Predicting Lung Disease Recovery

COMP 4448: DS Tools 2
Logan Hahn

Purpose & Significance

- Employ and compare three classification models to determine which could more accurately predict whether a lung disease patient will recover.
- Predicting recovery outcomes in lung disease patients can provide a massive value to both the patients and the healthcare providers.

Research Question

Which algorithm, Logistic Regression, Random Forest, or XGBoost, would be the most effective in predicting whether or not a patient with lung disease will recover based on their demographic, lifestyle, and clinical characteristics?

Dataset

Source: https://www.kaggle.com/datasets/samikshadalvi/lungs-diseases-dataset

Input Variables:

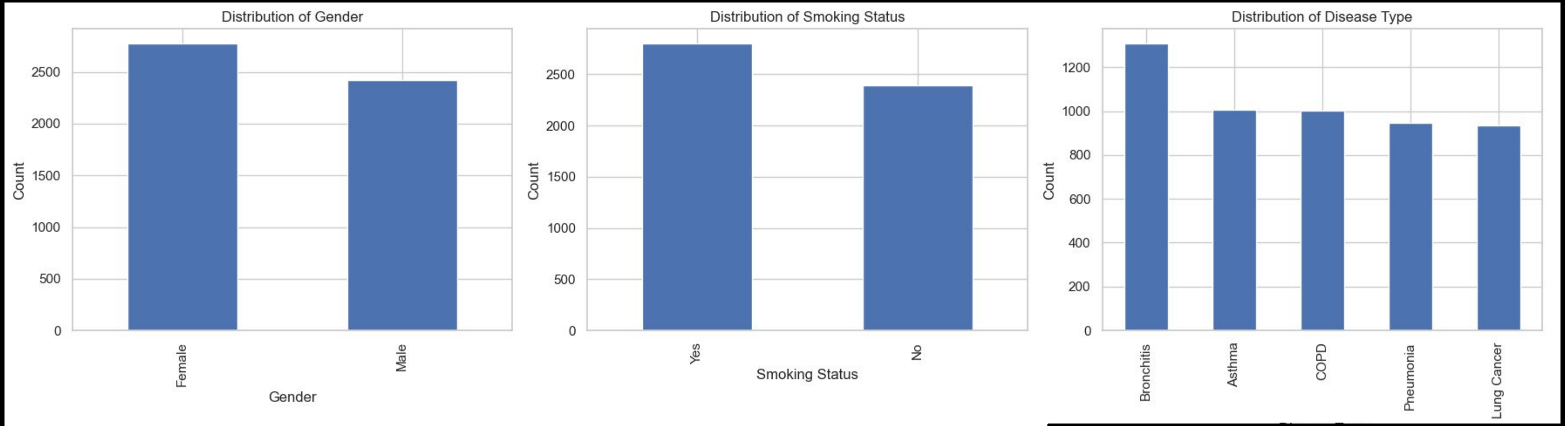
- Categorical: Gender, Smoking Status, Disease Type, Treatment Type
- Numerical: Age, Lung Capacity, Hospital Visits

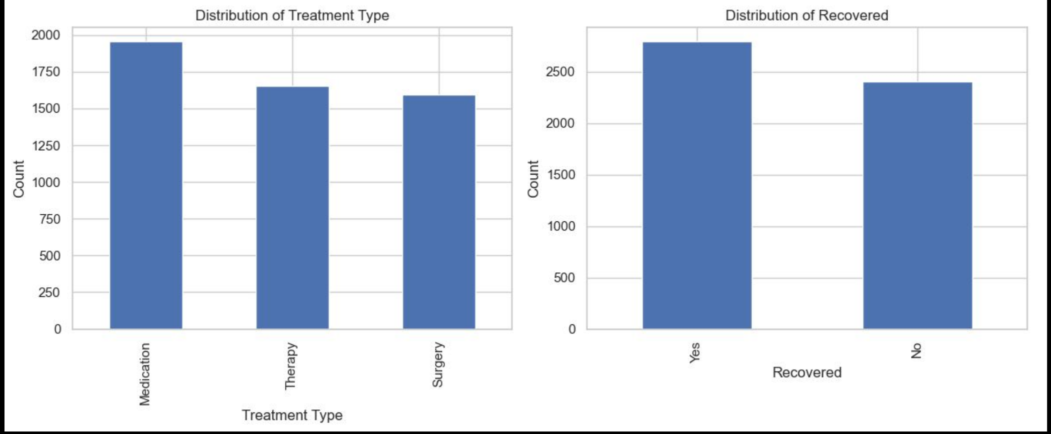
Target Variable:

- Recovered: Categorical, if a patient recovered or not

5200 Rows, 8 Columns

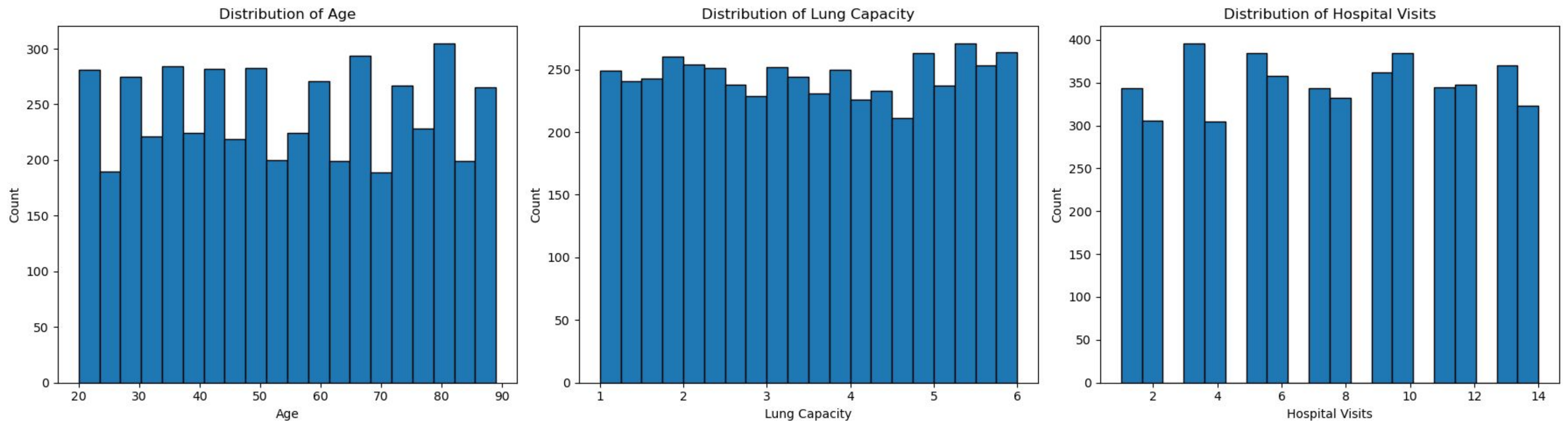
Exploratory Data Analysis





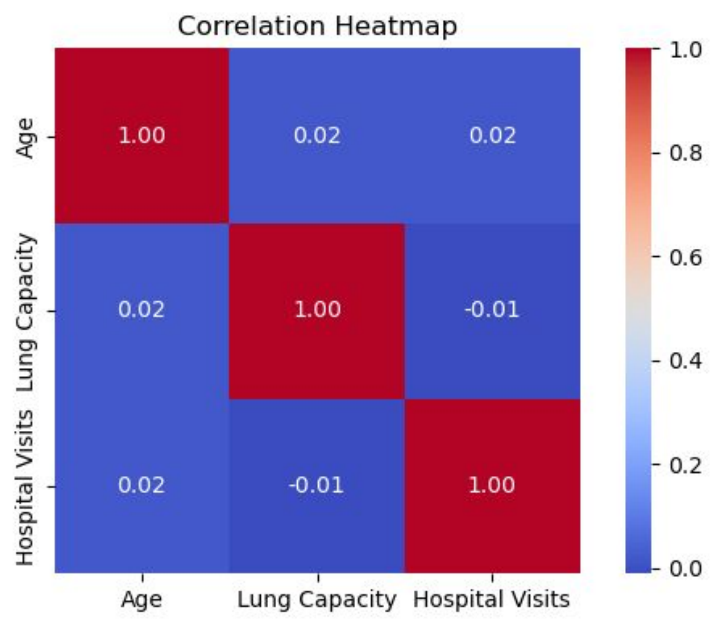
- Target variable is well balanced. (Yes=52%, No=48%)
- Distributions show categorical features are well represented

