Wind Turbine

Data Science Case Study

PROBLEM STATEMENT

- ➔ Wind turbines are key to renewable energy production. Each turbine's operational health must be monitored to avoid failures and maintain efficiency.

- ➔ Provided with monthly aggregated data from turbines, and the objective is to predict whether a wind turbine is in a normal(2) or abnormal(0) state (binary classification)

BINARY CLASSIFICATION PROBLEM

EDA & FEATURE ENGINEERING

DATASET - FEATURES

✅ Dataset Summary

Feature	Type	Description
`WTGID`	Categorical	Wind Turbine ID
`Loc`	Categorical	Location
`MonthStartDate`	Date	Month starting date (can extract Month)
`f1` to `f6`	Numerical	Engineered numerical features
`target`	Categorical	Classification label (binary or multi)

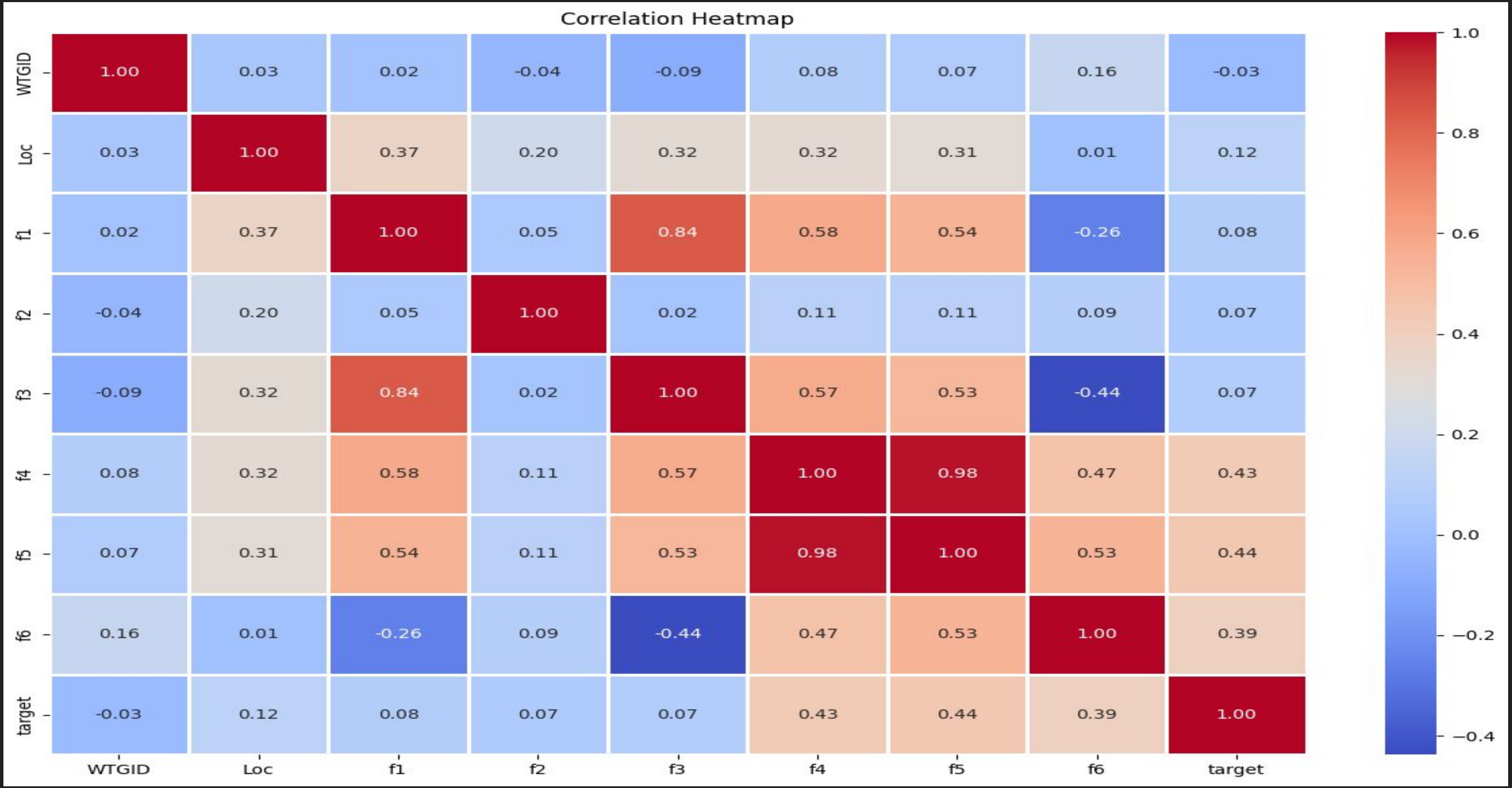
Key Observations from the dataset

- ➔ The input feature has categorical, numerical and date features
- ➔ The output features is binary classified
- ➔ The dataset must be split based on Training and Validation
- ➔ The dataset has no NULL values
- ➔ There is some correlation between the independent features
- ➔ The dataset has outliers and needs to be treated
- ➔ The dataset is highly imbalanced and needs resampling
- ➔ Gradient Boost ML classifier is selected
- ➔ Important Features for target classification (Decision Tree: Gini Index, Information Gain)
- ➔ Necessary evaluation metrics : Accuracy, F1-score, Precision, Recall, AUC_ROC Score
- ➔ Comparison of Gradient Boost classifier with other ML classifier is to be carried out

