ANALYZING VACCINATION COVERAGE TRENDS DURING THE 2009 H1N1 PANDEMIC

PHASE 2

Submitted to: CSE 587: Data Intensive Computing

Fall'2023

Manali Ramchandani, MS
Syed Razauddin Shahlal, MS
Tajammul Shuja Sayyad, MS

University at Buffalo
School of Engineering and Applied Sciences

Deliverables

1. **Algorithms/Visualizations [25 marks]:**

   Algorithms/Visualizations [25 marks]: Apply 6 different significant and relevant algorithms (ML, MR, and/or statistical models) to your data and create visualizations for the results. For 487 students: at least 1 of the 6 algorithms must be one that was not discussed in class. For 587 students: at least 2 must be from outside of class. Algorithms discussed in class are: Linear Regression, k-Means, k-NN, Naive Bayes, and Logistic Regression. The outside algorithms can come from class textbooks, or other sources. Cite the appropriate sources for each outside algorithm you choose to apply.

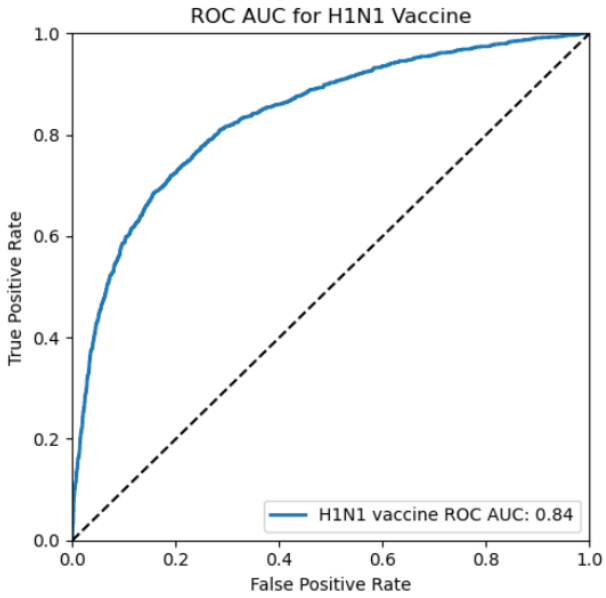
   7 different algorithms applied that is relevant to our data are below:

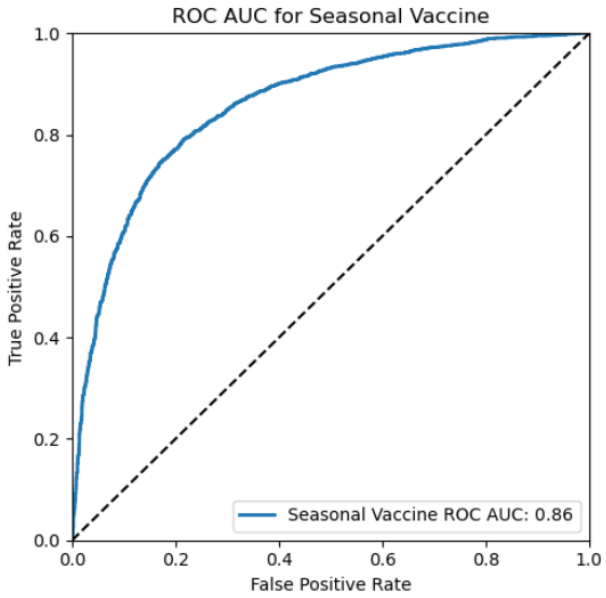
1. Logistic Regression:

```
LogisticRegression(max_iter=1000)
_____________________________________
ROC AUC for H1N1 vaccine: 0.8367361938312594
ROC AUC for Seasonal vaccine: 0.8590042728548255
Mean ROC AUC: 0.8478702333430425
Classification report for H1N1 vaccine:
              precision    recall  f1-score   support

           0       0.86      0.95      0.91      4220
           1       0.71      0.43      0.54      1122

    accuracy                           0.84      5342
   macro avg       0.79      0.69      0.72      5342
weighted avg       0.83      0.84      0.83      5342
```

Classification report for Seasonal vaccine:

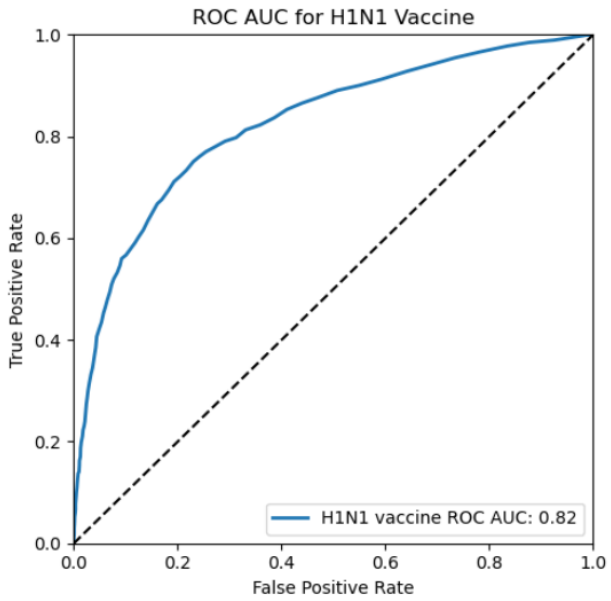
	precision	recall	f1-score	support
0	0.80	0.82	0.81	2917
1	0.78	0.75	0.76	2425
accuracy			0.79	5342
macro avg	0.79	0.79	0.79	5342
weighted avg	0.79	0.79	0.79	5342

