PAE is not saved directly in this work in all its dimensions for lightness, just the max PAE saved by default alongside the other confidence scores.
- The pTM, ipTM are saved, alongside the “multimer score” (0.8*ipTM + 0.2*pTM).
- The plDDT is saved for the entire complex sequence, and from it is extracted the max, median, mean, std of the whole plDDT list. The peptide part of the plDDT is also separated as a sub-list I refer to as pep_plddt, and summarized by the 4 same operations.

These values are further aggregated over the 1 to 10 ranks computed for each complex, by each of the 4 options of taking the first rank, the mean, the median or the standard deviation. These represent the intuitions of taking the best option, a middle value, or watching the variation over runs. The number of iterations that the system ran for before early stopping is also added as one more feature, or the maximum amount of 10 if it did not stop.

For the distance features, the 3D distogram is used to extract a few 2D maps at runtime, and these are summarized into 1D features during computation. The distogram is of shape (n_AA, n_AA, 32) and represents a binned probability distribution of the distance between each pair of amino acids. It is a prediction separate to the output structure and the 2D distance map extracted from it is not a distance map constructed from the final structure.

For this work, the 3D distogram is split into the minimum distance 2D map (coming from an argmax depth-wise over the whole distogram), and the probability maps. The default method to compute the probability maps is to apply a softmax depth-wise on the distogram to interpret the value of each distance bin as a probability of the AA distance related being in that bin, and as such the typical 2D probability map is computed by summing up the probabilities from the first bin to the bin that corresponds to 8Å, matching the CASP-set rule that two AAs are in contact if their distance is smaller than this threshold.

This work tests out the optimality of this threshold by saving not only the classic 2D probability map that ends up being a sum from bin 0 to bin 18, but also the same map if the threshold had been set to land in bin 16, 17, 19, 20 or 21 (approximately from 7.3Å to 8.9Å).

For each of the 2D maps (minimum distance, probability map@16, …, probability map@21), the decision is made to only look at one inter-object rectangular section, as both the peptide AA x peptide AA square and the protein AA x protein AA square may have the highest certainty of contact, especially in the main diagonal of the chain contacts. The two peptide x protein rectangular sections are assumed to be symmetrical enough, and only one is observed.

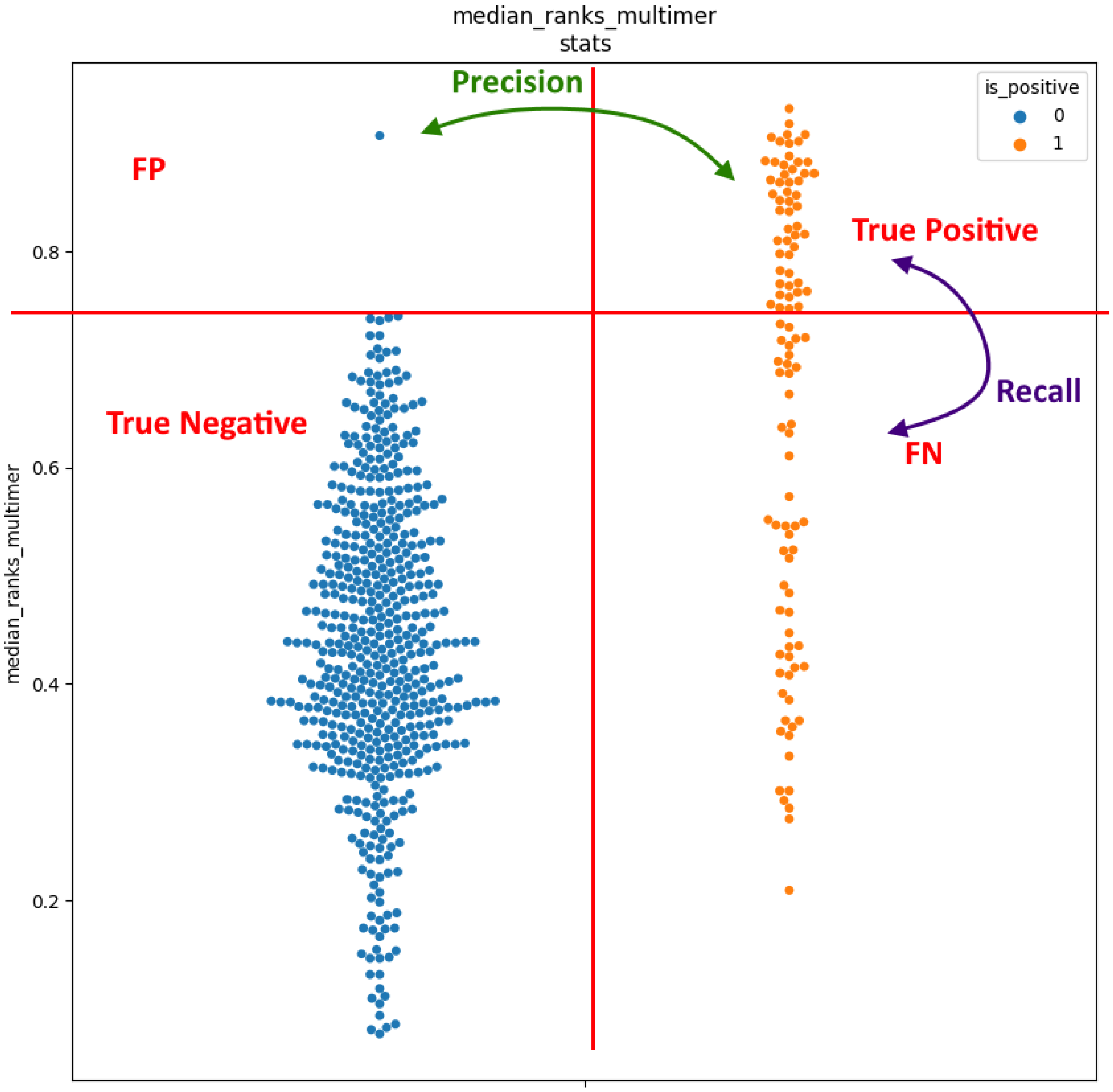
As such, for each of the predicted model, the information extracted is :

