IIT BHU

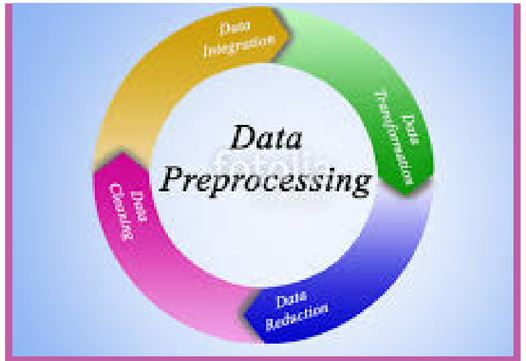
COSMIC CLASSIFIER

Presented by

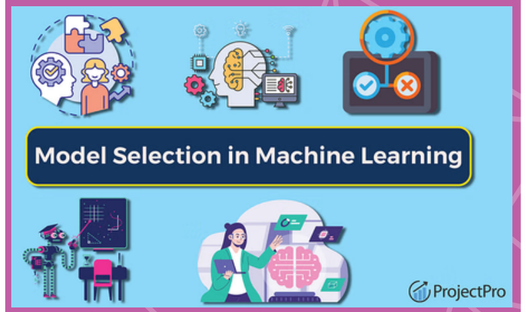
Manvi Jangid
Ricktho Sarkar
Abhinav Barman