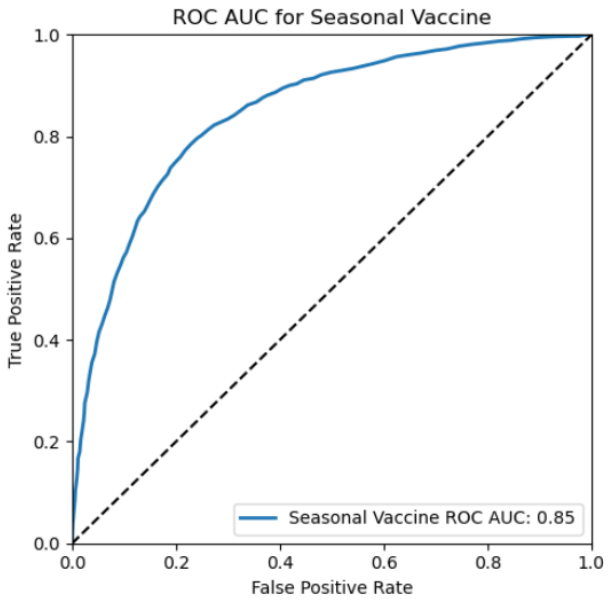




2. Random Forest Classifier:

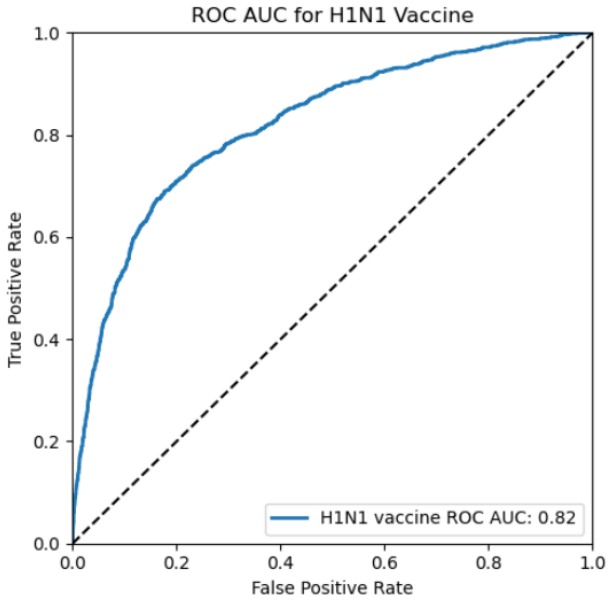
```
RandomForestClassifier()
_________________________________________
ROC AUC for H1N1 vaccine: 0.8241823377347492
ROC AUC for Seasonal vaccine: 0.8476442044326009
Mean ROC AUC: 0.8359132710836751
Classification report for H1N1 vaccine:
              precision    recall  f1-score   support

           0       0.85      0.96      0.90      4220
           1       0.71      0.37      0.49      1122

    accuracy                           0.84      5342
   macro avg       0.78      0.67      0.69      5342
weighted avg       0.82      0.84      0.82      5342

Classification report for Seasonal vaccine:
              precision    recall  f1-score   support

           0       0.79      0.81      0.80      2917
           1       0.76      0.74      0.75      2425

    accuracy                           0.78      5342
   macro avg       0.78      0.77      0.78      5342
weighted avg       0.78      0.78      0.78      5342
```

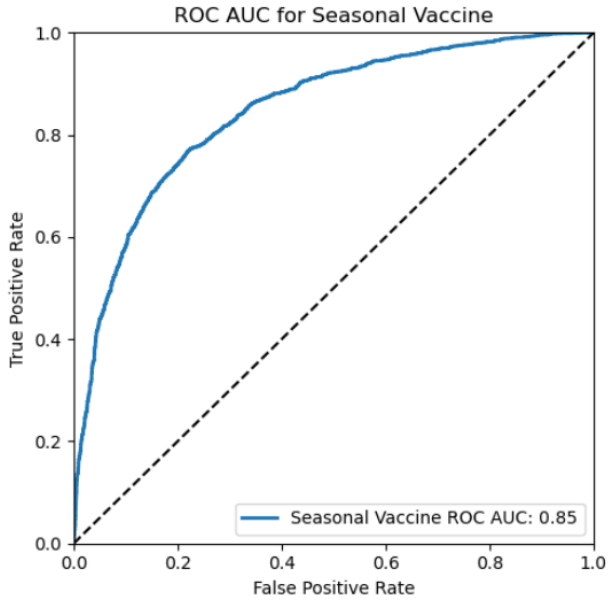




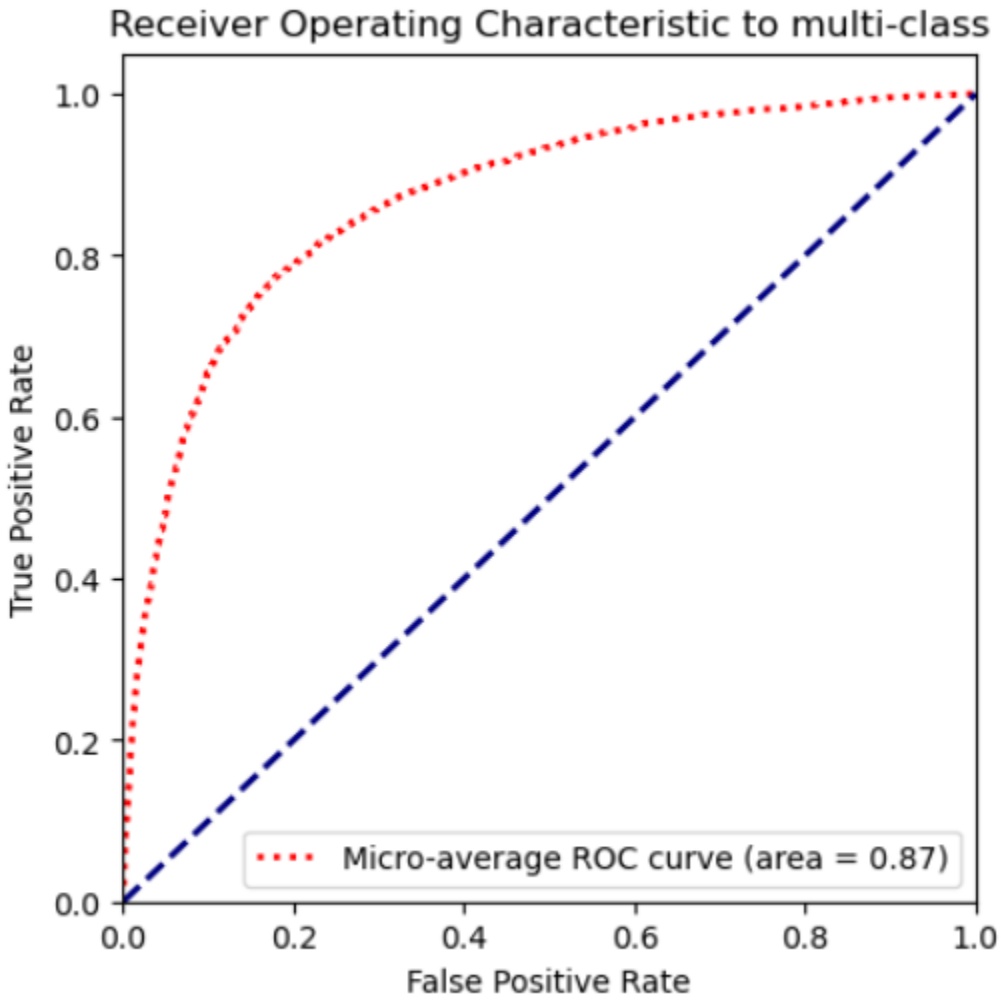
3. XGBoost Classifier:

```
_________________________________________
ROC AUC for H1N1 vaccine: 0.8201214824576967
ROC AUC for Seasonal vaccine: 0.8466074098159032
Mean ROC AUC: 0.8333644461367999
Classification report for H1N1 vaccine:
              precision    recall  f1-score   support

           0       0.86      0.94      0.90      4220
           1       0.65      0.44      0.52      1122

    accuracy                           0.83      5342
   macro avg       0.76      0.69      0.71      5342
weighted avg       0.82      0.83      0.82      5342

Classification report for Seasonal vaccine:
              precision    recall  f1-score   support

           0       0.79      0.80      0.80      2917
           1       0.76      0.74      0.75      2425

    accuracy                           0.77      5342
   macro avg       0.77      0.77      0.77      5342
weighted avg       0.77      0.77      0.77      5342
```