- the minimum, median, mean and standard deviation of the distance map,
- the maximum, median, mean, standard deviation of each probability map@slice for slice number 16 to 21.

As these are values summarized per predicted model, they are further aggregated with the same 4 options as the confidence scores.

2.3 F-Beta based threshold calibration

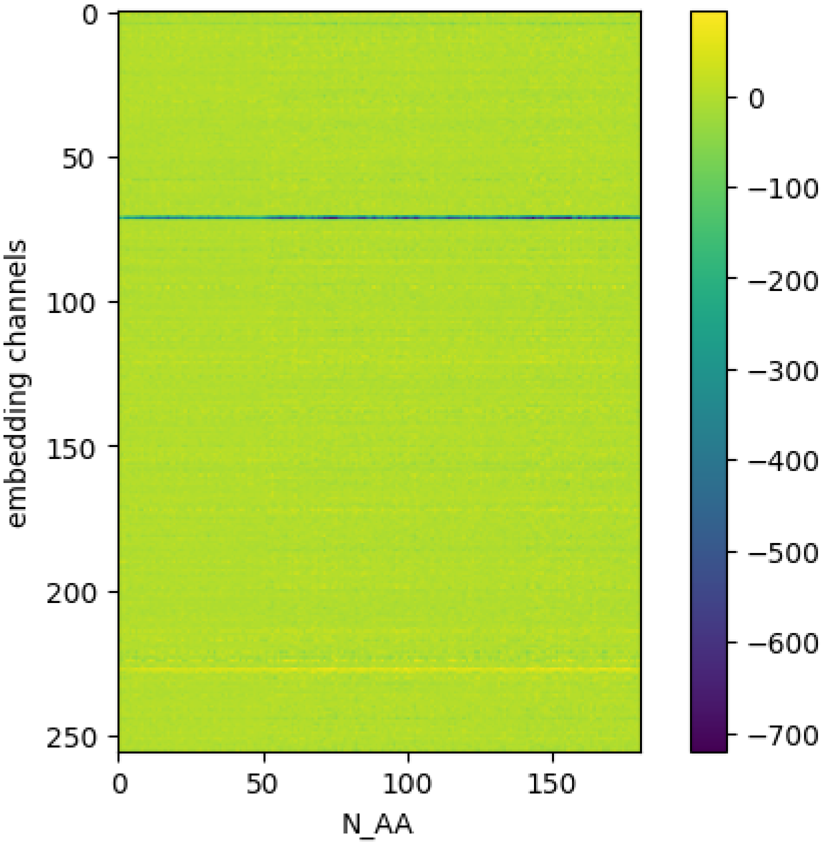
One example of such a threshold, on the feature where the median over all the multimer scores is taken :



Scatterplot of the median_ranks_multimer feature

In this figure, the horizontal red line represents where the optimal F-Beta threshold would land if we used the entire dataset for the calibration. Above this threshold is where all points would be considered positive, leading to the indicated zones for computing performance.

