Phyla Challenge -2

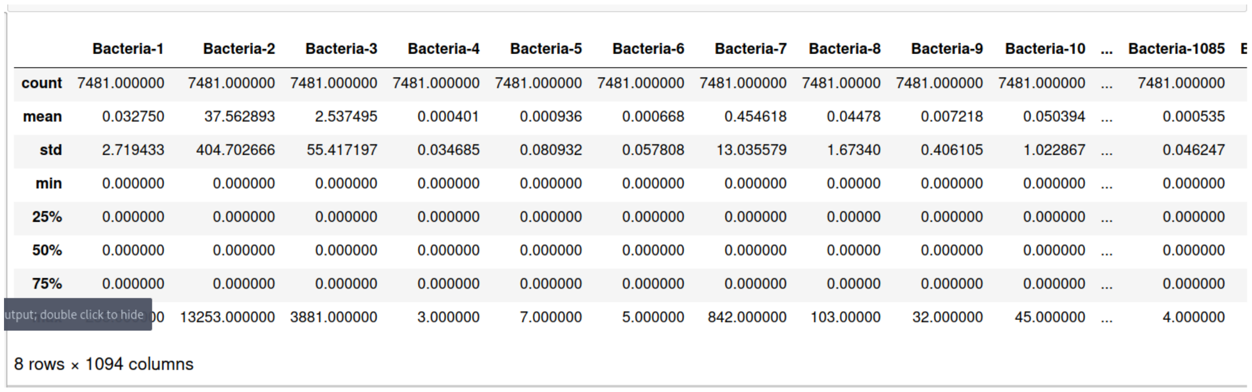
Aim: Develop a multi-label classification model to classify different diseases based on the microorganisms in the gut microbiome.

Solution Abstract:

The data was found to contain high variance in several columns and imbalance amongst the labels{disease class}. Also, there were several features{columns} that had no{almost null} influence in the final prediction. After proper EDA{Exploratory Data Analysis}, several processing and normalization steps including scaling, resampling, encoding, etc, were undertaken to make data suitable for the final algorithms. Several features that had minimal impact on the prediction were carefully carved out from the training data. A huge variety of algorithms were tested from almost all possible categories of algorithms like clustering, linear and non-linear algorithms to solve the problem and their results were noted for the final experiment. Finally, a Stacking-ensemble model containing several weak learners like LogisticRegression, RandomForestClassifier, KNeighborsClassifier, etc were used to provide predictions to the powerful meta-learner{LGBMClassifier}. The final results include a F-1 score of 0.93556 and kappa score of 0.91407 on the given dataset without overfitting.

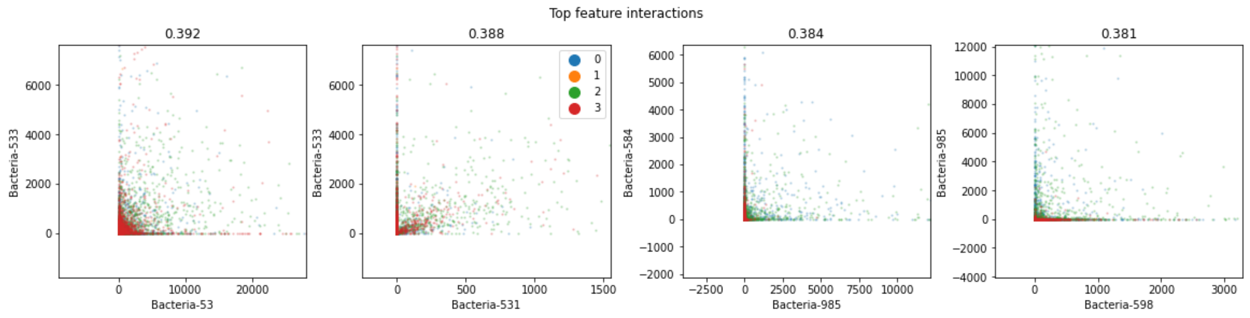
Exploratory Data Analysis:

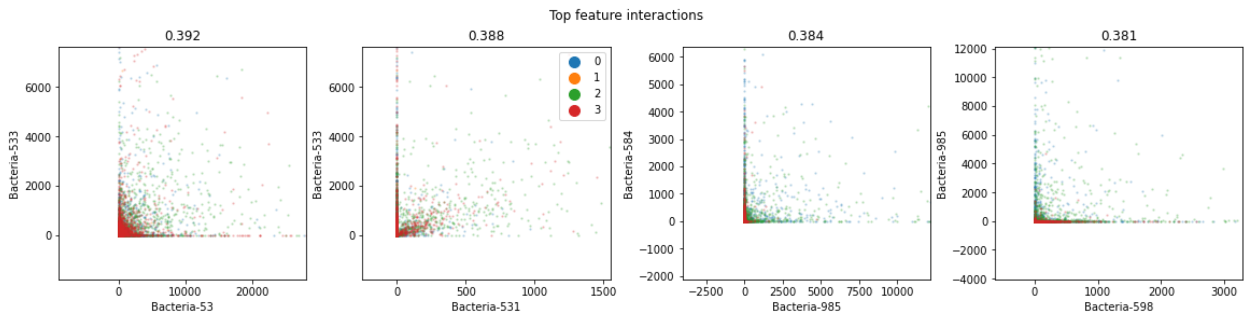
After successfully separating the numerical features of the dataset, statistical information like mean, percentiles(25,50,75), mode, maximum value, standard deviation, etc were calculated for each feature. This information helps us to find the kind of scaling and normalization required in further steps. Furthermore, the availability of nulls, NaN’s and unique values in features were found. Also the distribution of our labels on the top extracted features was plotted on a histogram to analyze any evident distinction. Finally, PCA plots were drawn to see the impact of different levels of PCA(Principal component analysis) on the variance of the dataset. The Pearson's correlation matrix and some scatter plots(incl. Pairplot) was analyzed to gain insights of the data.



	Bacteria-1	Bacteria-2	Bacteria-3	Bacteria-4	Bacteria-5	Bacteria-6	Bacteria-7	Bacteria-8	Bacteria-9	Bacteria-10	...	Bacteria-1085	E
count	7481.000000	7481.000000	7481.000000	7481.000000	7481.000000	7481.000000	7481.000000	7481.00000	7481.000000	7481.000000	...	7481.000000	
mean	0.032750	37.562893	2.537495	0.000401	0.000936	0.000668	0.454618	0.04478	0.007218	0.050394	...	0.000535	
std	2.719433	404.702666	55.417197	0.034685	0.080932	0.057808	13.035579	1.67340	0.406105	1.022867	...	0.046247	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	...	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	...	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	...	0.000000	
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	...	0.000000	
utput; double click to hide	00	13253.000000	3881.000000	3.000000	7.000000	5.000000	842.000000	103.00000	32.000000	45.000000	...	4.000000	

8 rows × 1094 columns





Data Cleaning and Processing:

1) A double check on NaN's, nulls or anomalous looking data was done and any findings were corrected. Also presence of any duplicate rows was also analyzed and removed.

2) Numerical and Categorical Features were separated.

3) **LabelEncoder** was used on the 'disease' feature to convert it into a numerical feature.

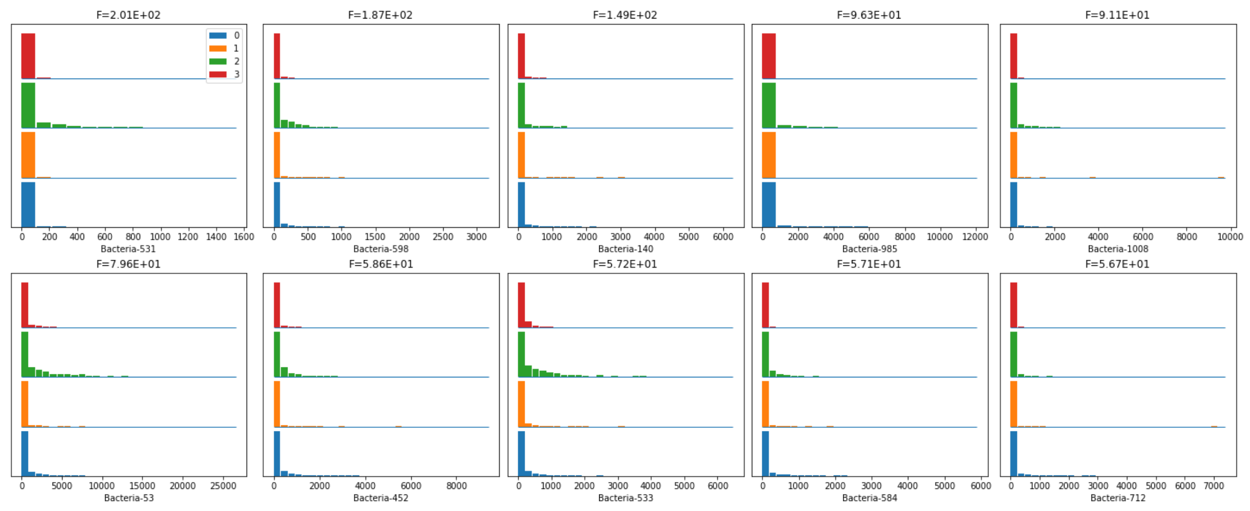
4) The range and mean of each feature was calculated and analyzed.