METHODOLOGY

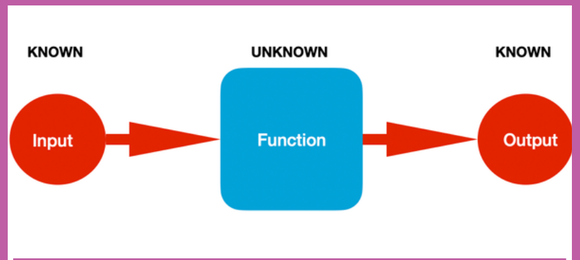
EDA



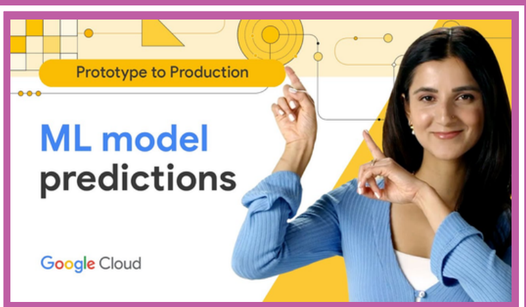
DATA PREPROCESSING



MODEL SELECTION

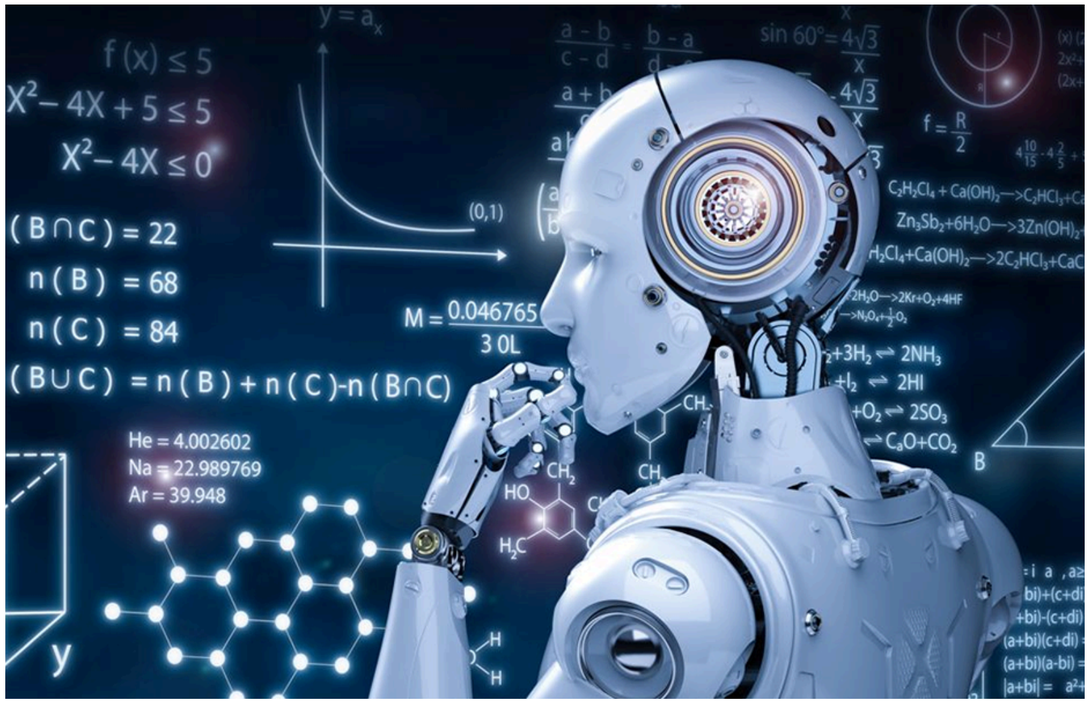


MODEL APPLICATION ON TEST SET



MODEL EVALUATION

EDA
AND PREPROCESSING



MISSING VALUE HANDLING

1. Percentage of null values of all columns were checked.
2. We converted the categorical columns ‘Magnetic Field Strength ’ and ‘Radiation Levels’ into numerical columns by just extracting the numerical parts of the column values. This was done as it would help us in applying KNN Imputation which requires numeric data only as it involves calculation of Euclidean Distance.
3. KNN Imputation was applied on all the feature columns. For the target column we avoided imputation and instead went for removal of rows with null values in target column as it was found to have improved the model performance.
4. The reason for improved performance has been concluded to be the fact that feature column imputation retains valuable data, while target column imputation may tend to introduce unreliable labels.

BOXPLOTS AND WINSORIZATION

- Boxplots have been plotted for the various feature and target columns of the dataset to check for outliers.
- Subsequently we have applied Winsorization Technique where we have used limits as 0.05 for both higher and lower sides of the distributions of the columns i.e. for each column, bottom 5% values will be replaced by the 5th percentile value and top 5% will be replaced by the 95th percentile value.

Key benefits of Winsorization are:

- Reduces the impact of extreme values on statistical analysis.
- Preserves all data points, unlike outlier removal techniques.
- Useful for handling skewed distributions in financial and scientific data.

SMOTE TO HANDLE DATASET IMBALANCE

1

BEFORE SMOTE

Prediction	
1.0	6393
7.0	5929
3.0	5814
9.0	5650
2.0	5647
6.0	5638
0.0	5637
8.0	5568
4.0	5553
5.0	5132