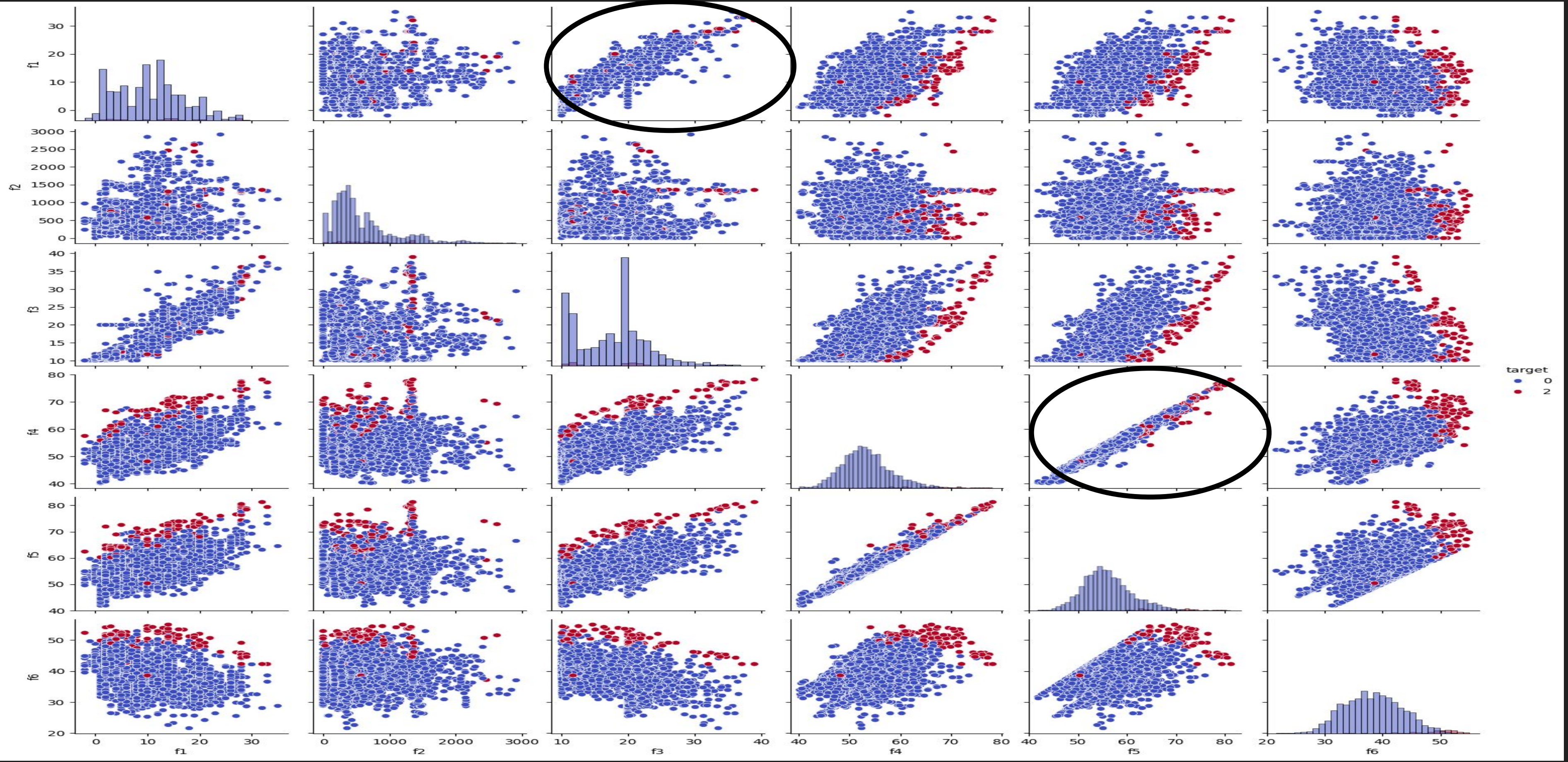
Correlation between Features

	WTGID	Loc	f1	f2	f3	f4	f5	f6	target
WTGID	1.000000	0.034182	0.017065	-0.040475	-0.088163	0.084385	0.071490	0.163242	-0.026732
Loc	0.034182	1.000000	0.365937	0.204213	0.320323	0.321758	0.309555	0.008818	0.118511
f1	0.017065	0.365937	1.000000	0.054668	0.836823	0.579921	0.541918	-0.258967	0.080514
f2	-0.040475	0.204213	0.054668	1.000000	0.024391	0.109901	0.110504	0.092536	0.066290
f3	-0.088163	0.320323	0.836823	0.024391	1.000000	0.570278	0.526217	-0.437809	0.074970
f4	0.084385	0.321758	0.579921	0.109901	0.570278	1.000000	0.982203	0.471510	0.426322
f5	0.071490	0.309555	0.541918	0.110504	0.526217	0.982203	1.000000	0.534139	0.442400
f6	0.163242	0.008818	-0.258967	0.092536	-0.437809	0.471510	0.534139	1.000000	0.393216
target	-0.026732	0.118511	0.080514	0.066290	0.074970	0.426322	0.442400	0.393216	1.000000

