**Problem 2**

**Solution: -**

1. **Why did you choose the particular algorithm?**

Ans: During the EDA process, it was uncovered that data has issues with multicollinearity[1], and multidimensionality[1] (49 features). Therefore this ruled out algorithms like logistic regression that are parametric and assume there is no multicollinearity. Secondly due the high dimensionality of the data, models like KNN fail to perform because of the dimensionality curse.

Hence it was evident that one needed to use models that were robust to multicollinearity and highly skewed data, in addition the models have to be comfortable dealing with complex and multidimensional data. Which prompted me to opt for ensemble techniques like XGBoost, LightGBM, Adaboost, and RandomForest. These methods are robust to multicollinearity and excel at handling complex, high-dimensional data. Their ability to capture nonlinear relationships and interactions between features, along with built-in regularization mechanisms, makes them well-suited for achieving accurate predictions in such scenarios. The predictions of these models were combined using a voting classifier to decrease the variance of predictions.

1. **What are the different tuning methods used for the algorithm?**

Ans: With a dataset containing approximately 9500 rows, GridSearchCV was feasible due to the manageable size of the data. Since GridSearchCV exhaustively searches through a predefined hyperparameter grid, evaluating all possible combinations, it was suitable for the dataset's relatively smaller size. In contrast, RandomizedSearchCV, which randomly selects hyperparameter values from specified distributions, might not have explored the entire search space effectively with the limited amount of data. Additionally, manual tuning could have been time-consuming and subjective, especially given the dataset's size. Therefore, GridSearchCV provided a systematic and comprehensive approach to hyperparameter tuning, ensuring that I could explore various combinations thoroughly within a reasonable computational time.

1. **Did you consider any other choice of algorithm? Why or why not?**

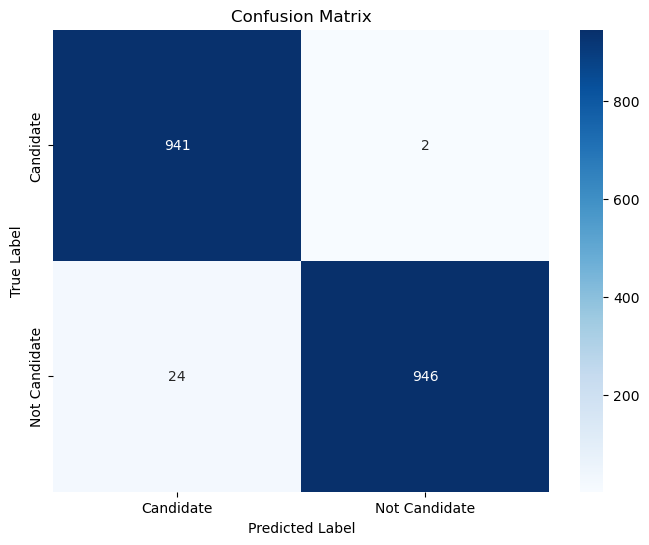
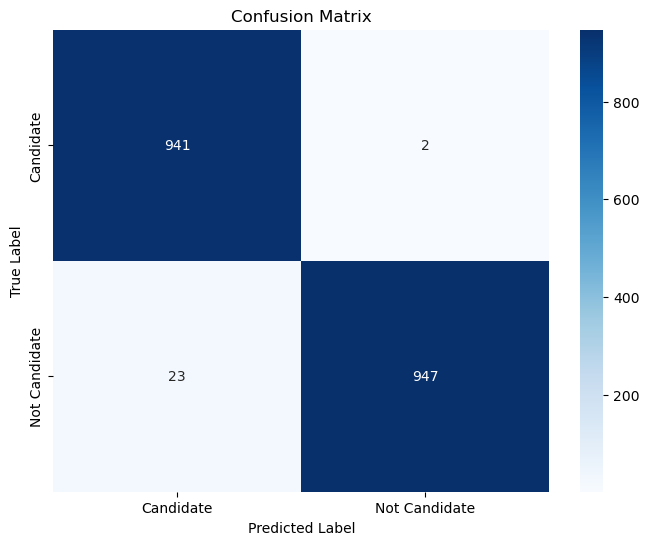
Ans: I considered several algorithms for the task, including XGBoost, LightGBM, Random Forest, and AdaBoost. These algorithms were chosen for their robustness to handle complex and multidimensional data, as well as their ability to handle multicollinearity effectively. Additionally, ensemble methods like Random Forest and AdaBoost are known for their ability to reduce overfitting and improve generalization performance. XGBoost and LightGBM are gradient boosting frameworks that are highly efficient and can handle large datasets with high dimensionality.