dtype: int64

2

AFTER SMOTE

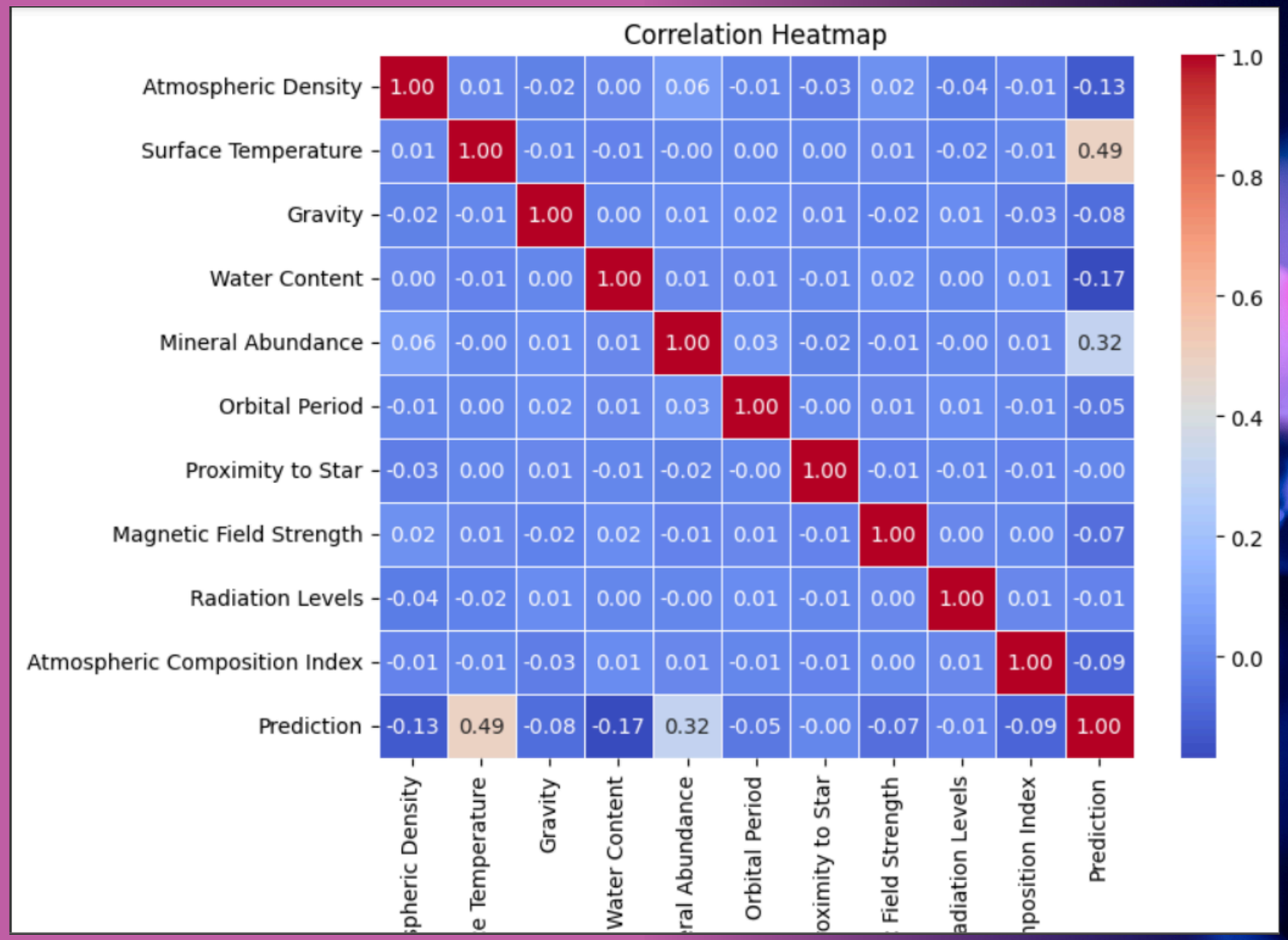
```
Prediction
5.0    6393
0.0    6393
4.0    6393
1.0    6393
9.0    6393
2.0    6393
3.0    6393
6.0    6393
7.0    6393
8.0    6393
Name: count, dtype: int64
```

3

WHY SMOTE?