Correlation between Features



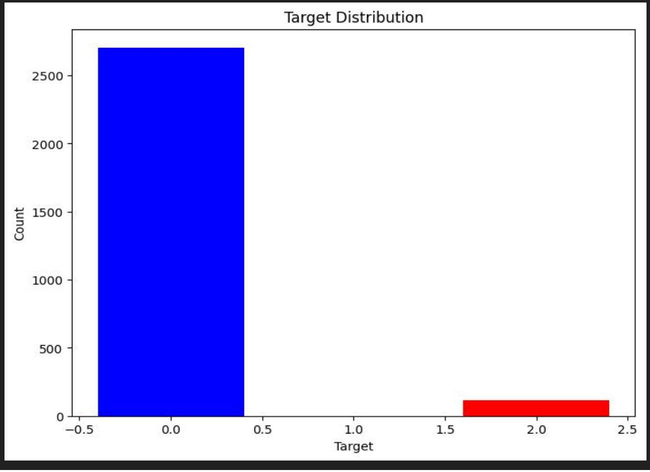
Correlation between features (pairwise)



Imbalance Dataset - UPSAMPLING

➔ The data is highly imbalance

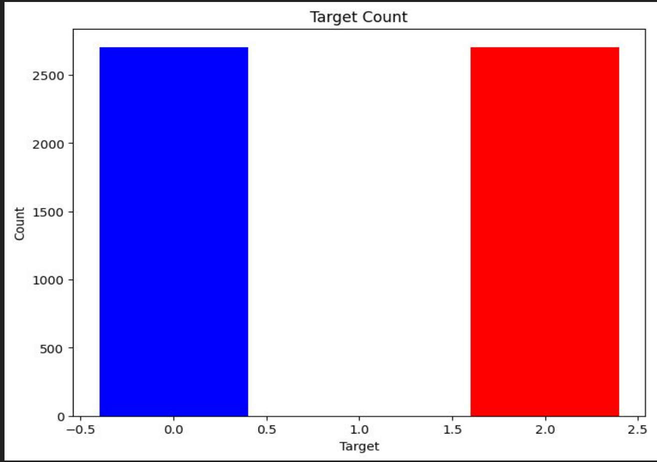
	count
target	
0	2705
2	115



➔ Solution : Upsampling Target feature “2” using resample() with replacement

	count
target	
0	2705
2	2705

dtype: int64



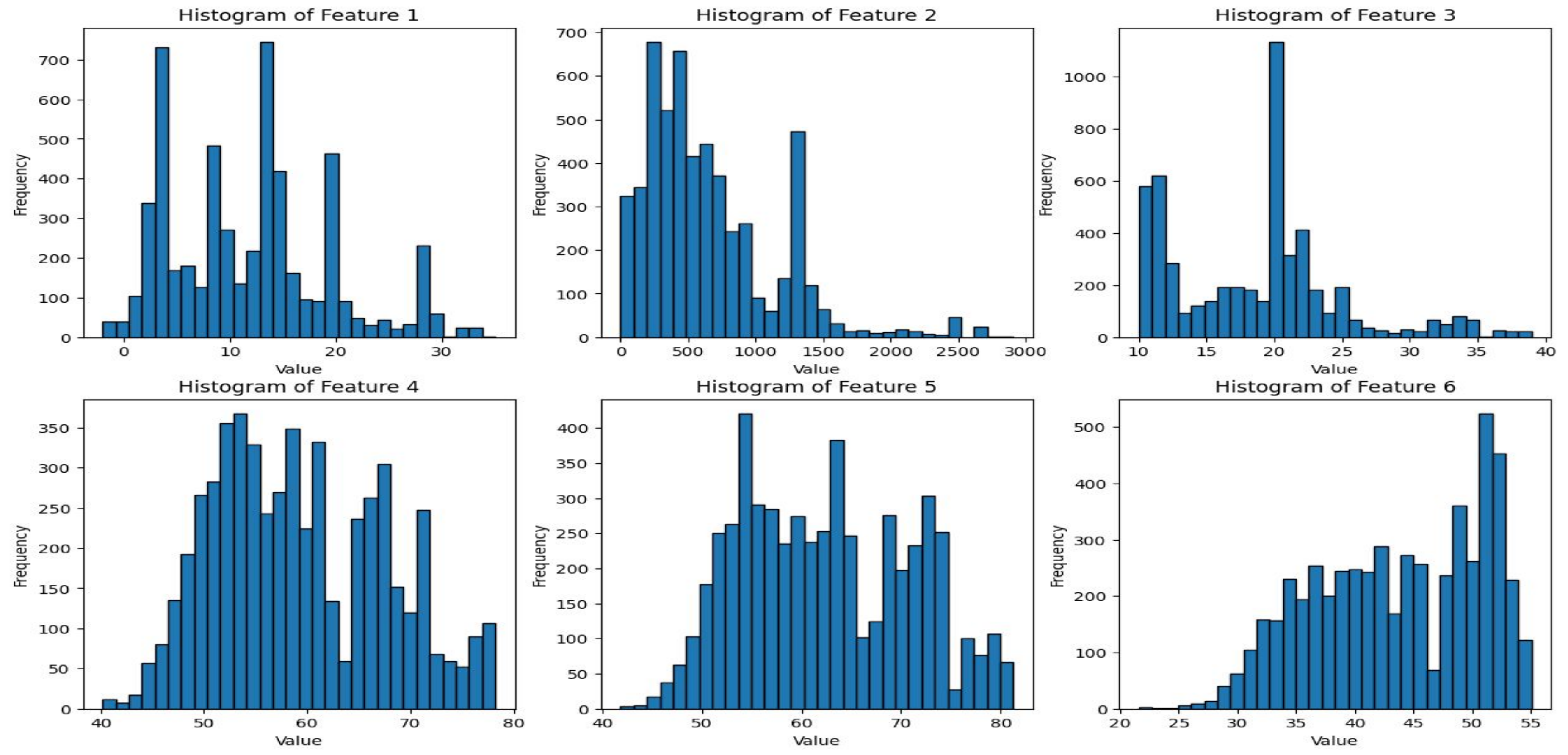
Data Description before upsampling

	WTGID	Loc	f1	f2	f3	f4	f5	f6	target
count	2820.000000	2820.000000	2820.000000	2820.000000	2820.000000	2820.000000	2820.000000	2820.000000	2820.000000
mean	216.631915	0.061702	10.729078	590.811120	17.829852	53.851848	56.548878	38.719293	0.081560
std	129.716773	0.240656	6.663051	478.954251	5.361120	5.518501	5.701417	5.392273	0.395631
min	1.000000	0.000000	-2.000000	0.010603	10.021329	40.188585	41.860309	21.589952	0.000000
25%	99.000000	0.000000	5.000000	270.728276	12.318893	50.208231	52.659107	34.779167	0.000000
50%	210.000000	0.000000	10.000000	440.309314	18.886098	53.157792	55.676294	38.474893	0.000000
75%	308.000000	0.000000	15.000000	764.052515	20.475774	56.575462	59.412532	42.268017	0.000000
max	479.000000	1.000000	35.000000	2908.814992	38.974796	78.220308	81.260332	55.025626	2.000000