2.7 MSA embeddings



Example of the saved "single representation" for one complex

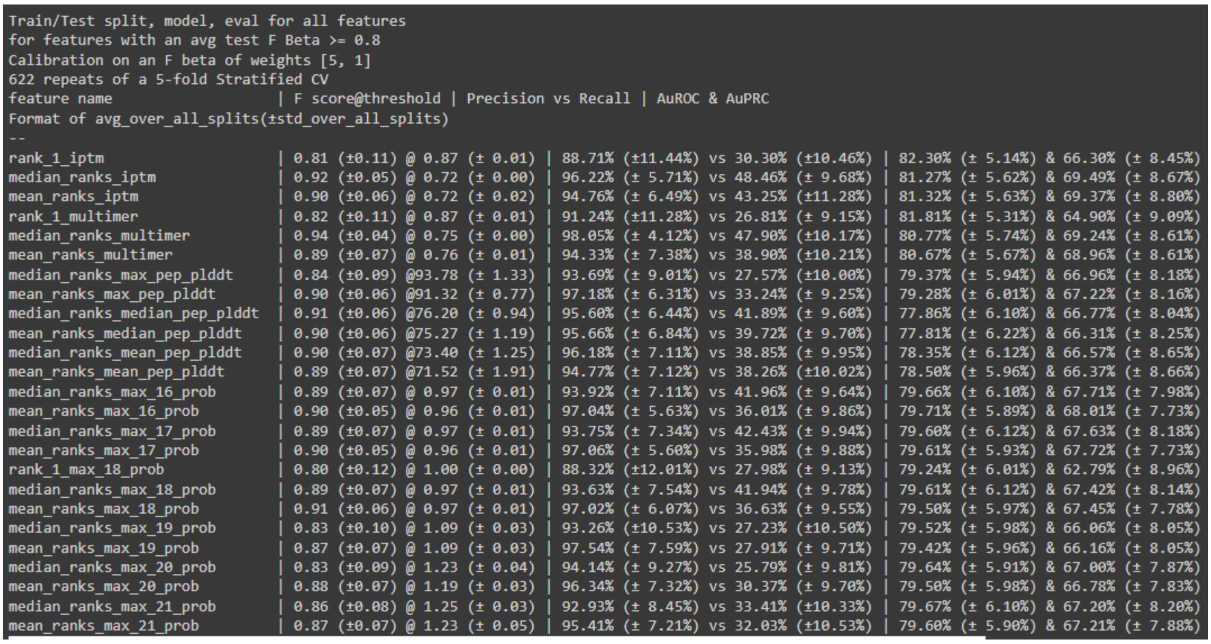
The darker and lighter rows in the image, seen at channel 71 and 227 respectively, are what I would assume to be some sort of artifact that I cannot explain, although the pattern holds remarkably well across the entire dataset with a quick check (the two channels are always respectively of a much lower and slightly higher average value that their surroundings).

Possible improvements could be :

- to also use the Pairwise representation, though they are much heavier
- to use either or both representation directly as input tensors of a large model

