Work Log Report for Machine Learning Models

# Before 24/09/2024

-Before implementing any machine learning models, we decided to clean the validated data suitable for Random Forest Classification models. To do this;

-Dropped Columns such as Building No, Fan\_on\_group , cumulative\_fan\_on\_mins, date, time and year for all Dataframes and more.

-Hot Encode some columns for random forest because if we didn’t, then random forest will not work.

-Convert date time columns for suitable variables types for Random Forest

-We additionally filtered rows where fanstatus was on so that the ML Models could work properly.

-We dropped indices to make it cleaner

-Dropped the ‘Faulty’ Column as it was no longer needed.

-Extracting and Converting DateTime:

-Decided to convert all datetime columns into datetime objects using pandas to\_datetime() function.

-Allows for extraction, resampling and time based filtering.

-Handles Missing or Invalid Dates converting them to NAT instead of raising errors.

-Decided to extract day of year from datetime column and create new column called Day of Year in all dataframes.

-Can be useful for analyzing seasonal trends/patterns in data that repeat annually.

-Feature Engineering; helps model learn time-based patterns.

-Decided to make new column called Minutes\_Past\_Midnight that represents the number of mintes that have passed since midnight for each timestamp in DateTime.

-Represents time in a continous format, which can be useful for analysis -More useful for algorithm than separating hour and minute columns.

-Allows to analyze daily patterns like determining common events and behaviors.

-Undersampling the Data

-We under sample the majority class in the Dataframes to balance the dataset by reducing number of rows for the majority class .

-First filter the rows where Fan\_status is on or off.

-Random Sampling to select a subset of rows from major classes to undersample majority class and balance dataset.

-Identifying rows to drop: we retrieve row indices of major class and then compare the indices of the original majority class dataframe and the sampled dataframe to find the indices that were NOT selected for sampling. These Indices represent the rows that should be dropped.

-Dropping the Rows.

-If we do not balance the datasets, one class such as Fan\_status = on may dominate other Fan\_status = off. Which creates bias machine learning models.

-Filtering Out Rows

-We filter out rows based on Faulty == False so that we can keep only the rows where Faulty == False.

-Aligning Target Labels; Using filtered row indices from X\_train\_df1 and etc, to ensure that target labels y\_train\_df1 match the feature rows after filtering.

-Essential to maintain correct relationship between Features and Labels after filtering.

-We remove faulty data where Faulty == True, leaving non-faulty rows for training and evaluation.

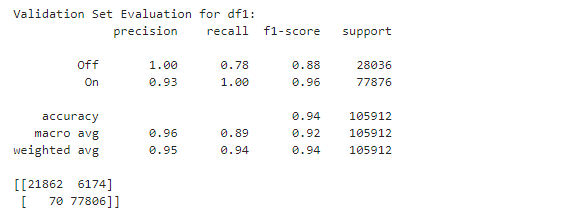
-Consistency; Making sure we align the labels.

-We decided to split the data into 80:20. With 80% of the data for training purposes, and 20% being split into 10% for Testing and 10% for Validation sets.

-Initialized 3 simpleImputers to ensure each imputer is fitted based on individual training data’s means.

-We imputed the values in X\_train, x\_val and x\_test for each dataframe.

-Decided to initialize the rf\_model\_df1 which is the random forest model for dataframe 1 with hyperparameters set as n\_estimators=100, max-depth=15, max\_features=’sqrt’, min\_samples\_split=10, min\_samples\_leaf=4, random\_state=42). This is reduce overfitting and it does as it produces the following results;



-The dataset is extremely predictable, but with enough hyperparameters with finetuning, it will provide reasonable and accurate predictions.

# KNN

Decided to choose StandardScaler for K-Nearest-neighbors due to KNN being distance based. As KNN finds the closest neighbors to a given data point, using Euclidean distance or manhattan distance to calculate that distance between two datapoints. These distances are impacted heavily by the scale of the features. If one feature has a larger range of values than the others, it will dominate the distance metric, even if it is not more important than others.

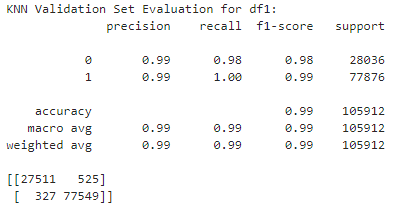
Examples such as Age and income, without scaling the distance between two points will be heavily influenced by the income feature since the range is much larger.

StandardScaler standardizes the features by removing the mean and scaling them to unit variance (mean of 0 and standard deviation of 1). Makes all features comparable on same scale.

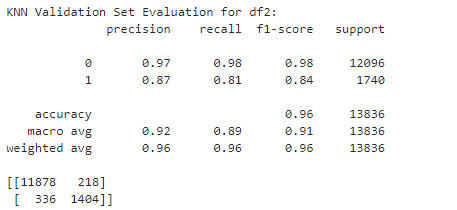
StandardScaler is applied to input features and not the target variable Y aka the Fan\_status. It ensures that features such as Zone\_temp and dew\_temp are scaled to the same range, which is important to the distance based KNN.

# First Results with Grid Search on KNN & XGB [Fastest Parameters]

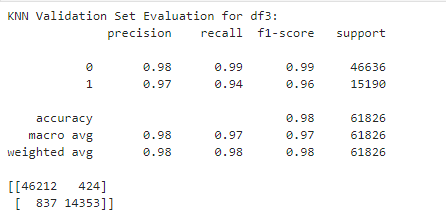
DF1 KNN:



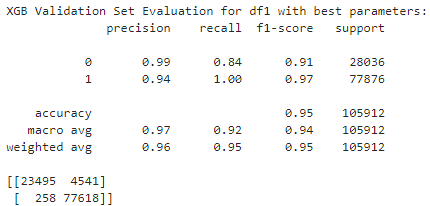
DF2 KNN:



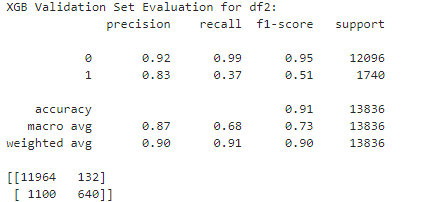
DF3 KNN:



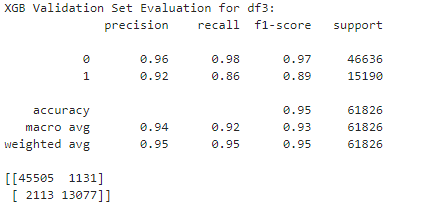
DF1 XGB:



DF2 XGB:

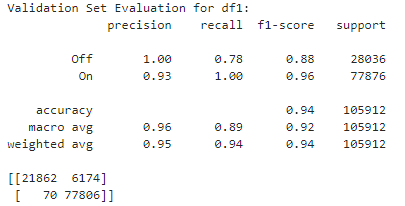


DF3 XGB:

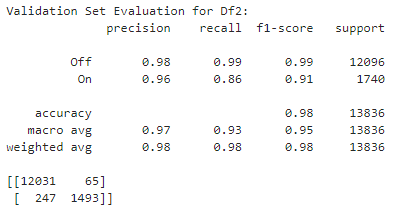


# Random Forest Initial Results with no GridSearch

DF1 with Testing Parameters; rf\_model\_df1 = RandomForestClassifier(n\_estimators=100, max\_depth=15, max\_features='sqrt', min\_samples\_split=10, min\_samples\_leaf=4, random\_state=42, n\_jobs=-1)



DF2 with 0 Parameters;



DF3 with 0 Parameters;