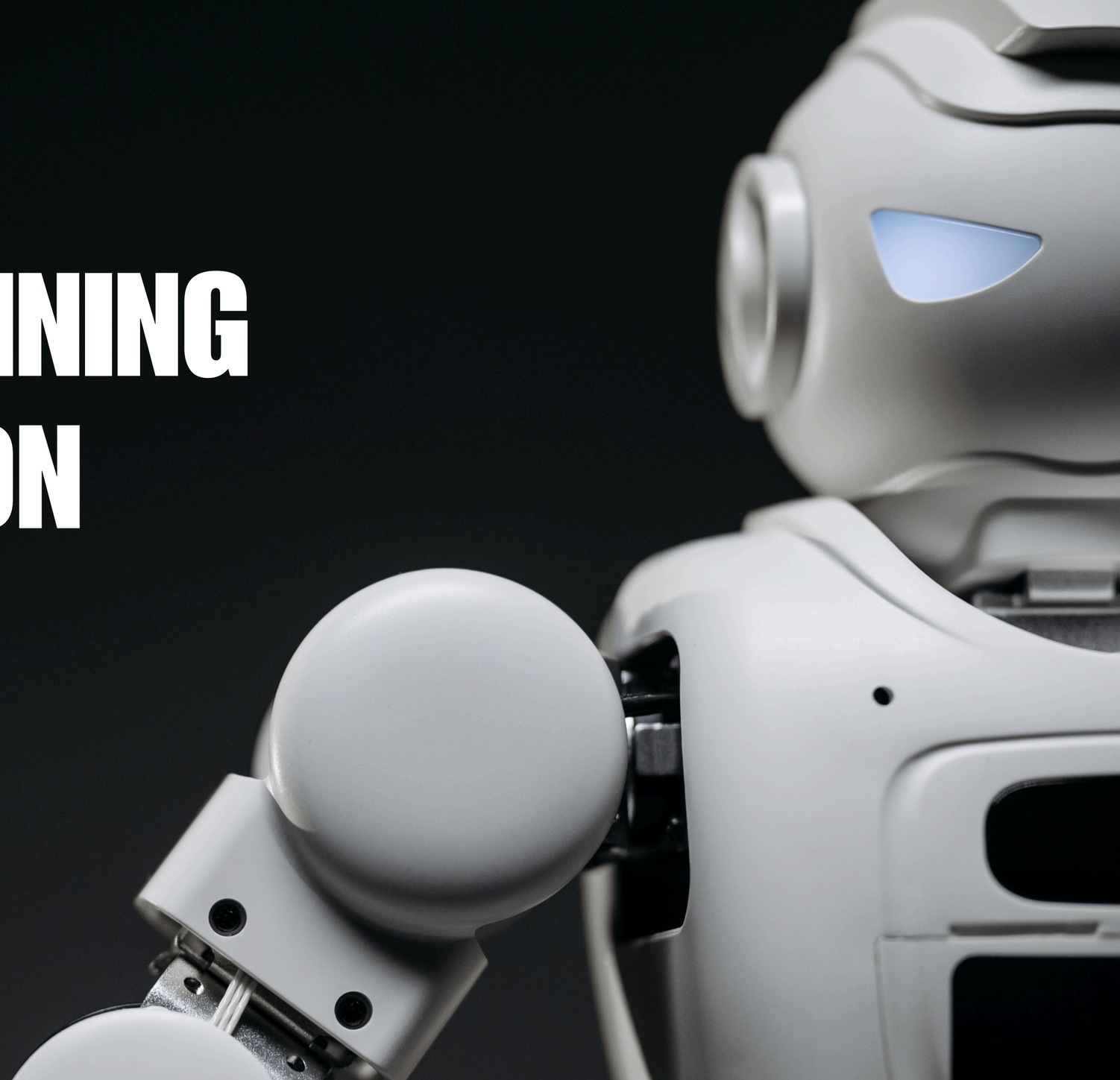
- SMOTE (Synthetic Minority Over-sampling Technique) is a method used to handle class imbalance in classification problems. It generates synthetic samples for the minority class to balance the dataset instead of simply duplicating existing instances.
- SMOTE was chosen as the primary imbalance handling technique because it effectively addressed class imbalance without removing valuable data (as in undersampling) or simply duplicating samples (as in random oversampling).
- Also unlike class weighting, SMOTE introduced synthetic diversity, improving model performance. Since the application of SMOTE yielded significant improvements, there was no need to explore alternative techniques, making it the most efficient and effective choice for this problem

CORRELATION HEATMAP



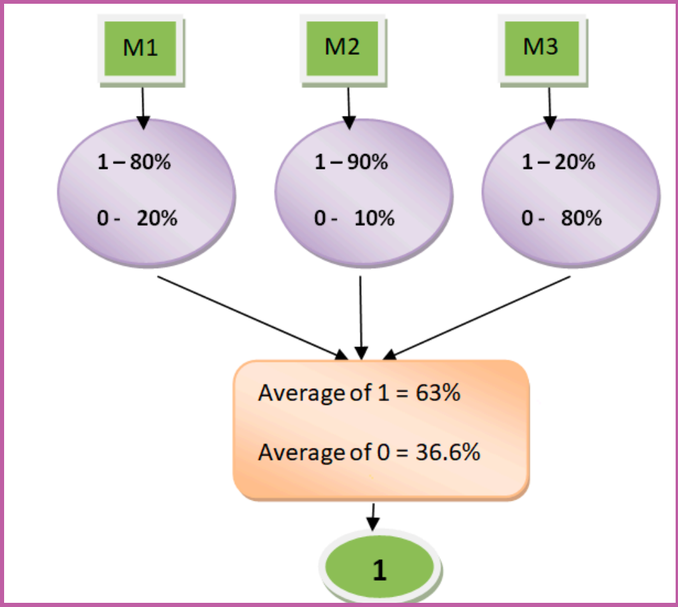
From the correlation heatmap it was concluded that there wasn't any strong/significant correlation between any of the columns.