XGBoost Classifier with GridSearchCV:




```
Average ROC AUC score: 0.87
```

4. Neural Network Sequential model with ADAM optimizer:

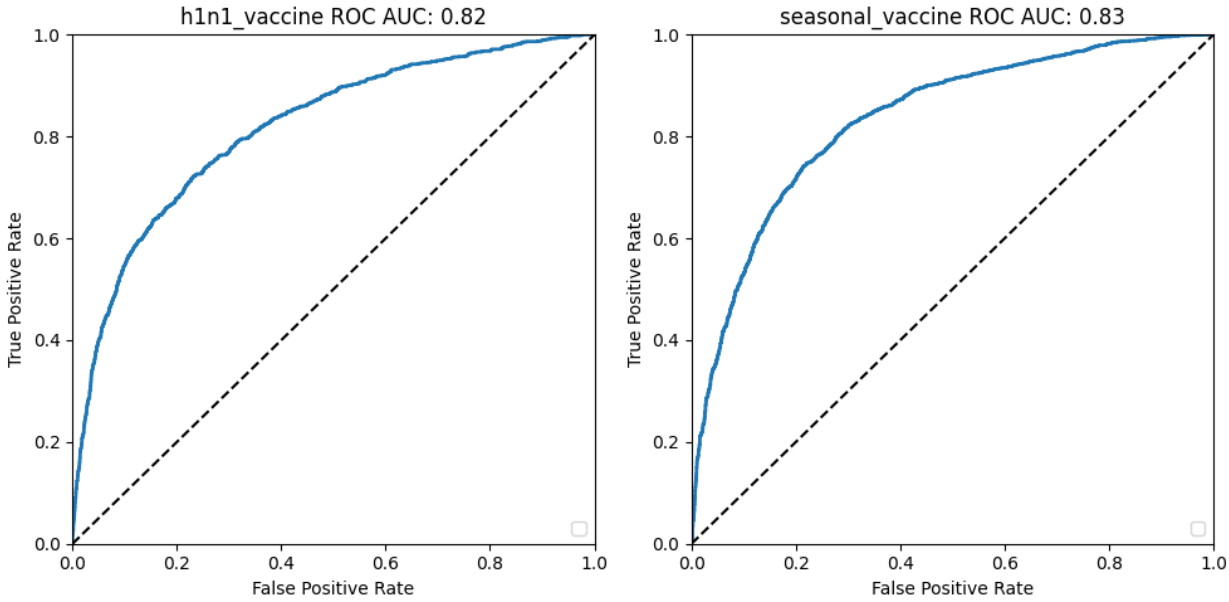
```
Epoch 1/10
535/535 [==============================] - 8s 6ms/step - loss: 0.4162 - accuracy: 0.8197 - val_loss: 0.3948 - val_accuracy: 0.8292
Epoch 2/10
535/535 [==============================] - 3s 5ms/step - loss: 0.3838 - accuracy: 0.8376 - val_loss: 0.3939 - val_accuracy: 0.8315
Epoch 3/10
535/535 [==============================] - 4s 7ms/step - loss: 0.3743 - accuracy: 0.8385 - val_loss: 0.3967 - val_accuracy: 0.8301
Epoch 4/10
535/535 [==============================] - 5s 9ms/step - loss: 0.3655 - accuracy: 0.8445 - val_loss: 0.3922 - val_accuracy: 0.8306
Epoch 5/10
535/535 [==============================] - 6s 12ms/step - loss: 0.3545 - accuracy: 0.8500 - val_loss: 0.3975 - val_accuracy: 0.8310
Epoch 6/10
535/535 [==============================] - 5s 9ms/step - loss: 0.3451 - accuracy: 0.8517 - val_loss: 0.4024 - val_accuracy: 0.8317
Epoch 7/10
535/535 [==============================] - 4s 8ms/step - loss: 0.3328 - accuracy: 0.8562 - val_loss: 0.4090 - val_accuracy: 0.8200
Epoch 8/10
535/535 [==============================] - 6s 10ms/step - loss: 0.3202 - accuracy: 0.8627 - val_loss: 0.4167 - val_accuracy: 0.8268
Epoch 9/10
535/535 [==============================] - 4s 7ms/step - loss: 0.3077 - accuracy: 0.8693 - val_loss: 0.4257 - val_accuracy: 0.8165
Epoch 10/10
535/535 [==============================] - 2s 4ms/step - loss: 0.2976 - accuracy: 0.8766 - val_loss: 0.4390 - val_accuracy: 0.8198
167/167 [==============================] - 0s 2ms/step - loss: 0.4261 - accuracy: 0.8272
Test Accuracy: 82.72%
167/167 [==============================] - 0s 2ms/step
ROC AUC for h1n1_vaccine: 81.64%
```