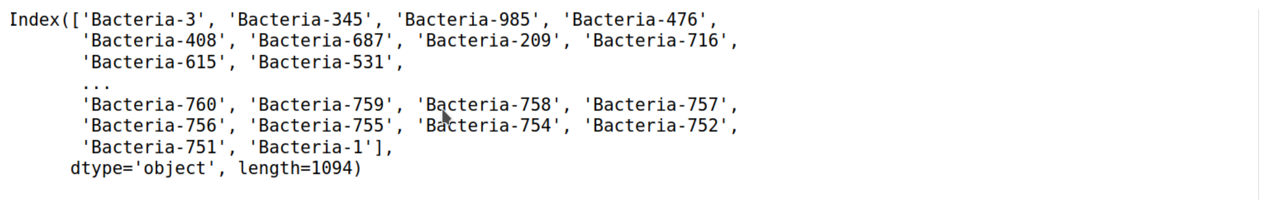
Feature Extraction:

Balanced Random Forest, XGClassifier and Pearson's correlation were used to find the most important features. Results of all these methods were carefully compared to find the most important features and their impact on the variance. {Finally XGClassifiers results were used}

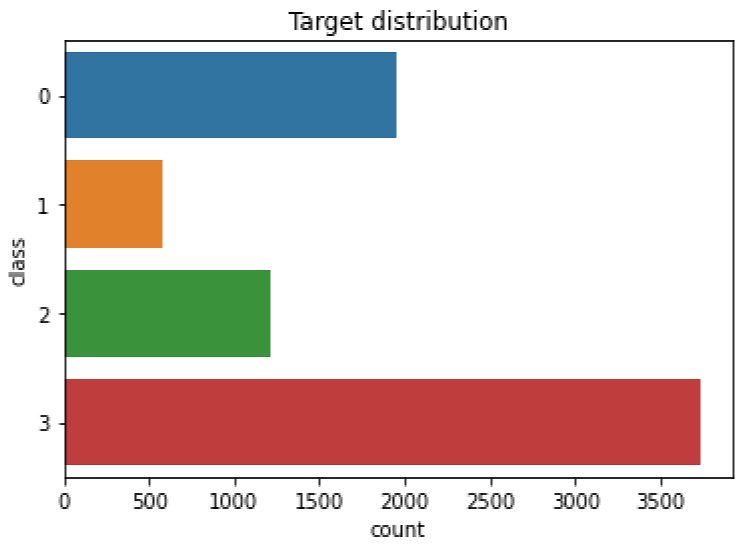
Using the top 896 features found by the XGClassifier, we were able to reduce data by over 20% while reducing variability of data by just 1.3%.

Reducing more features would have reduced variability by more than 1.5%, thus even hampering our useful data.




```
Index(['Bacteria-3', 'Bacteria-345', 'Bacteria-985', 'Bacteria-476',
       'Bacteria-408', 'Bacteria-687', 'Bacteria-209', 'Bacteria-716',
       'Bacteria-615', 'Bacteria-531',
       ...
       'Bacteria-760', 'Bacteria-759', 'Bacteria-758', 'Bacteria-757',
       'Bacteria-756', 'Bacteria-755', 'Bacteria-754', 'Bacteria-752',
       'Bacteria-751', 'Bacteria-1'],
      dtype='object', length=1094)
```

Feature Sampling and Scaling:



The dataset was imbalanced and had inequalities in the count of each label given in the dataset, thus resampling and balancing of the dataset was necessary. Several features were found to have a huge range, thus using scalers like Min-Max scaler would have dampened several of our important readings(feature value’s). Also, it is highly advisable to use RobustScaler in high range feature data as it works on 25 and 75 percentile points rather than maximum or minimum readings.

For feature resampling, we used SMOTE-Tomek Links, which is an mixture of oversampling and undersampling techniques, which firstly oversamples/generate new synthetic samples to balance dataset and then automatically treats the anomalous and un-trustworthy synthetically generated points by removing them. This technique helps us to reduce the amount of False-Positives(Type-II error) that usually increases on usage of oversampling.

Finally, the training and testing set were split in a 70:30 ratio, wherein the 30% test set would be used to find final metrics. All the scaling procedures were fit just on the training set, thus taking proper care that no information leakage happens during the training. {a situation where training set learns some information from test set}

Final Prediction Model:

The implemented solution is an ensemble of several algorithms, where several weak learners are used to form the basis for the prediction of the meta{powerful} learner. The weak learners used were: Random Forest, Logistic Regression, XGboost and the K Nearest Neighbours. The power or meta learner used in the stacking algorithm was the LGBMClassifier, which is a gradient boosting non-linear classifier.

