Anirudha - approach for DS Test

Hello! Thank you for this ML challenge. I have detailed the steps taken to achieve the result in this page.

Github link: https://github.com/anirudhasundaresan/cryoocyte_challenge
Please follow Github link for more detailed analysis. It will give you an idea about the entire workflow and development of the algorithm. You can either follow the code in *pipeline.py* or follow the results in *pipeline_notebook.ipynb.* A collection of random thoughts that surfaced during the model building process is listed out in *insights_action.*

Tools used: Python 3.7.2 (sklearn, matplotlib, pandas, seaborn, numpy)

Step 0 - Split the *train.csv* into *train_anirudha_train.csv* and *train_anirudha_val.csv* for validation purposes. Have not conducted analysis on the test data, as it might corrupt the developed model.

Step 1 - Examined the train data. Exploratory data analysis done. Identified as regression problem. Mapped out correlations between the features and correlations between feature and target. Generated heat map and correlation plots for top correlated matches. Checked for outliers.

Step 2 - Scaled and normalized the data and applied PCA to reduce dimensionality.

Step 3 - Built ML pipeline to test 5 different models. KNN regressors, regularized linear regression models (lasso, ridge and elastic net), and multi layer perceptron. An exhaustive grid search for their hyperparameters was conducted. Training and validation plots (cross-validation on the *train_anirudha_train.csv*) have been logged as well, to detect overfitting, if any.

- KNN regressor work okay, but when data is highly dimensional, it becomes computationally expensive and hence, harder to train.
- Multi-layer perceptrons are good to fit non-linear data, and they perform reasonably well.
- The best performance, I suspect, is from the regularized model. I personally prefer elastic nets, but they all perform almost equally well.
    - Since we are dealing with many features, I figured lasso might be a good approach to figure out which features are not important at all. But, since I am doing PCA beforehand, a few features corresponding to the smaller Eigen values automatically get zeroed out by lasso, which is expected.
    - Ridge regression is a great regularized model here, and it does not select features. It merely punishes weights if they get too large.
    - ElasticNet is a combination of the above two regression models. And is, my preference for this example. This does not select features per se, but

diminishes the weights for the features which the lasso model thinks are not important.

- All the models have been trained on the *train_anirudha_train.csv* and validated against the *train_anirudha_val.csv* using RMSE as the scoring metric.

Step 4 - Refit all the trained models with the best chosen hyperparameters on the entire training set provided to me - *train_.csv.*

Step 5 - The final predictions on the test set are in the *.csv* files (attached in this mail, as well as on the Github link). Please use this order below as my model preferences.

- *ElasticNet.csv*
- *Ridge.csv*
- *Lasso.csv*
- *MLP.csv*
- *KNeighborsRegressor.csv*