MODEL SELECTION, TRAINING AND EVALUATION



WHICH MODEL?

- After testing with different models ,we decided to go with a Soft Voting ensemble consisting of Random Forest and XGBoost models, as it provided us with the best results.
- Fig. showing Soft Voting



MODEL TRAINING

- For training we have used GridSearch CV on both Random Forest and XGBoost models of the ensemble as a part of hyperparameter tuning process to find the best model.

```
grid_rf = GridSearchCV(rf, param_grid_rf, cv=3, scoring='accuracy', n_jobs=-1)
```

- 3 fold cross-validation was employed.
- Scoring was assigned accuracy in order to find the best possible hyperparameter combination through GridSearchCV for the Random Forest.
- Similar steps were followed for finding the best hyperparameter combination for the XGBoost.
- Finally we used the best hyperparameter combination Random Forest and XGBoost models to create our Voting Ensemble and then evaluated it.

```
from sklearn.metrics import f1_score
ensemble = VotingClassifier(
    estimators=[('RandomForest', best_rf), ('XGBoost', best_xgb)],
    voting='soft'
)

ensemble.fit(X_train, y_train)

y_pred = ensemble.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='weighted')

print(f'Ensemble Accuracy: {accuracy:.4f}')
print(f'Ensemble F1 Score: {f1:.4f}')
```

```
Ensemble Accuracy: 0.9135
Ensemble F1 Score: 0.9134
```