Classification report for h1n1_vaccine:

	precision	recall	f1-score	support
0	0.88	0.91	0.89	4220
1	0.60	0.53	0.56	1122
accuracy			0.83	5342
macro avg	0.74	0.72	0.73	5342
weighted avg	0.82	0.83	0.82	5342

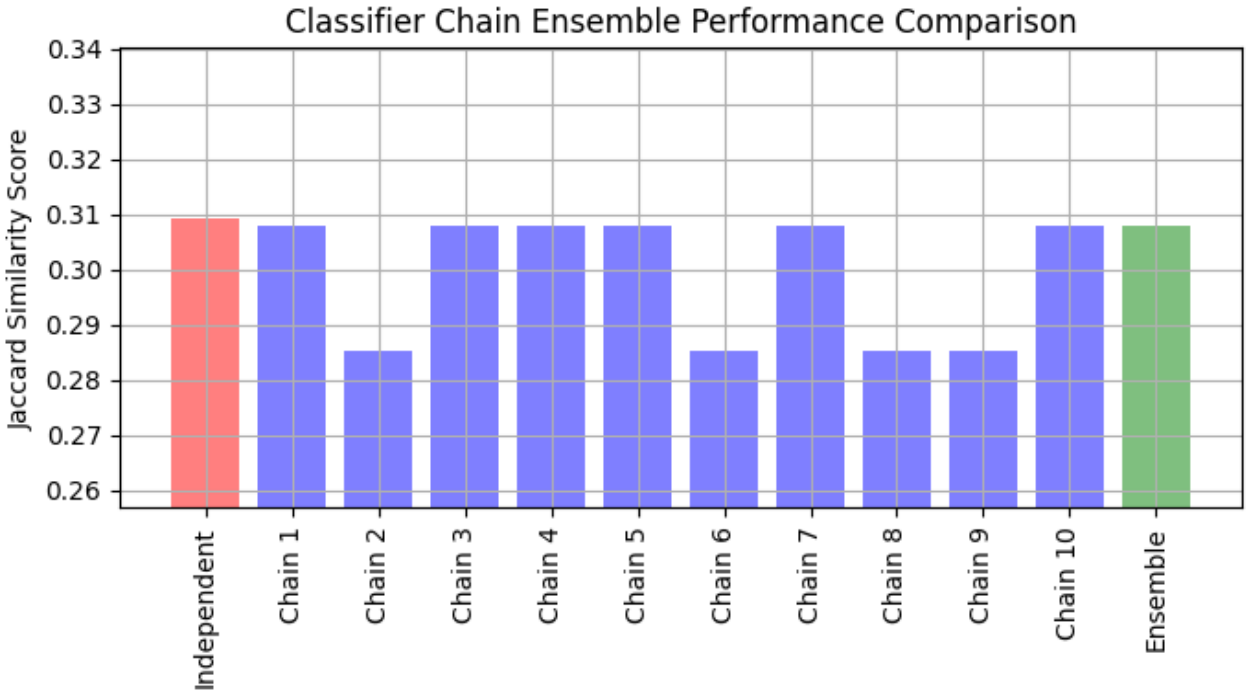
```
Epoch 1/10
535/535 [==============================] - 4s 6ms/step - loss: 0.5176 - accuracy: 0.7517 - val_loss: 0.4966 - val_accuracy: 0.7634
Epoch 2/10
535/535 [==============================] - 3s 5ms/step - loss: 0.4823 - accuracy: 0.7742 - val_loss: 0.4911 - val_accuracy: 0.7629
Epoch 3/10
535/535 [==============================] - 2s 4ms/step - loss: 0.4699 - accuracy: 0.7814 - val_loss: 0.4851 - val_accuracy: 0.7667
Epoch 4/10
535/535 [==============================] - 2s 4ms/step - loss: 0.4585 - accuracy: 0.7898 - val_loss: 0.4856 - val_accuracy: 0.7721
Epoch 5/10
535/535 [==============================] - 2s 4ms/step - loss: 0.4468 - accuracy: 0.7935 - val_loss: 0.4995 - val_accuracy: 0.7620
Epoch 6/10
535/535 [==============================] - 2s 4ms/step - loss: 0.4353 - accuracy: 0.7993 - val_loss: 0.4982 - val_accuracy: 0.7671
Epoch 7/10
535/535 [==============================] - 3s 6ms/step - loss: 0.4230 - accuracy: 0.8084 - val_loss: 0.5076 - val_accuracy: 0.7627
Epoch 8/10
535/535 [==============================] - 2s 5ms/step - loss: 0.4082 - accuracy: 0.8144 - val_loss: 0.5184 - val_accuracy: 0.7547
Epoch 9/10
535/535 [==============================] - 2s 4ms/step - loss: 0.3951 - accuracy: 0.8230 - val_loss: 0.5284 - val_accuracy: 0.7571
Epoch 10/10
535/535 [==============================] - 2s 4ms/step - loss: 0.3825 - accuracy: 0.8300 - val_loss: 0.5361 - val_accuracy: 0.7536
167/167 [==============================] - 1s 4ms/step - loss: 0.5233 - accuracy: 0.7649
Test Accuracy: 76.49%
167/167 [==============================] - 1s 3ms/step
ROC AUC for seasonal_vaccine: 83.28%
```