I did not opt for algorithms like Logistic Regression, K-Nearest Neighbors (KNN), or Naive Bayes due to their inherent limitations. Logistic Regression assumes linear relationships between features and the target variable, which may not be suitable for complex and nonlinear relationships present in the data. KNN suffers from the curse of dimensionality, making it inefficient for high-dimensional data. Naive Bayes assumes independence among features, which may not hold true for many real-world datasets.

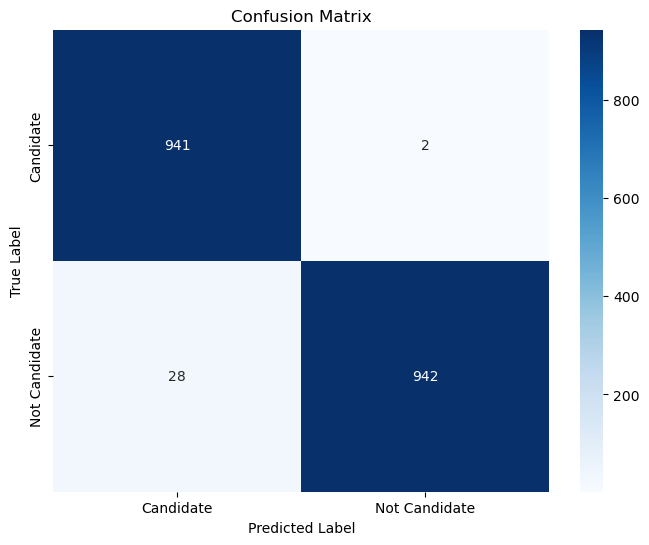
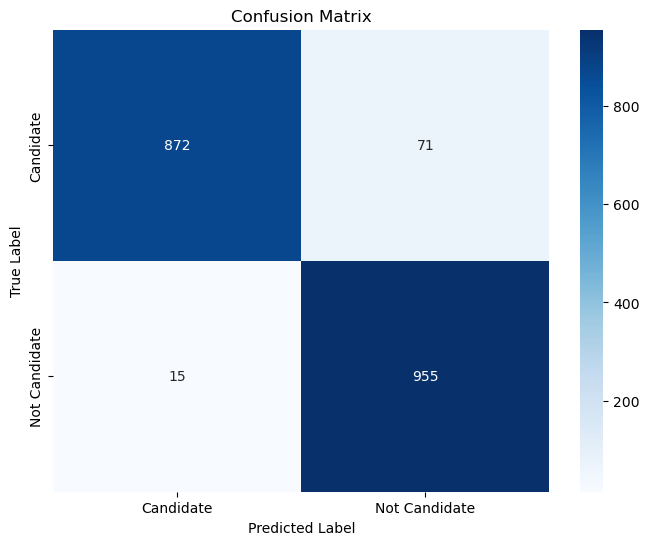
1. **Model performance and metrics:**

Ans: Models on the whole performed identically on Test data, with XGBoost and LightGBM achieving the highest performance of the lot. The models were able to score close to 99% on accuracy, precision and recall. This indicates an allround good performing model, however the overly optimistic score of 99% could also be because the train-test split of 80-20, left a very small portion of test data. 20% of 9500 roughly 1900 samples, the small test set could exaggerate the appearance of models in metrics. However a good performance on both training and test data are always indicative of highly generalizable models even when exposed to unknown training data. Random forest was the poorest of the lot, because of its poor precision score of 92%, meaning 92% of the time it is misidentifying non candidates as candidates ie false alarms. This is because random forest does not employ boosting to sequentially improve the model by improving its performance on misclassified samples.