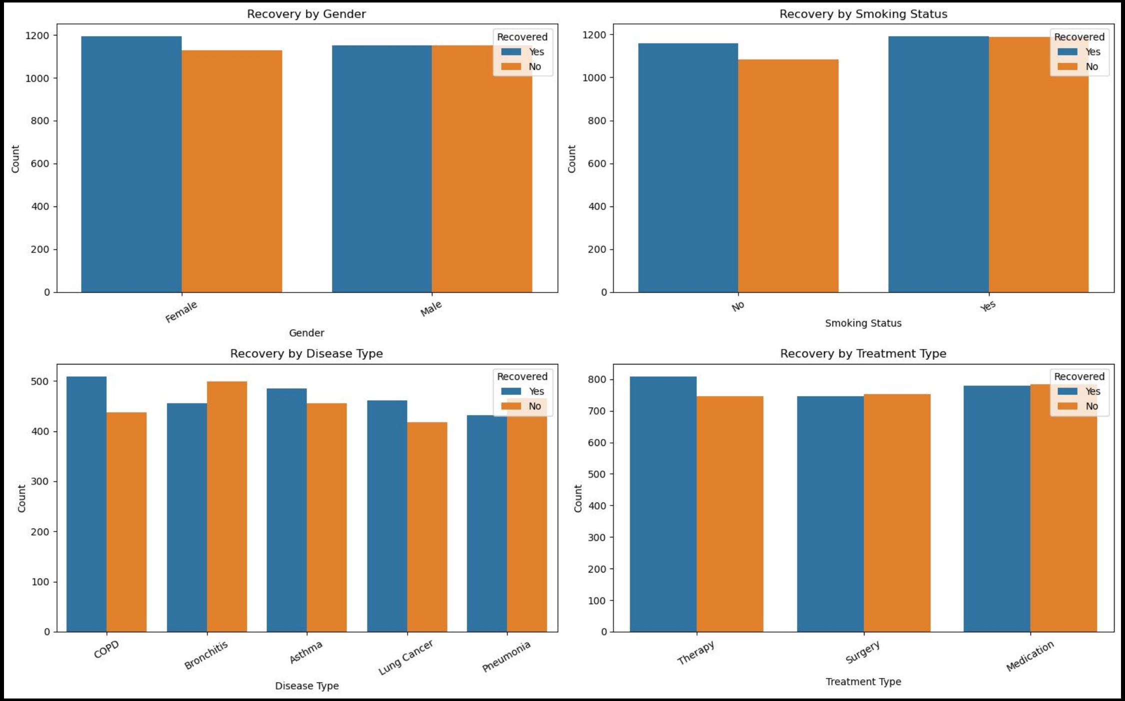
Exploratory Data Analysis



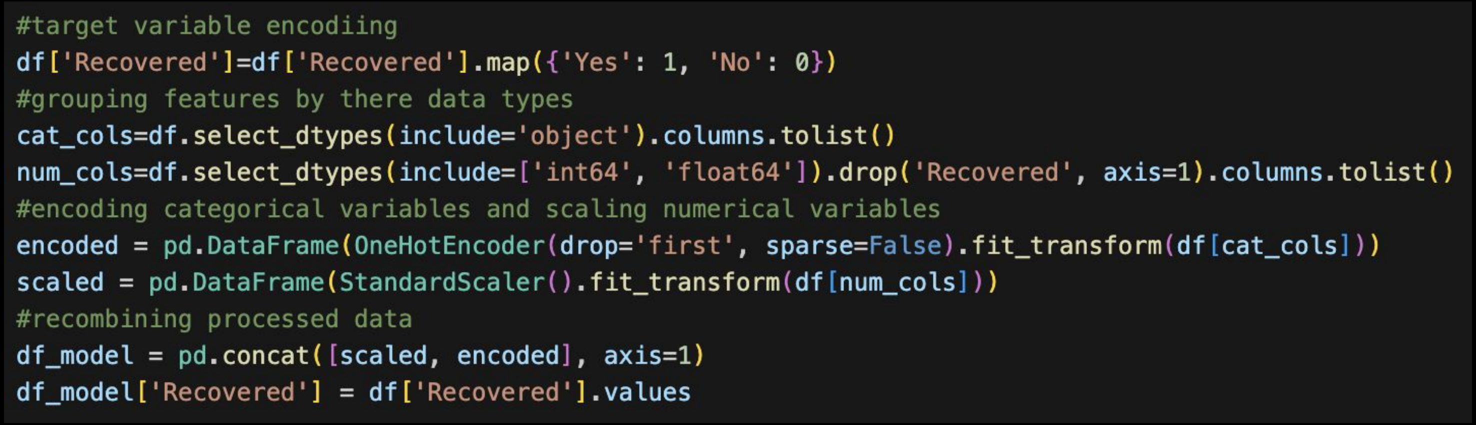
Numerical features also appear evenly distributed and well represented in the dataset.