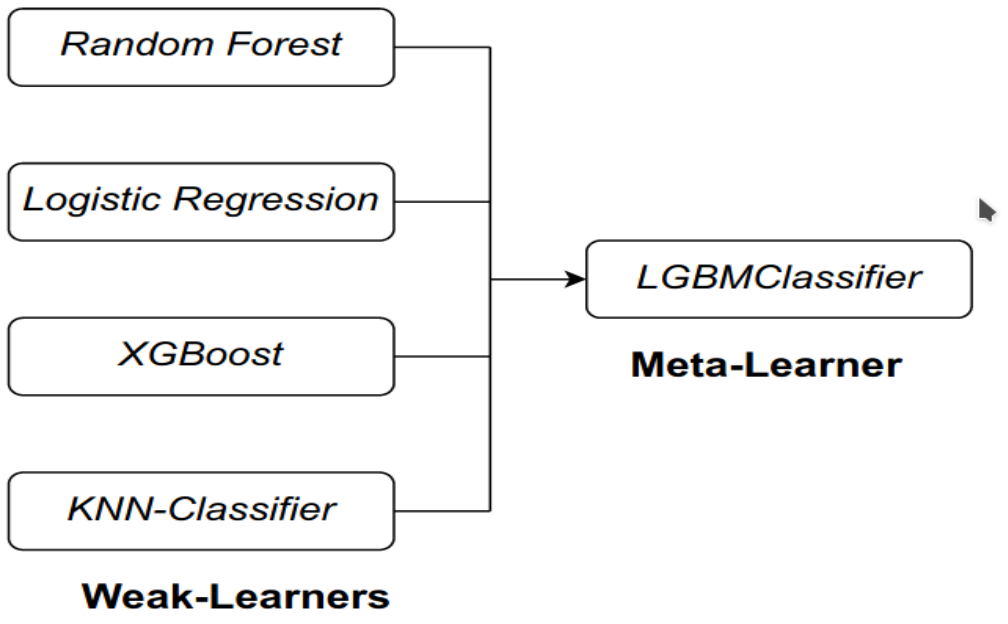
=> All the weak learners were firstly individually trained on the dataset to find the best hyper-parameters for them.

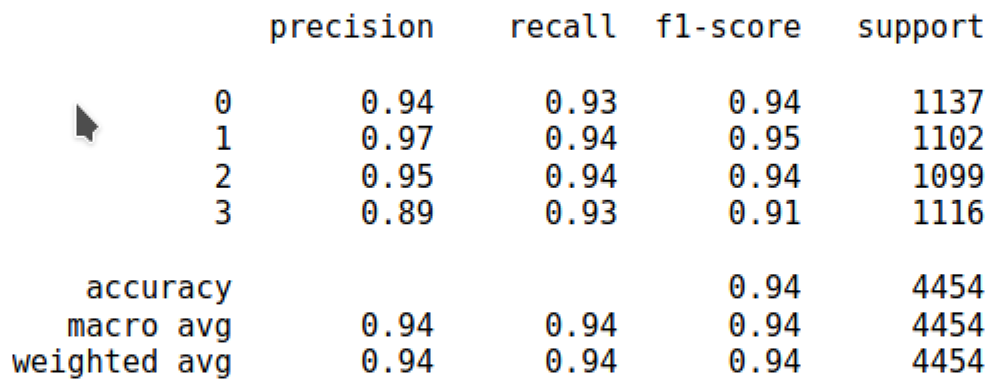
=> Stacking several weak learners helps us to gain their learnings while neglecting their drawbacks.

=> Several combinations of weak learners and meta-learners were tried before finalizing our model.

=> Overfitting was checked at each step of training

Algorithm	F-1	Kappa Score
Stacked-Ensemble	0.93556	0.91407





```
              precision    recall  f1-score   support

           0       0.94      0.93      0.94      1137
           1       0.97      0.94      0.95      1102
           2       0.95      0.94      0.94      1099
           3       0.89      0.93      0.91      1116

    accuracy                           0.94      4454
   macro avg       0.94      0.94      0.94      4454
weighted avg       0.94      0.94      0.94      4454
```

Precision-Recall-F1 score Table

Other Approaches Tried:

1) **One-Vs-Rest:** This is the second best performing technique testing during our analysis and testing phase, wherein, we split a multi-class classification problem into several binary-classification problems. {reducing complexity for the algorithm}

2) **Bagging Classifier:** Bagging is another ensemble technique, wherein several meta-estimator algorithms are trained on the dataset and a cumulative prediction is formulated through mutual voting.{Soft or hard voting}

3) **Deep Learning:** A small neural network with just 2 layers was trained using the dataset by keeping the activation function of the last neuron as softmax. This technique failed miserably as the data wasn't sufficient for the deep learning algorithms.

4) **Different Gradient Boosting Algorithms**: Some state-of-the-art gradient boosting algorithms like AdaBoost were also tried on the cleaned data.