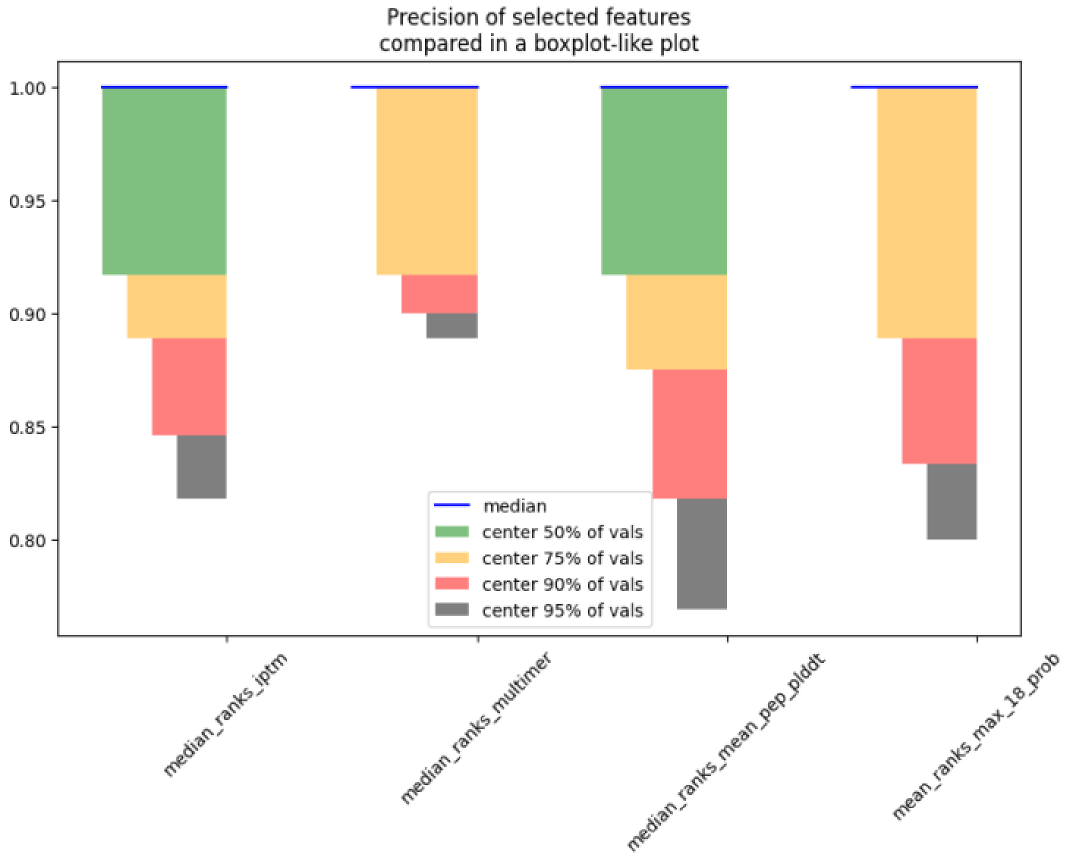
3 Results

3.1 Single feature classification

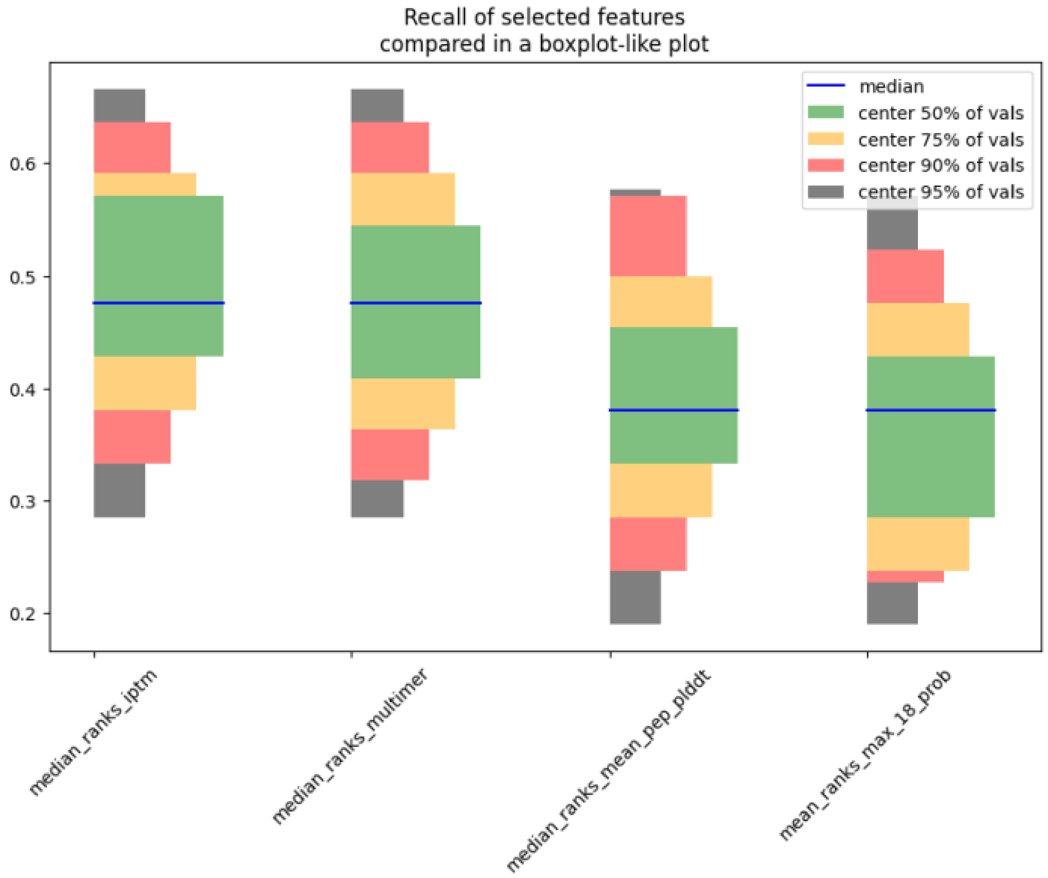


```
Train/Test split, model, eval for all features
for features with an avg test F Beta >= 0.8
Calibration on an F beta of weights [5, 1]
622 repeats of a 5-fold Stratified CV
feature name                   | F score@threshold | Precision vs Recall | AuROC & AuPRC
Format of avg_over_all_splits(±std_over_all_splits)
--
rank_1_iptm                    | 0.81 (±0.11) @ 0.87 (± 0.01) | 88.71% (±11.44%) vs 30.30% (±10.46%) | 82.30% (± 5.14%) & 66.30% (± 8.45%)
median_ranks_iptm              | 0.92 (±0.05) @ 0.72 (± 0.00) | 96.22% (± 5.71%) vs 48.46% (± 9.68%) | 81.27% (± 5.62%) & 69.49% (± 8.67%)
mean_ranks_iptm                | 0.90 (±0.06) @ 0.72 (± 0.02) | 94.76% (± 6.49%) vs 43.25% (±11.28%) | 81.32% (± 5.63%) & 69.37% (± 8.80%)
rank_1_multimer                | 0.82 (±0.11) @ 0.87 (± 0.01) | 91.24% (±11.28%) vs 26.81% (± 9.15%) | 81.81% (± 5.31%) & 64.90% (± 9.09%)
median_ranks_multimer          | 0.94 (±0.04) @ 0.75 (± 0.00) | 98.05% (± 4.12%) vs 47.90% (±10.17%) | 80.77% (± 5.74%) & 69.24% (± 8.61%)
mean_ranks_multimer            | 0.89 (±0.07) @ 0.76 (± 0.01) | 94.33% (± 7.38%) vs 38.90% (±10.21%) | 80.67% (± 5.67%) & 68.96% (± 8.61%)