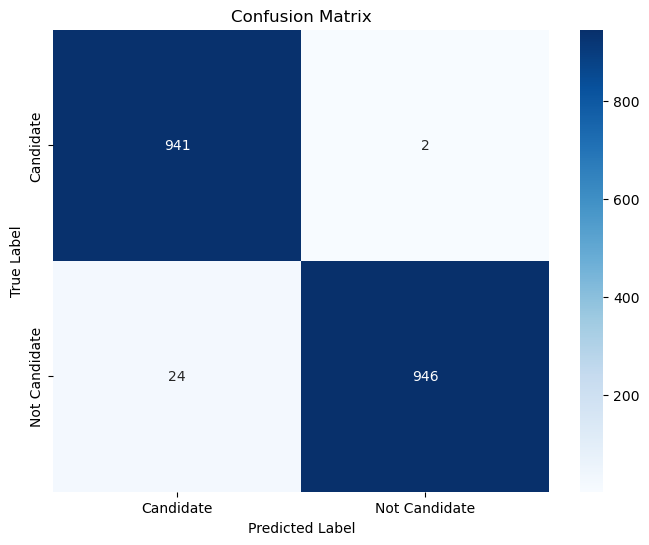
The ROC curves of all the models are good, which can be represented by the ROC curve hugging the top left corner tightly. The Voting classifier performs optimally as expected and overcomes the weaknesses of individual weak learners like Random forest. Next page is a collection of model metrics/

** **

**XGboost confusion matrix LightGBM confusion\_matrix**

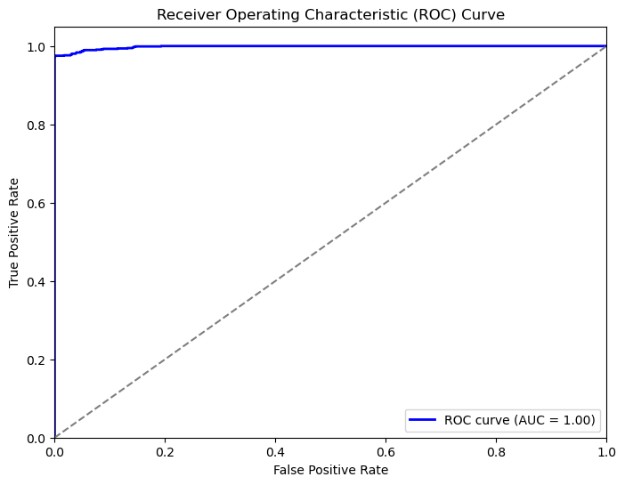
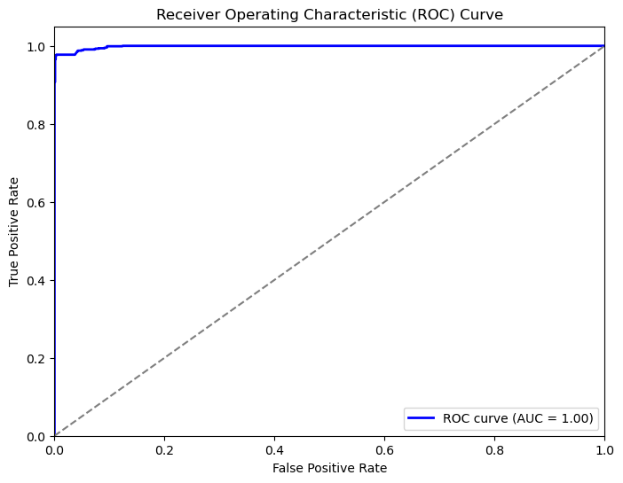
** **