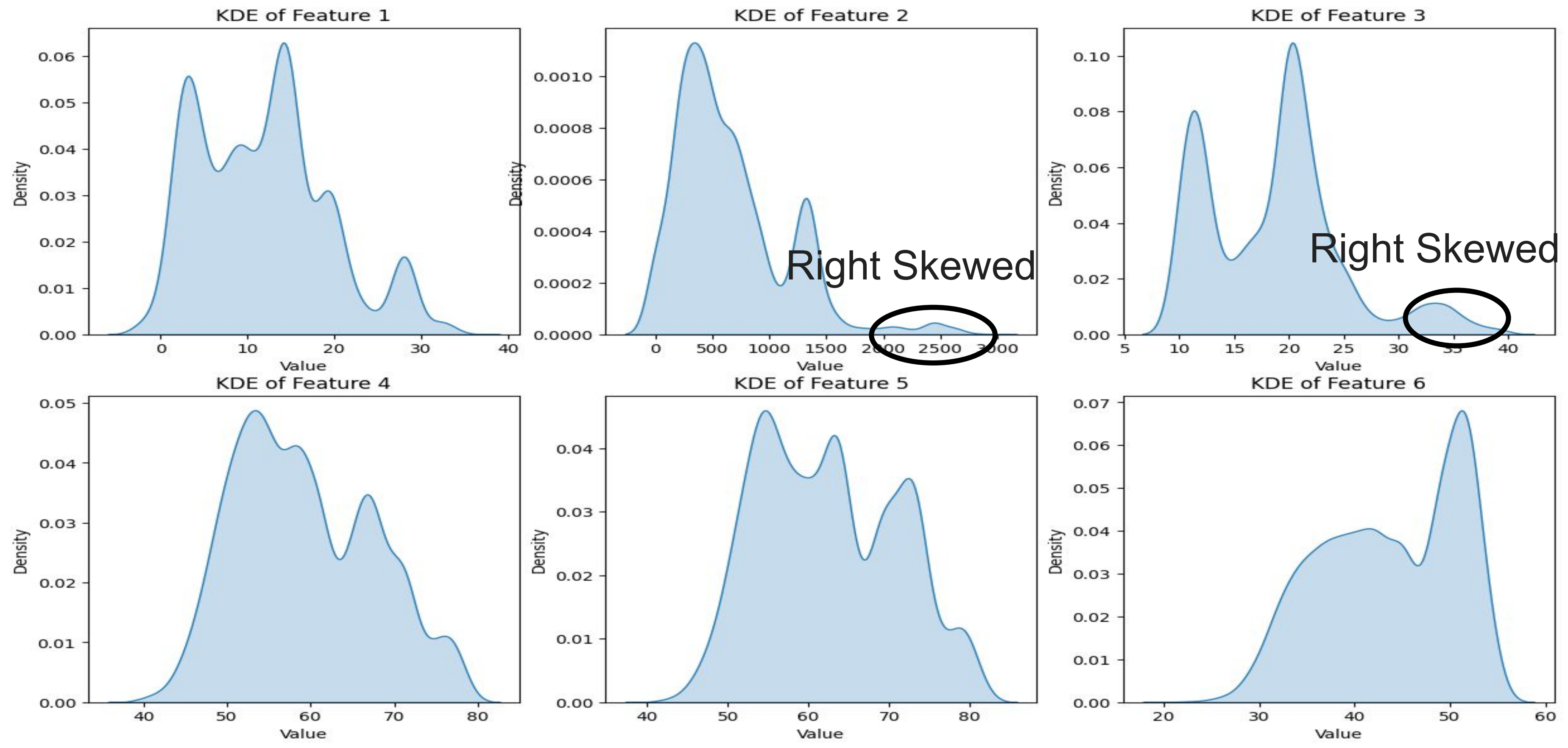
Data Description after upsampling

	WTGID	Loc	f1	f2	f3	f4	f5	f6	target
count	5410.000000	5410.000000	5410.000000	5410.000000	5410.000000	5410.000000	5410.000000	5410.000000	5410.000000
mean	208.182810	0.118669	11.855083	656.653795	18.703285	59.262975	62.383708	43.680562	1.000000
std	132.328112	0.323429	7.569758	489.235102	6.290800	8.257017	8.539835	7.079110	1.000092
min	1.000000	0.000000	-2.000000	0.010603	10.021329	40.188585	41.860309	21.589952	0.000000