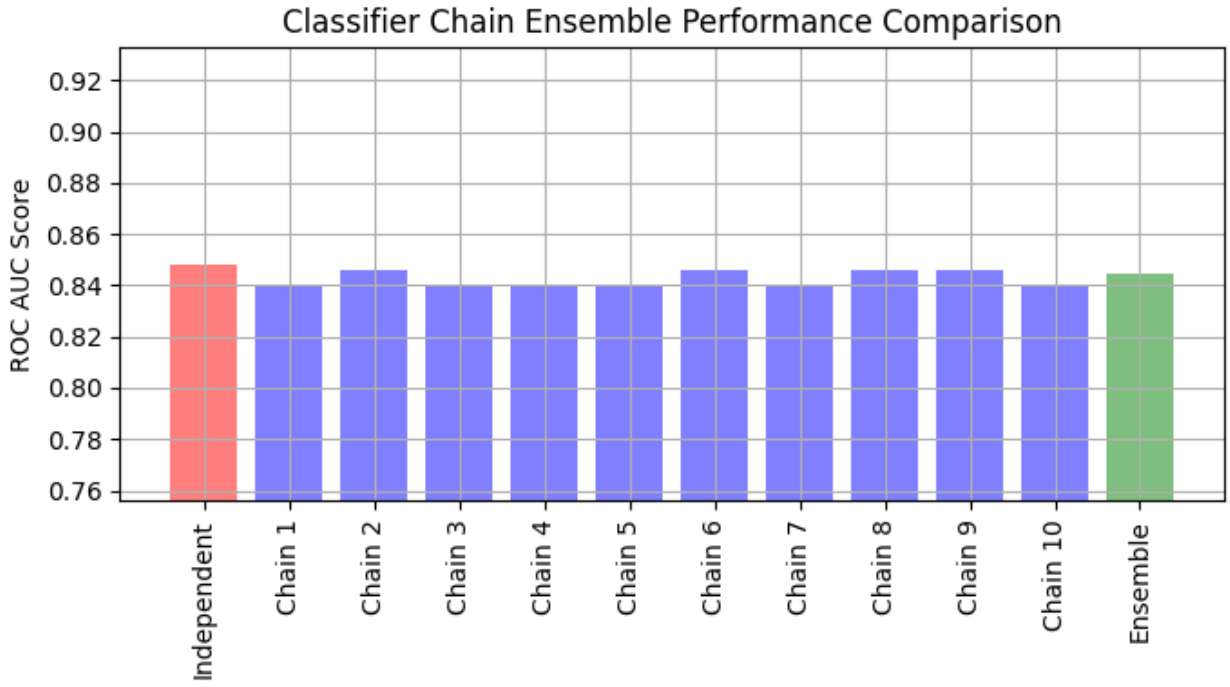
```
Classification report for seasonal_vaccine:
WARNING:matplotlib.legend:No artists with labels found
WARNING:matplotlib.legend:No artists with labels found
              precision    recall  f1-score   support

           0       0.78      0.79      0.79      2917
           1       0.74      0.74      0.74      2425

    accuracy                           0.76      5342
   macro avg       0.76      0.76      0.76      5342
weighted avg       0.76      0.76      0.76      5342
```

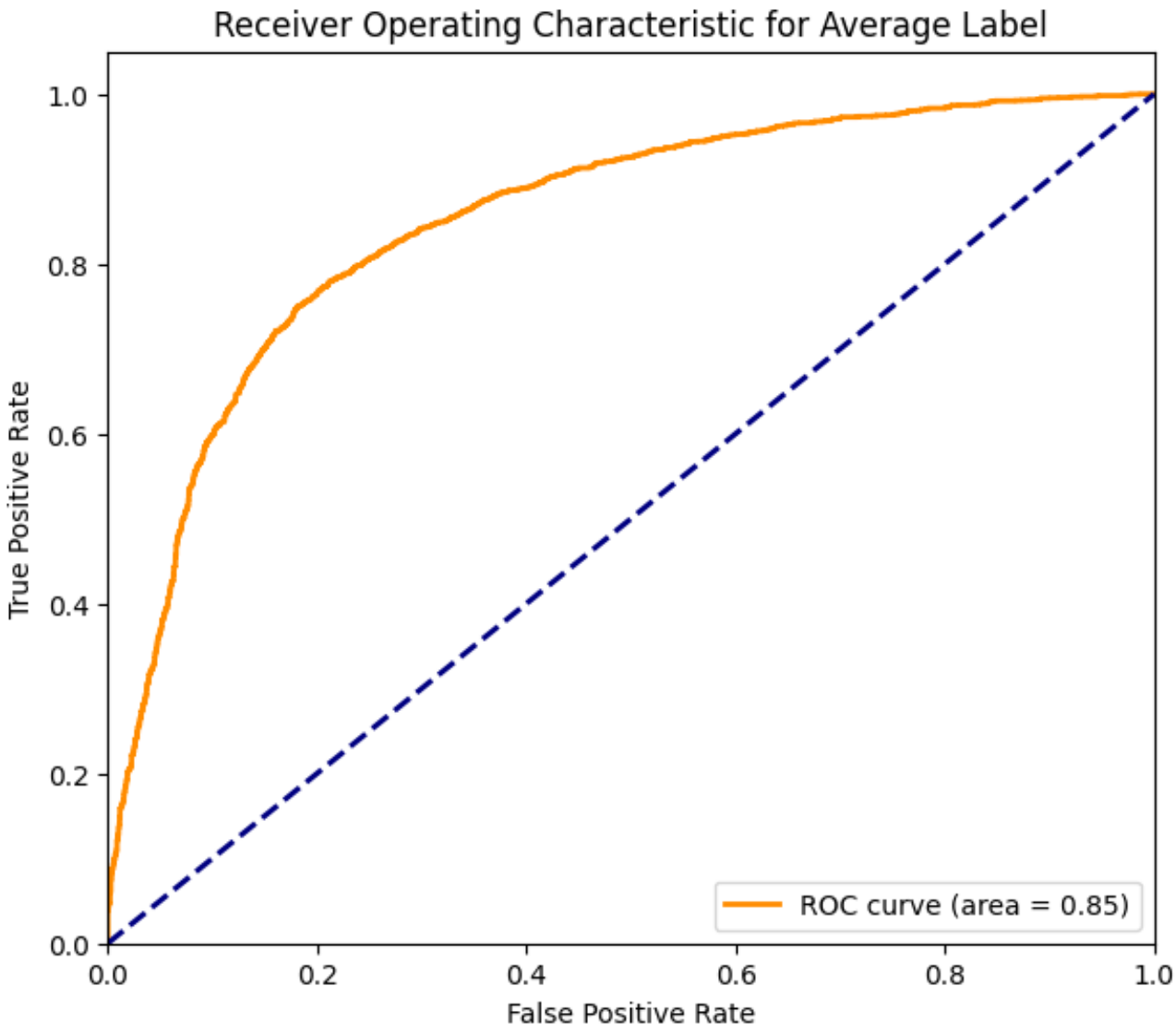



```
Mean ROC AUC: 0.8245531428027821
```