**Adaboost Randomforest**

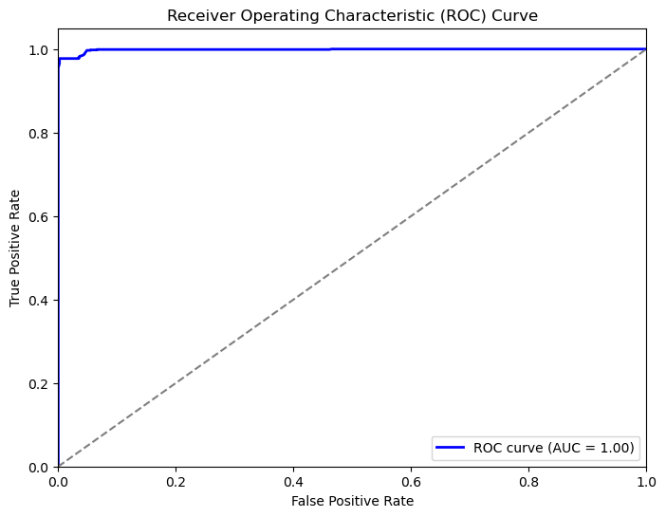
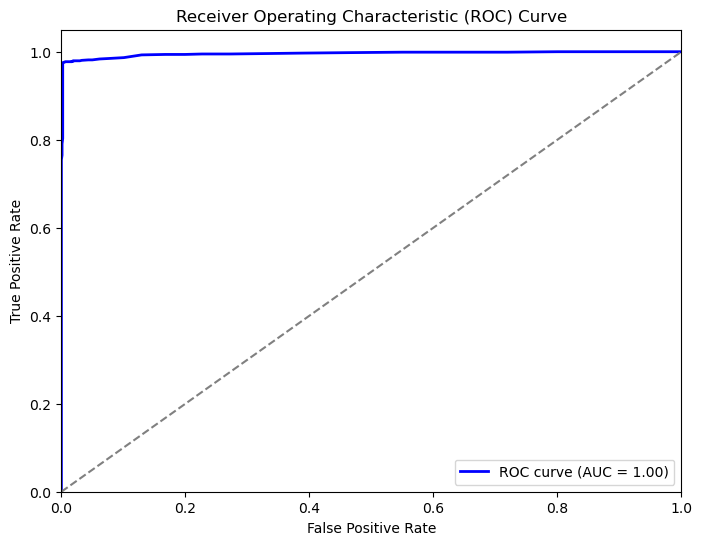
****

**Votingclassifier**

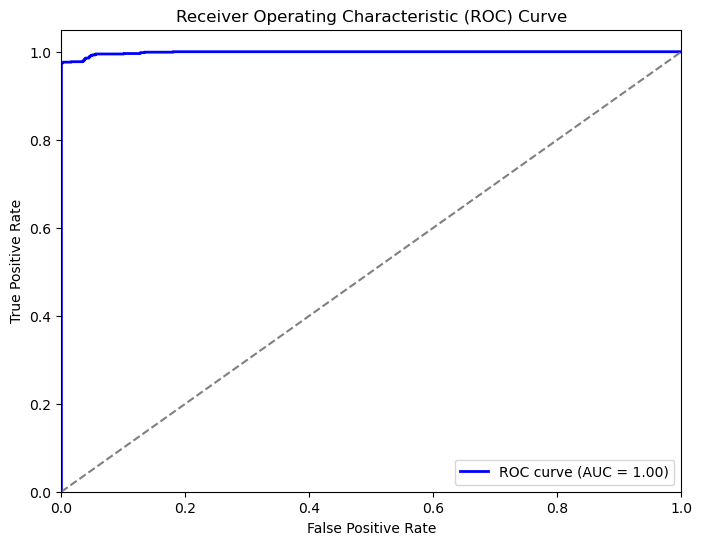
**\*NOTE- THE ROC(BLUE LINE) IS TIGHTLY HUGGING THE LEFT CORNER IE YAXIS AND OVERLAPPING**

** **

**XGBoost ROC curve LightGBM ROC**

** **

**Adaboost ROC RandomForest ROC**

****

**VotingClassifier ROC**

[1] <kepler_raw_report.html>

[2]<kepler_treated_report.html>