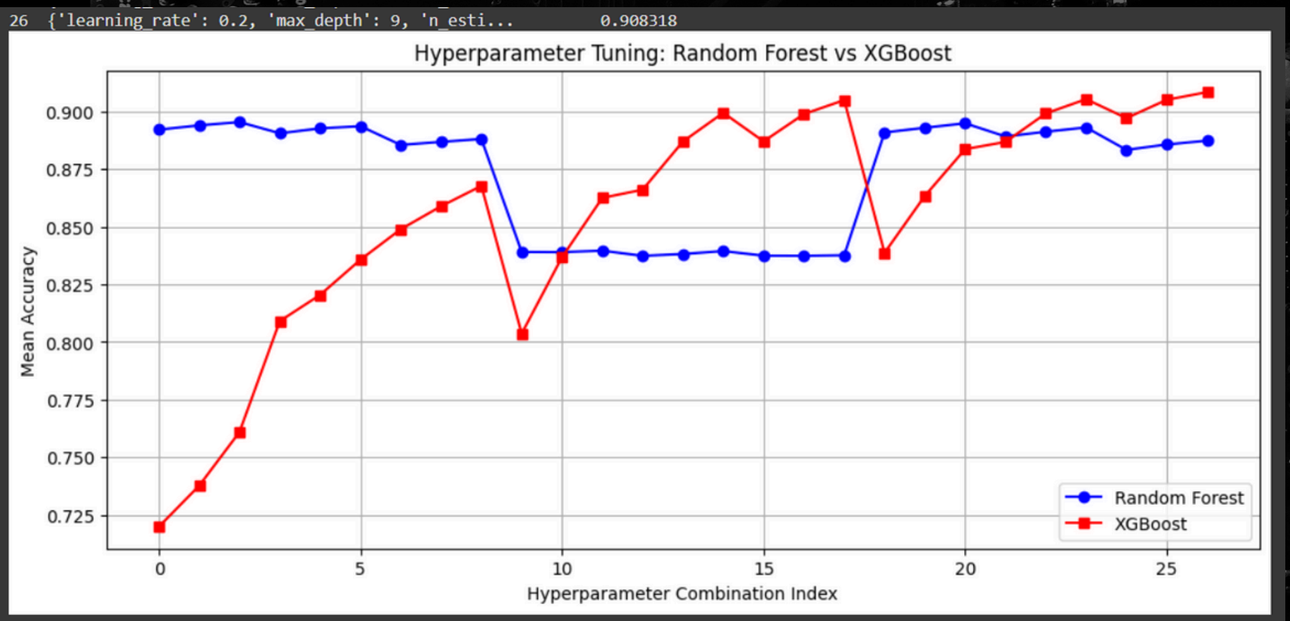
MODEL EVALUATION

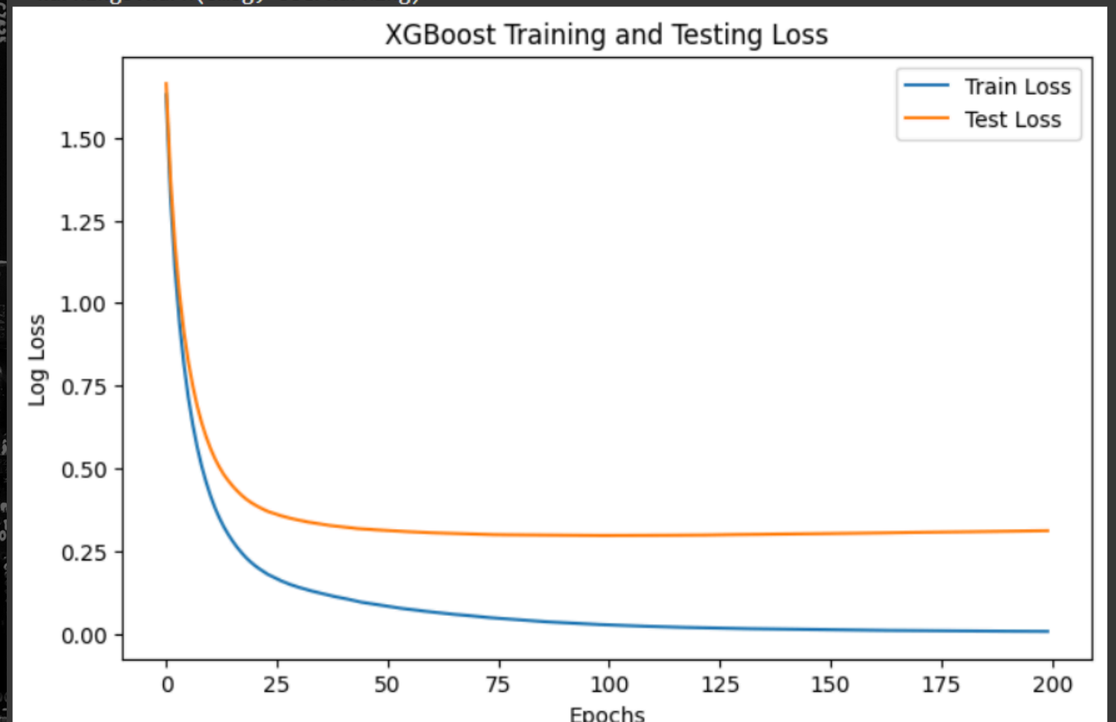
- As specified in the problem statement, we have evaluated our model on Accuracy .
- Additionally we have also taken out the F1 Score of our model as an added assurance that our model is working well.
- Although we addressed class imbalance, we included F1-score to ensure that our model maintains a balance between precision and recall. Even after balancing the dataset, the model might still favor certain classes or misclassify difficult samples. F1-score provides a more holistic evaluation, ensuring that no class is unfairly overlooked and that both false positives and false negatives are minimized.

Hyperparameter combination index

XGBoost Training and Testing Loss

```
26  {'learning_rate': 0.2, 'max_depth': 9, 'n_esti...        0.908318
```





MODEL APPLICATION ON TEST SET

```
test_df[['Magnetic Field Strength', 'Radiation Levels']] = test_df[['Magnetic Field St
    lambda x: int(x.replace('Category_', '')) if isinstance(x, str) else x
)
```

```
<ipython-input-27-7c77def5b39c>:1: FutureWarning: DataFrame.applymap has been deprecate
  test_df[['Magnetic Field Strength', 'Radiation Levels']] = test_df[['Magnetic Field S
```

```
y_pred2 = ensemble.predict(test_df)


output_df = pd.DataFrame({'Predictions': y_pred2})

output_df.to_csv("predictions.csv", index=False)

print("Predictions saved to predictions.csv")
```

```
Predictions saved to predictions.csv
```

predictions.csv

1 to 10 of 10000 en

Predictions
7.0
2.0
1.0
0.0
4.0
1.0
4.0
4.0
2.0
9.0

Show 10 per page

1 2 10 100 900

QUESTIONS AND ANSWER


THANK YOU