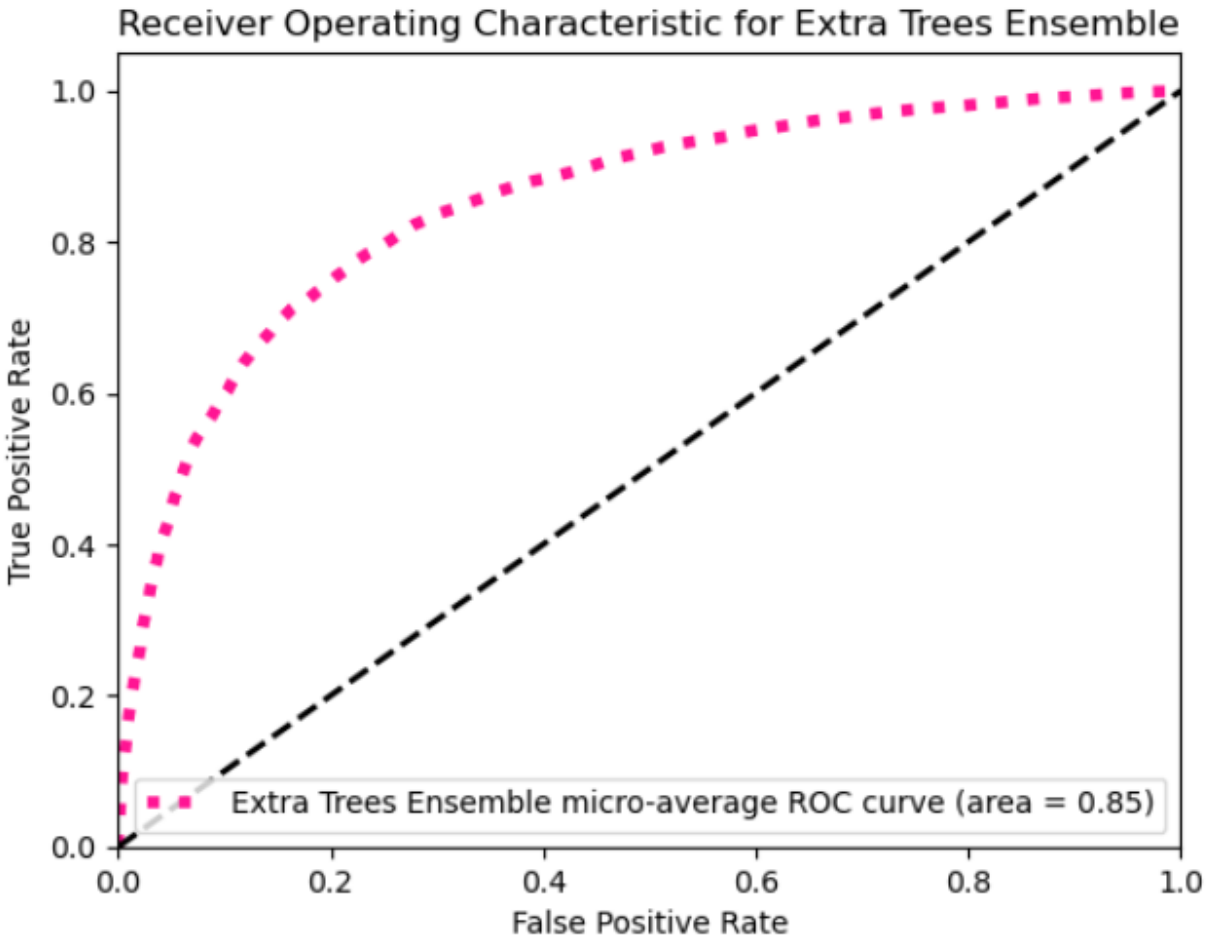
5. Classifier Chain with Logistic Regression:



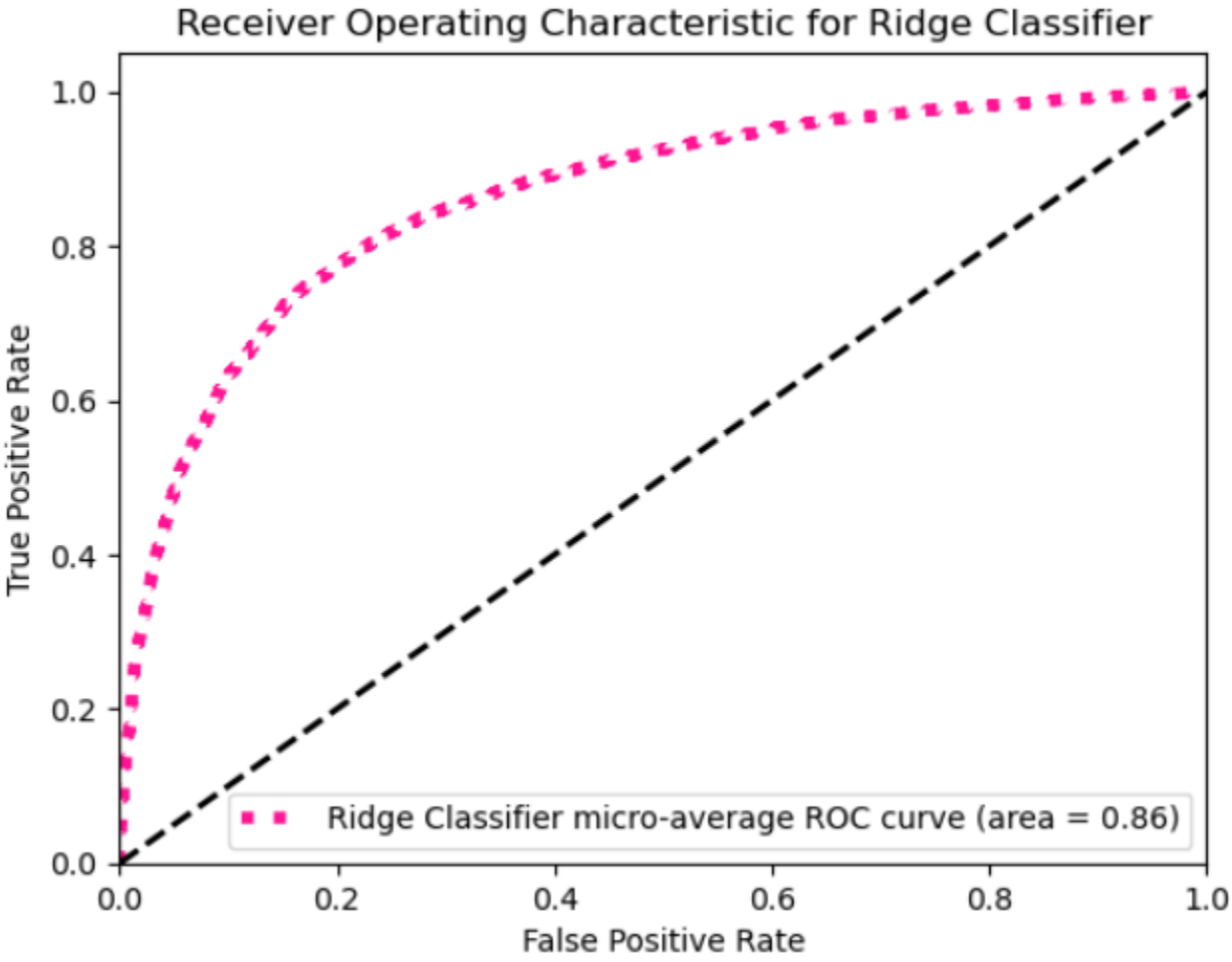



Receiver Operating Characteristic for Average Label
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.85)

6. Extra Trees Classifier:



7. Ridge Classifier:



2. Explanation and Analysis [25 marks]:

For each of the 6 above algorithms, provide justification for why you chose the algorithm for your particular problem, work you had to do to tune/train the model, and discuss the effectiveness of the algorithm when applied to your data to answer questions related to your problem statement. This should include discussion of any relevant metrics for demonstrating model effectiveness, as well as any intelligence you were able to gain from application of the algorithm to your data.

Evaluation Metrics

For our multilabel classification problem we decided to go with ROC_AUC_Score and classification report as our evaluation metrics. So, for of our different models we calculated ROC_AUC_Score to check how well our models were doing.

Why ROC AUC ?

In many real-world scenarios, some labels might occur more frequently than others. The ROC AUC score is less sensitive to class imbalance, making it a useful measure in these situations.

The ROC AUC score's ability to handle imbalanced datasets, its threshold independence, and its effectiveness in providing a holistic view of model performance across all labels make it a highly relevant and valuable metric for multilabel classification problems. It enables a comprehensive evaluation of a model's ability to distinguish between different labels, which is critical in applications where each instance can belong to multiple categories.

Applying different Models.

Binary Relevance is a method used in multi-label classification, which is a type of machine learning problem where each instance (or data point) can be associated with multiple labels simultaneously, rather than just one label as in traditional classification problems.

Binary Relevance simplifies the multi-label classification problem into multiple independent binary classification problems, one for each label in the dataset.

Advantages: Simplicity: It is a straightforward approach and easy to implement as it leverages existing binary classification algorithms.
Scalability: Can be scaled easily for many labels since it treats each label separately.

Under Binary Relevance we applied 3 different algorithms:

1. Logistic Regression

Logistic regression is relatively simple and efficient in terms of computation. It doesn't require as many resources as more complex models like deep neural networks.

Logistic regression models are often more interpretable than complex models like neural networks. The weights assigned to features can provide insights into how each feature influences the likelihood of each class, which can be valuable in many applications.

Result: ROC_AUC_Score was 0.847.

2. Random Forest

Random Forest, being an ensemble of decision trees, is inherently good at capturing non-linear relationships between features and labels. This makes it suitable for complex datasets where linear models like logistic regression might struggle.

Result: ROC_AUC_Score was 0.835.

3. XgBoost

XGBoost (eXtreme Gradient Boosting) has gained popularity for multilabel classification due to several powerful features and advantages it offers. XGBoost is based on the gradient boosting framework, which iteratively adds new models that are focused on correcting the errors made by previous models in the ensemble. This approach often leads to better predictive performance compared to models that don't adaptively focus on errors.
Result: ROC_AUC_Score was 0.835.

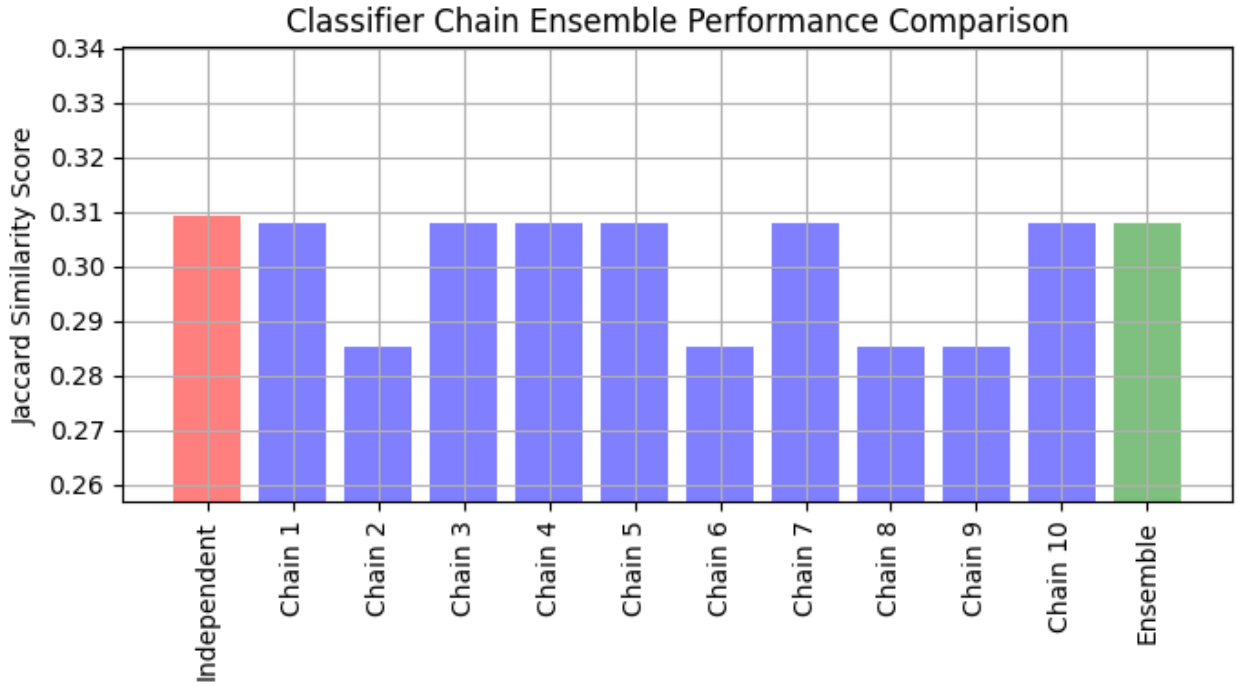
Next, we tried classifier chains.