median_ranks_max_pep_plddt     | 0.84 (±0.09) @93.78 (± 1.33) | 93.69% (± 9.01%) vs 27.57% (±10.00%) | 79.37% (± 5.94%) & 66.96% (± 8.18%)
mean_ranks_max_pep_plddt       | 0.90 (±0.06) @91.32 (± 0.77) | 97.18% (± 6.31%) vs 33.24% (± 9.25%) | 79.28% (± 6.01%) & 67.22% (± 8.16%)
median_ranks_median_pep_plddt  | 0.91 (±0.06) @76.20 (± 0.94) | 95.60% (± 6.44%) vs 41.89% (± 9.60%) | 77.86% (± 6.10%) & 66.77% (± 8.04%)
mean_ranks_median_pep_plddt    | 0.90 (±0.06) @75.27 (± 1.19) | 95.66% (± 6.84%) vs 39.72% (± 9.70%) | 77.81% (± 6.22%) & 66.31% (± 8.25%)
median_ranks_mean_pep_plddt    | 0.90 (±0.07) @73.40 (± 1.25) | 96.18% (± 7.11%) vs 38.85% (± 9.95%) | 78.35% (± 6.12%) & 66.57% (± 8.65%)
mean_ranks_mean_pep_plddt      | 0.89 (±0.07) @71.52 (± 1.91) | 94.77% (± 7.12%) vs 38.26% (±10.02%) | 78.50% (± 5.96%) & 66.37% (± 8.66%)
median_ranks_max_16_prob       | 0.89 (±0.07) @ 0.97 (± 0.01) | 93.92% (± 7.11%) vs 41.96% (± 9.64%) | 79.66% (± 6.10%) & 67.71% (± 7.98%)
mean_ranks_max_16_prob         | 0.90 (±0.05) @ 0.96 (± 0.01) | 97.04% (± 5.63%) vs 36.01% (± 9.86%) | 79.71% (± 5.89%) & 68.01% (± 7.73%)
median_ranks_max_17_prob       | 0.89 (±0.07) @ 0.97 (± 0.01) | 93.75% (± 7.34%) vs 42.43% (± 9.94%) | 79.60% (± 6.12%) & 67.63% (± 8.18%)
mean_ranks_max_17_prob         | 0.90 (±0.05) @ 0.96 (± 0.01) | 97.06% (± 5.60%) vs 35.98% (± 9.88%) | 79.61% (± 5.93%) & 67.72% (± 7.73%)