Exploratory Data Analysis





Preprocessing:

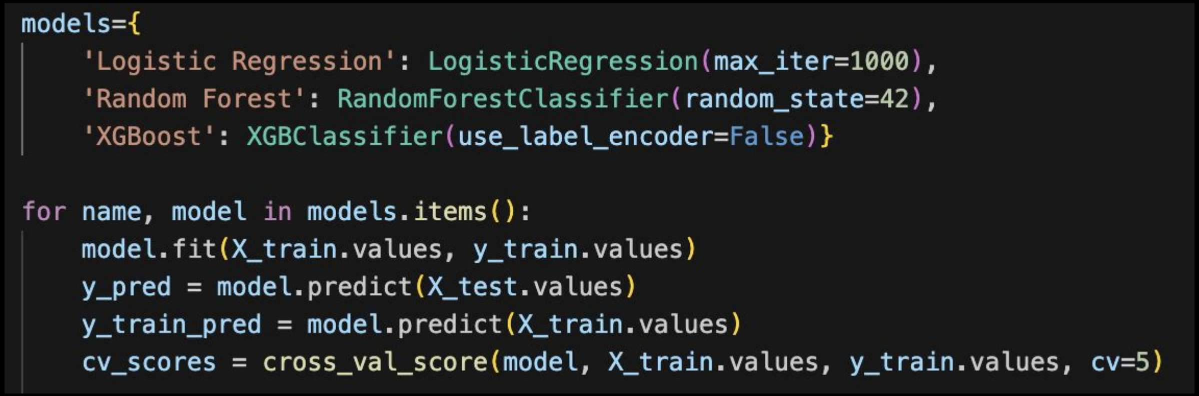


```
#target variable encodiing
df['Recovered']=df['Recovered'].map({'Yes': 1, 'No': 0})
#grouping features by there data types
cat_cols=df.select_dtypes(include='object').columns.tolist()
num_cols=df.select_dtypes(include=['int64', 'float64']).drop('Recovered', axis=1).columns.tolist()
#encoding categorical variables and scaling numerical variables
encoded = pd.DataFrame(OneHotEncoder(drop='first', sparse=False).fit_transform(df[cat_cols]))
scaled = pd.DataFrame(StandardScaler().fit_transform(df[num_cols]))
#recombining processed data
df_model = pd.concat([scaled, encoded], axis=1)
df_model['Recovered'] = df['Recovered'].values
```

Splitting:

Training and test splits were 20/80

Model Building

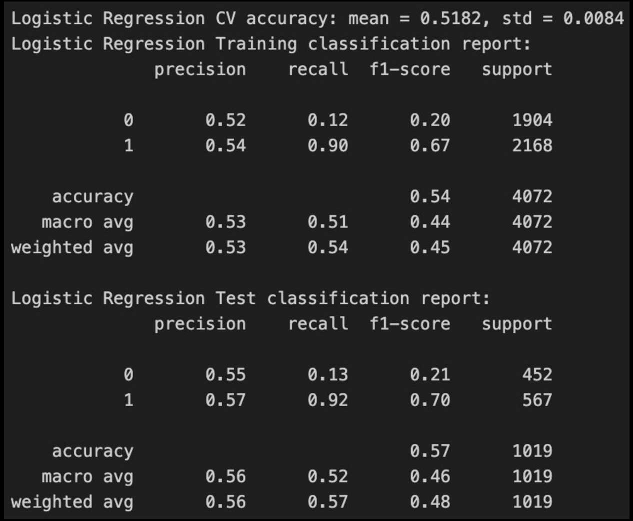


```
models={
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Random Forest': RandomForestClassifier(random_state=42),
    'XGBoost': XGBClassifier(use_label_encoder=False)}

for name, model in models.items():
    model.fit(X_train.values, y_train.values)
    y_pred = model.predict(X_test.values)
    y_train_pred = model.predict(X_train.values)
    cv_scores = cross_val_score(model, X_train.values, y_train.values, cv=5)
```

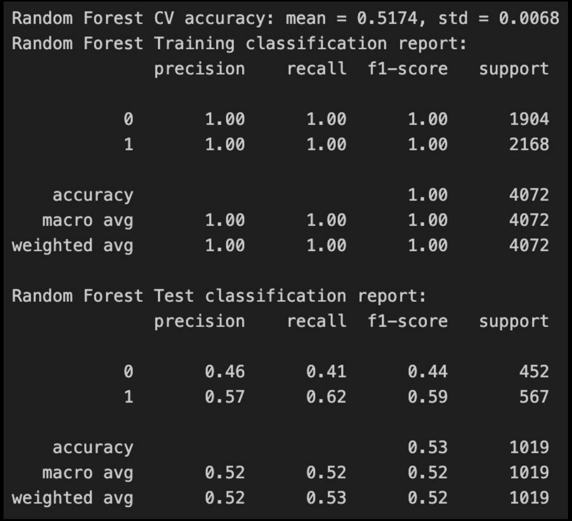
Trained each model on full training set and evaluated with cross validation.

Model Comparisons Pre-Tuning

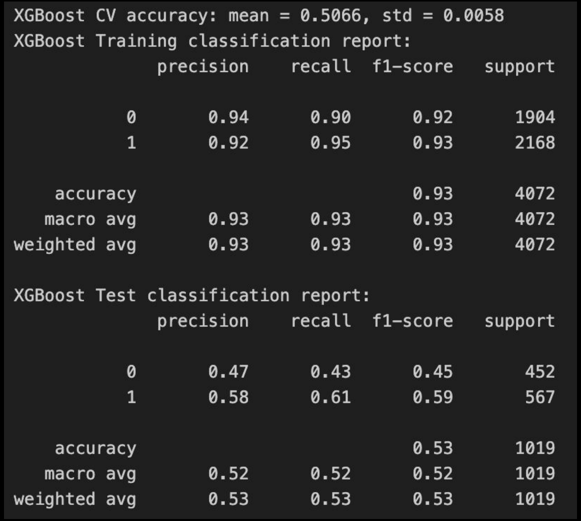
Poor performances all around, barely above random guessing (50%)



```
Logistic Regression CV accuracy: mean = 0.5182, std = 0.0084
Logistic Regression Training classification report:
              precision    recall  f1-score   support

           0       0.52      0.12      0.20      1904
           1       0.54      0.90      0.67      2168

    accuracy                           0.54      4072
   macro avg       0.53      0.51      0.44      4072
weighted avg       0.53      0.54      0.45      4072

Logistic Regression Test classification report:
              precision    recall  f1-score   support

           0       0.55      0.13      0.21       452
           1       0.57      0.92      0.70       567

    accuracy                           0.57      1019
   macro avg       0.56      0.52      0.46      1019
weighted avg       0.56      0.57      0.48      1019
```



```
Random Forest CV accuracy: mean = 0.5174, std = 0.0068
Random Forest Training classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1904
           1       1.00      1.00      1.00      2168

    accuracy                           1.00      4072
   macro avg       1.00      1.00      1.00      4072
weighted avg       1.00      1.00      1.00      4072

Random Forest Test classification report:
              precision    recall  f1-score   support

           0       0.46      0.41      0.44       452
           1       0.57      0.62      0.59       567

    accuracy                           0.53      1019
   macro avg       0.52      0.52      0.52      1019
weighted avg       0.52      0.53      0.52      1019
```