25%	77.000000	0.000000	5.000000	291.433854	12.296072	52.728338	55.037849	37.862391	0.000000
50%	228.000000	0.000000	12.000000	530.890185	19.994263	58.239950	62.155287	44.328693	1.000000
75%	301.000000	0.000000	16.000000	897.068449	21.814049	66.154815	69.467032	50.471797	2.000000
max	479.000000	1.000000	35.000000	2908.814992	38.974796	78.220308	81.260332	55.025626	2.000000

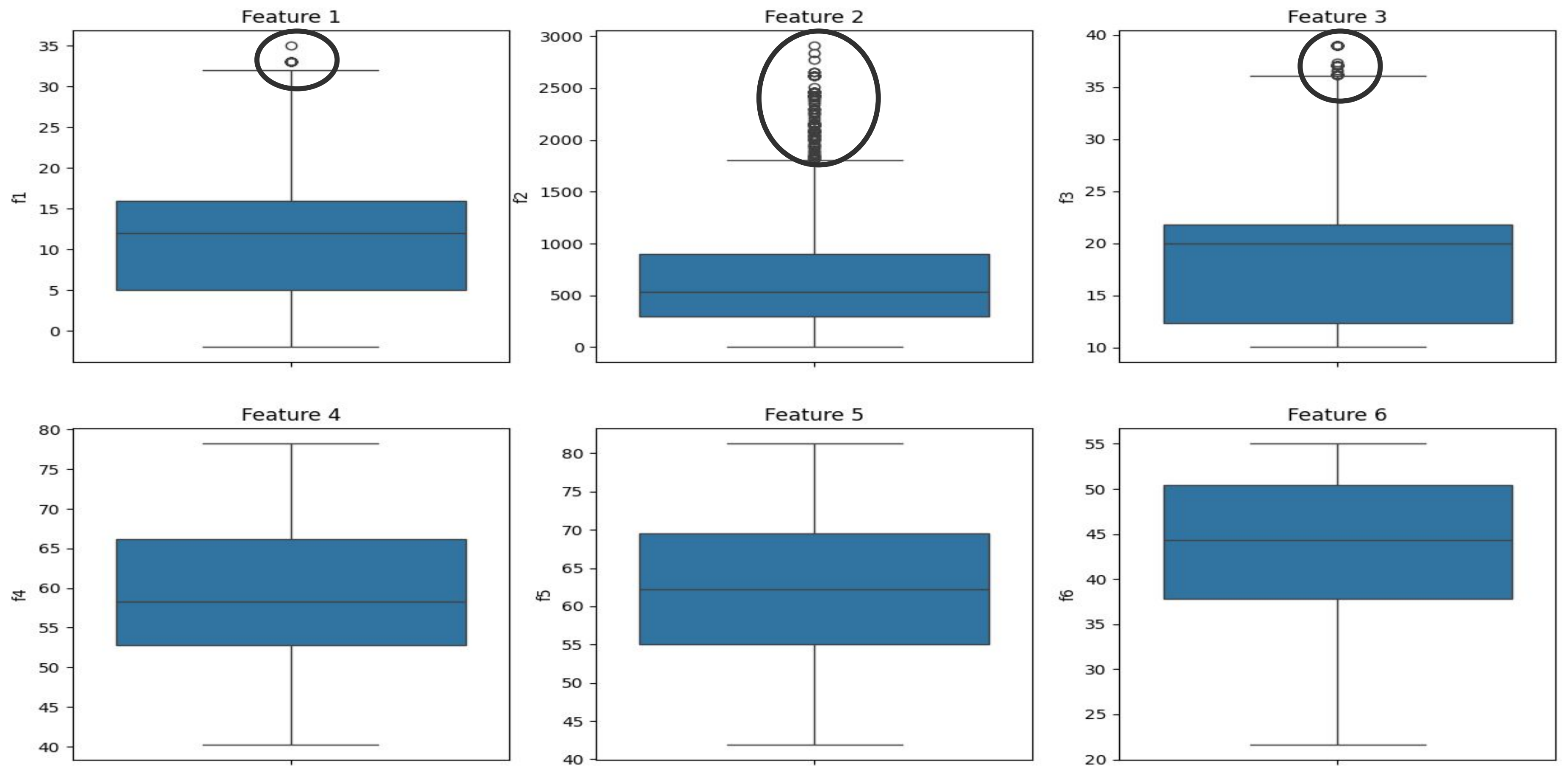
Histogram of Features (F1,F2,...,F6)