rank_1_max_18_prob             | 0.80 (±0.12) @ 1.00 (± 0.00) | 88.32% (±12.01%) vs 27.98% (± 9.13%) | 79.24% (± 6.01%) & 62.79% (± 8.96%)
median_ranks_max_18_prob       | 0.89 (±0.07) @ 0.97 (± 0.01) | 93.63% (± 7.54%) vs 41.94% (± 9.78%) | 79.61% (± 6.12%) & 67.42% (± 8.14%)
mean_ranks_max_18_prob         | 0.91 (±0.06) @ 0.97 (± 0.01) | 97.02% (± 6.07%) vs 36.63% (± 9.55%) | 79.50% (± 5.97%) & 67.45% (± 7.78%)
median_ranks_max_19_prob       | 0.83 (±0.10) @ 1.09 (± 0.03) | 93.26% (±10.53%) vs 27.23% (±10.50%) | 79.52% (± 5.98%) & 66.06% (± 8.05%)
mean_ranks_max_19_prob         | 0.87 (±0.07) @ 1.09 (± 0.03) | 97.54% (± 7.59%) vs 27.91% (± 9.71%) | 79.42% (± 5.96%) & 66.16% (± 8.05%)
median_ranks_max_20_prob       | 0.83 (±0.09) @ 1.23 (± 0.04) | 94.14% (± 9.27%) vs 25.79% (± 9.81%) | 79.64% (± 5.91%) & 67.00% (± 7.87%)
mean_ranks_max_20_prob         | 0.88 (±0.07) @ 1.19 (± 0.03) | 96.34% (± 7.32%) vs 30.37% (± 9.70%) | 79.50% (± 5.98%) & 66.78% (± 7.83%)
median_ranks_max_21_prob       | 0.86 (±0.08) @ 1.25 (± 0.03) | 92.93% (± 8.45%) vs 33.41% (±10.33%) | 79.67% (± 6.10%) & 67.20% (± 8.20%)
mean_ranks_max_21_prob         | 0.87 (±0.07) @ 1.23 (± 0.05) | 95.41% (± 7.21%) vs 32.03% (±10.53%) | 79.60% (± 5.90%) & 67.21% (± 7.88%)
```