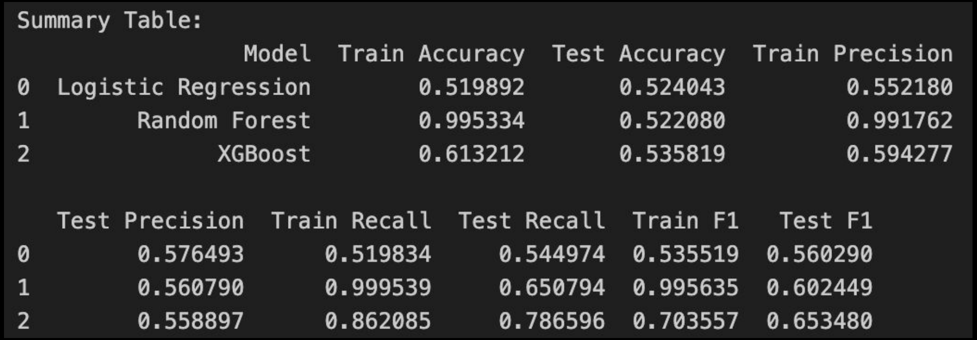


```
XGBoost CV accuracy: mean = 0.5066, std = 0.0058
XGBoost Training classification report:
              precision    recall  f1-score   support

           0       0.94      0.90      0.92      1904
           1       0.92      0.95      0.93      2168

    accuracy                           0.93      4072
   macro avg       0.93      0.93      0.93      4072
weighted avg       0.93      0.93      0.93      4072

XGBoost Test classification report:
              precision    recall  f1-score   support

           0       0.47      0.43      0.45       452
           1       0.58      0.61      0.59       567

    accuracy                           0.53      1019
   macro avg       0.52      0.52      0.52      1019
weighted avg       0.53      0.53      0.53      1019
```

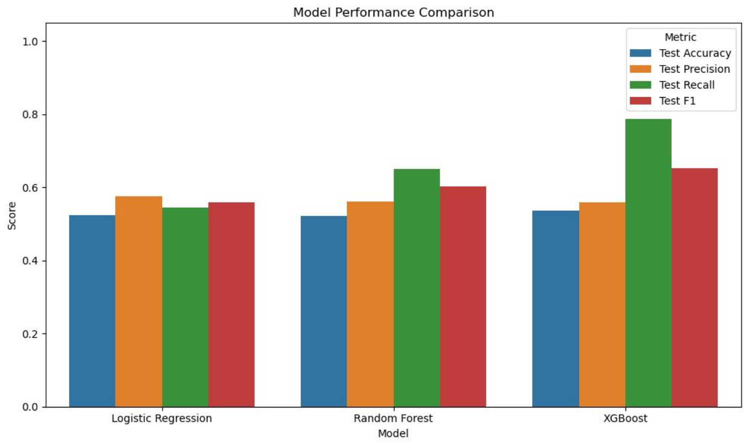
Model Hyperparameter Tuning

Tuned each model with GridSearchCV:

- Only slight improvements, nothing significant
- LR: F1 & Precision improved
- RF: F1 & Recall improved
- XGBoost: F1 & Recall improved



```
Summary Table:
                 Model  Train Accuracy  Test Accuracy  Train Precision
0  Logistic Regression        0.519892       0.524043         0.552180
1        Random Forest        0.995334       0.522080         0.991762
2              XGBoost        0.613212       0.535819         0.594277

   Test Precision  Train Recall  Test Recall  Train F1   Test F1
0        0.576493      0.519834     0.544974  0.535519  0.560290
1        0.560790      0.999539     0.650794  0.995635  0.602449
2        0.558897      0.862085     0.786596  0.703557  0.653480
```



Conclusions/Lessons Learned

- Each model had similar accuracies and performed poorly
- XGBoost performed the best overall (Highest F1-score and recall)
- Important to evaluate all scoring metrics to get full picture
- The dataset likely lacks strong predictive features and/or contains overlapping, noisy input variables.

End

THANK YOU