**SOLUTION DESCRIPTION**

To start off with this dataset, I imported it in the form of a dataframe and I immediately noticed that some of the columns had a lot of NaN values. To tackle this issue, I took multiple steps. First, I calculated the percentage of NaN values in each column of the dataframe. I dropped all the columns which had more than 75% of NaN values. Second for columns with less than 5% of NaN values, I simply dropped the rows which had the NaN value. For the columns with NaN percentage of >5% and <75%, I filled the NaN values up with the mean. Once I finished all these steps, I perform the same steps for the test data (except for the step where I drop the rows). I perform EDA and I create a correlation matrix to determine how each column is correlated to each other. I also create some boxplots which indicate the presence of some outliers. So, I try to detect these outliers with the help of Isolation Forest and I decide to drop the rows containing the outliers. After I create a new dataset ready for training, I make use of three models:

* KNN Classifier
* RandomForest Classifier
* XGBoost

XGBoost performed the best out of all three so I leveraged that model to predict on the provided test data.