4. Classifier chains

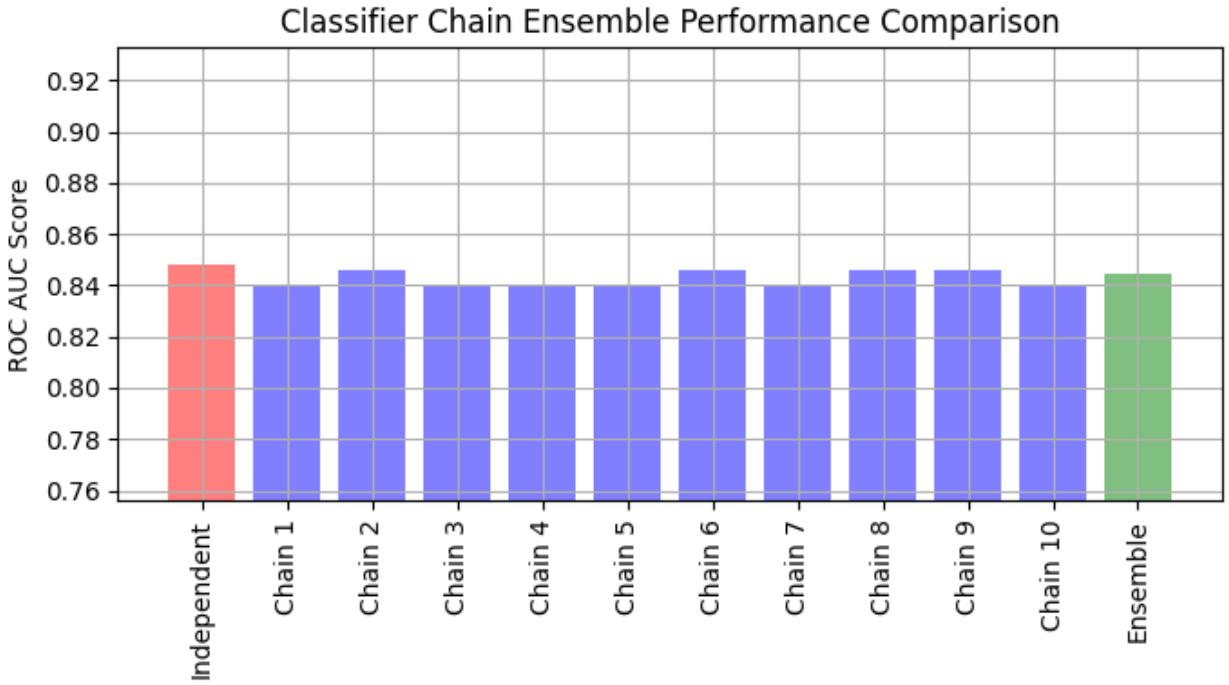
Classifier Chains extend the Binary Relevance method by incorporating label dependencies. Instead of treating each label as independent, Classifier Chains form a chain of classifiers where each classifier deals with one label and uses the predictions of all previous classifiers in the chain as additional features.

Result:
For this we calculated both the Jaccard score as well as the ROC_AUC_Score for all the models in our classifier chain as depicted in pic below.

Jaccard Similarity Score:



ROC_AUC_Score:



5. Neural Network using TensorFlow:

Neural networks, especially deep neural networks, can model complex, non-linear relationships in the data. This makes them highly effective for complicated tasks where simpler models might not perform as well. Unlike traditional machine learning models that require hand-engineered features, neural networks can automatically learn to identify the most relevant features for classification directly from the data.

Result:

The ROC_AUC_Score was 0.822. Here we notice that neural networks are performing slightly worse than our traditional models. This might be since neural networks' main power comes into play when we have enormous amount of data which can't be processed by traditional models.

6. Ridge classifier

The Ridge Classifier employs L2 regularization, which penalizes large coefficients in the model. This regularization helps prevent overfitting, making the model more robust, especially in cases with high dimensionality (as in our case we had 78 features) and collinearity among features.

Result: The ROC_AUC_Score was 0.86.

7. Extra-tree classifier:

The Extra Trees (Extremely Randomized Trees) Classifier builds an ensemble of decision trees. Each tree in the ensemble is constructed from the original dataset using the whole learning sample (rather than a bootstrap replica) and choosing split points at random. This introduces additional randomness into the model, compared to a regular Random Forest.

In our dataset we observed in phase one that there were many nonlinear relationships in the data and as it is effective in capturing complex, non-linear relationships in the data, making it suitable for complex multilabel classification tasks where linear models may not perform well.

Result: The ROC_AUC_Score was 0.85.

Conclusion:

Model	ROC_AUC_Score
Logistic Regression	0.847
Random Forest	0.835
Xgboost	0.835
Neural Network	0.822
Classifier Chains	0.85
Extra Tree	0.85
Ridge Classifier	0.86

After trying hyperparameter tuning, we decided to go with Xgboost where we fine-tuned the model to get the set of hyperparameters which gave us the mean ROC_AUC_Score as 0.87.
Thus to further build our frontend we'll use XGBoost with hyperparameter tuning as out model.