Distribution of Features (F1,F2,...,F6)



Outliers in Features (F1,F2,...,F6)

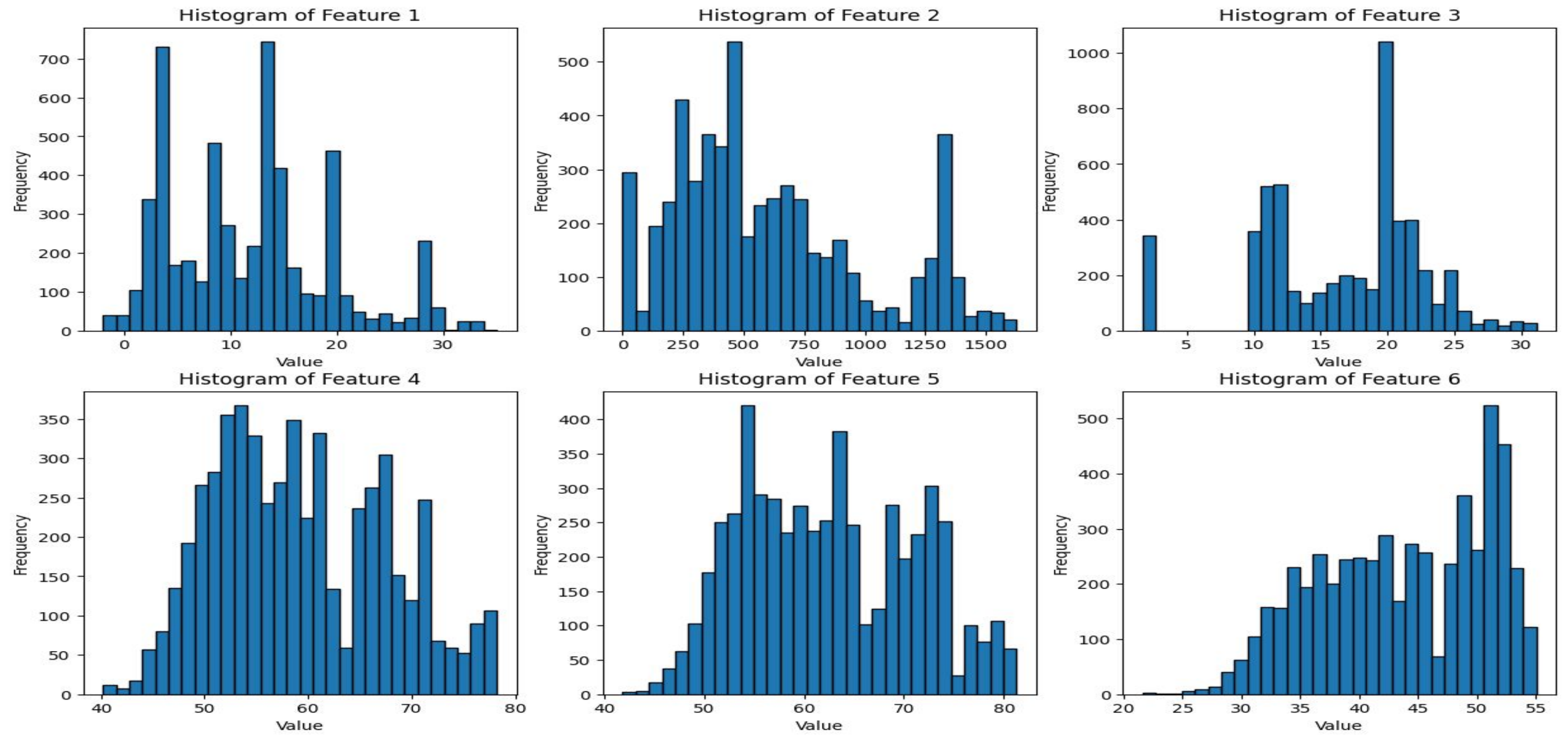


Treating Outliers in Data

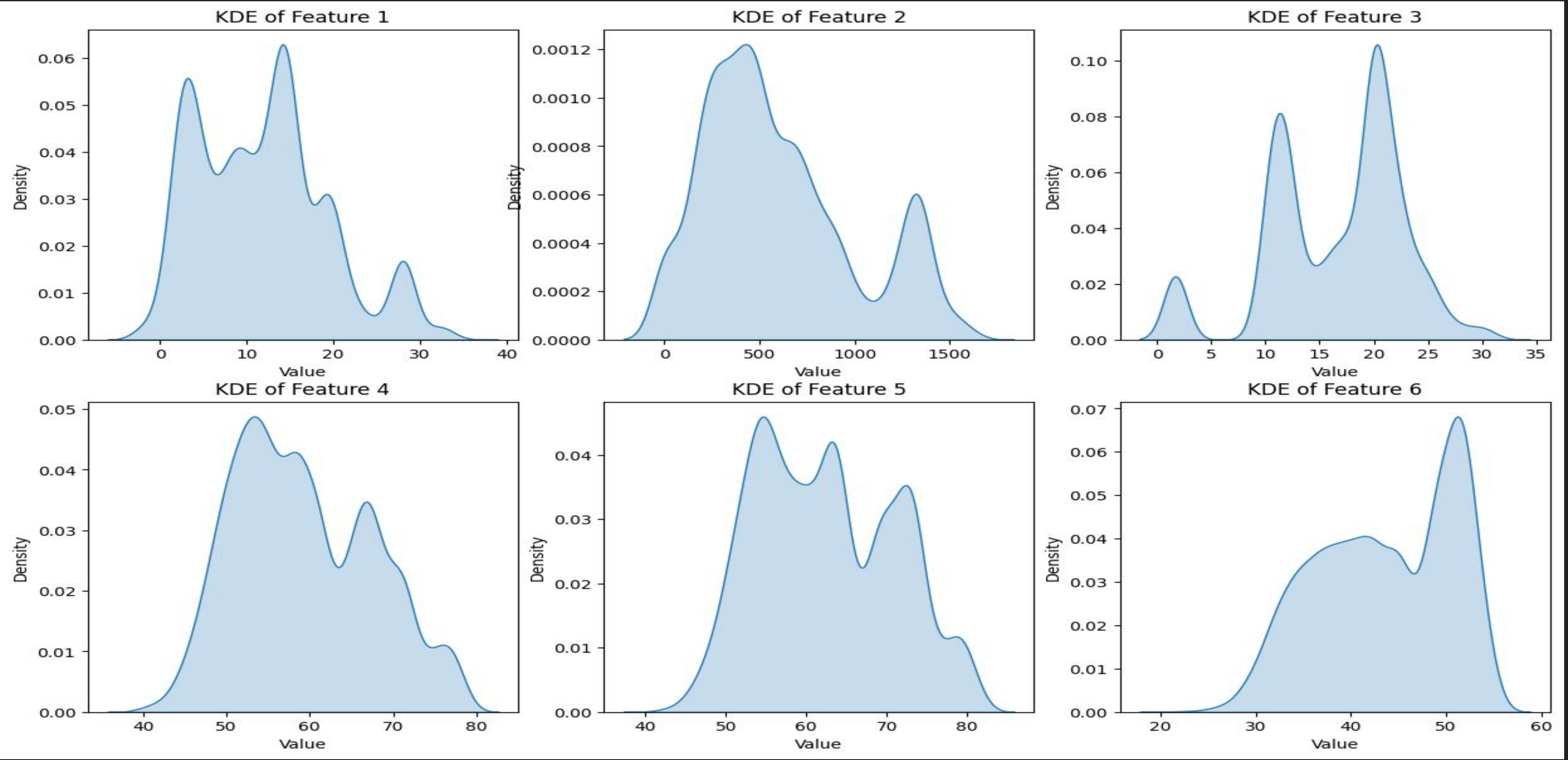
- ➔ Features “F1”,”F2”,”F3” has outliers
- ➔ The z-score threshold is -2 to +2
- ➔ Feature F2 outliers are handled by Median imputation : MEDIAN(F2)
- ➔ Feature F3 outliers are handled by : SQRT(LOG(MEAN(F3)))
- ➔ Dataset description after outlier removal