To get more information on the distribution of the most interesting features, their various percentile ranges can be compared :



Precision distribution plot of the most interesting single features

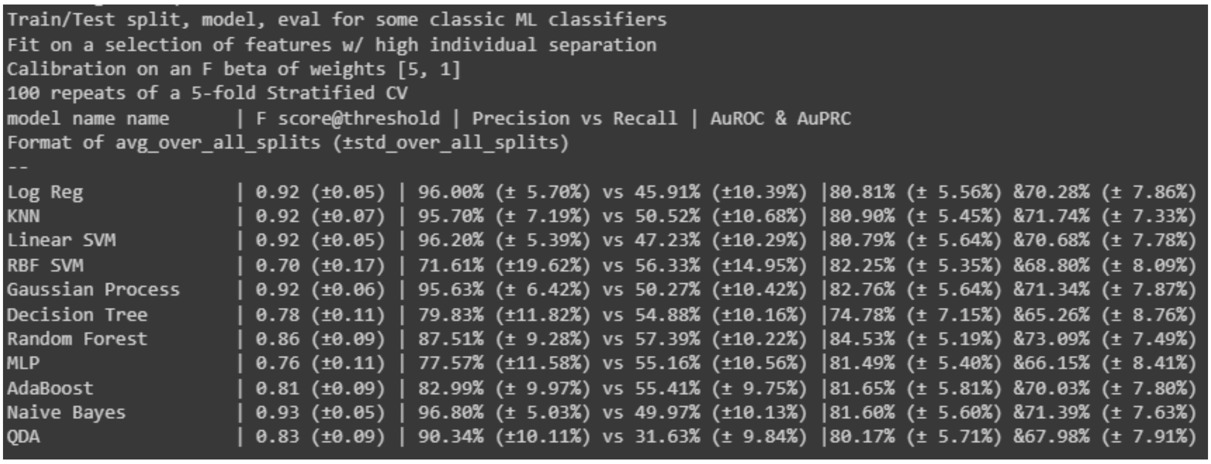


Recall distribution plot of the most interesting single features

Some additional information can be obtained :

- In 50% of the 622x5 runs, median_ranks_multimer is at 100% Precision. In 95% of them, it is still above 85%.
- The variation over its recall is also acceptable, going down to 30% Recall.

3.2 Basic ML classifiers



```
Train/Test split, model, eval for some classic ML classifiers
Fit on a selection of features w/ high individual separation
Calibration on an F beta of weights [5, 1]
100 repeats of a 5-fold Stratified CV
model name name      | F score@threshold | Precision vs Recall | AuROC & AuPRC
Format of avg_over_all_splits (±std_over_all_splits)
--
Log Reg              | 0.92 (±0.05) | 96.00% (± 5.70%) vs 45.91% (±10.39%) |80.81% (± 5.56%) &70.28% (± 7.86%)
KNN                  | 0.92 (±0.07) | 95.70% (± 7.19%) vs 50.52% (±10.68%) |80.90% (± 5.45%) &71.74% (± 7.33%)
Linear SVM           | 0.92 (±0.05) | 96.20% (± 5.39%) vs 47.23% (±10.29%) |80.79% (± 5.64%) &70.68% (± 7.78%)
RBF SVM              | 0.70 (±0.17) | 71.61% (±19.62%) vs 56.33% (±14.95%) |82.25% (± 5.35%) &68.80% (± 8.09%)
Gaussian Process     | 0.92 (±0.06) | 95.63% (± 6.42%) vs 50.27% (±10.42%) |82.76% (± 5.64%) &71.34% (± 7.87%)
Decision Tree        | 0.78 (±0.11) | 79.83% (±11.82%) vs 54.88% (±10.16%) |74.78% (± 7.15%) &65.26% (± 8.76%)
Random Forest        | 0.86 (±0.09) | 87.51% (± 9.28%) vs 57.39% (±10.22%) |84.53% (± 5.19%) &73.09% (± 7.49%)
MLP                  | 0.76 (±0.11) | 77.57% (±11.58%) vs 55.16% (±10.56%) |81.49% (± 5.40%) &66.15% (± 8.41%)
AdaBoost             | 0.81 (±0.09) | 82.99% (± 9.97%) vs 55.41% (± 9.75%) |81.65% (± 5.81%) &70.03% (± 7.80%)
Naive Bayes          | 0.93 (±0.05) | 96.80% (± 5.03%) vs 49.97% (±10.13%) |81.60% (± 5.60%) &71.39% (± 7.63%)
QDA                  | 0.83 (±0.09) | 90.34% (±10.11%) vs 31.63% (± 9.84%) |80.17% (± 5.71%) &67.98% (± 7.91%)
```

The variables fed into the ML models are ["median_ranks_multimer", "median_ranks_iptm", "rank_1_ptm", “mean_ranks_max_pep_plddt", "mean_ranks_median_pep_plddt", “mean_ranks_max_16_prob", "mean_ranks_max_18_prob", "mean_ranks_max_20_prob"] to try and cover multiple origins of features.

All solo features stats and plots

Plots available at
https://github.com/yoann-ba/ColabFold_PepPI_files/tree/main/solo_features_result

All MSA embeddings x PepPI label plots

Plots at
https://github.com/yoann-ba/ColabFold_PepPI_files/tree/main/msa_embeds/vs_interaction_label

All MSA embeddings x median_ranks_multimer plots

Plots at
https://github.com/yoann-ba/ColabFold_PepPI_files/tree/main/msa_embeds/vs_task_solo_feature

All MSA embeddings x success rate plots

Plots at
https://github.com/yoann-ba/ColabFold_PepPI_files/tree/main/msa_embeds/vs_success