	WTGID	Loc	f1	f2	f3	f4	f5	f6	target
count	5410.000000	5410.000000	5410.000000	5410.000000	5410.000000	5410.000000	5410.000000	5410.000000	5410.000000
mean	208.182810	0.118669	11.855083	602.006487	16.655486	59.262975	62.383708	43.680562	1.000000
std	132.328112	0.323429	7.569758	396.556641	6.185019	8.257017	8.539835	7.079110	1.000092
min	1.000000	0.000000	-2.000000	0.010603	1.693489	40.188585	41.860309	21.589952	0.000000
25%	77.000000	0.000000	5.000000	291.433854	11.782187	52.728338	55.037849	37.862391	0.000000
50%	228.000000	0.000000	12.000000	487.336491	18.535453	58.239950	62.155287	44.328693	1.000000
75%	301.000000	0.000000	16.000000	823.826058	20.790087	66.154815	69.467032	50.471797	2.000000
max	479.000000	1.000000	35.000000	1627.081587	31.125399	78.220308	81.260332	55.025626	2.000000

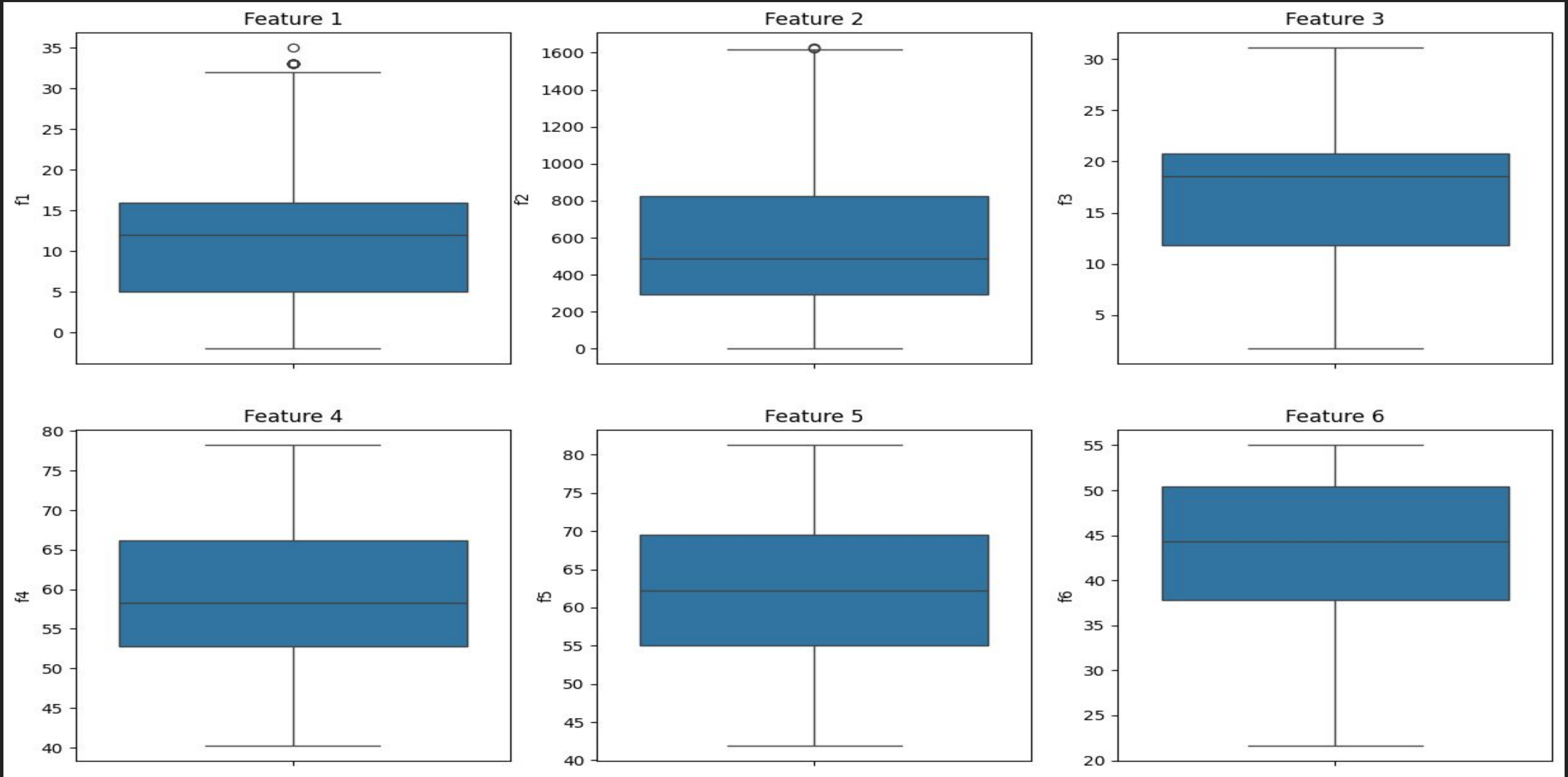
Distribution of Dataset - Post outlier removal



Data Distribution - Post outlier removal



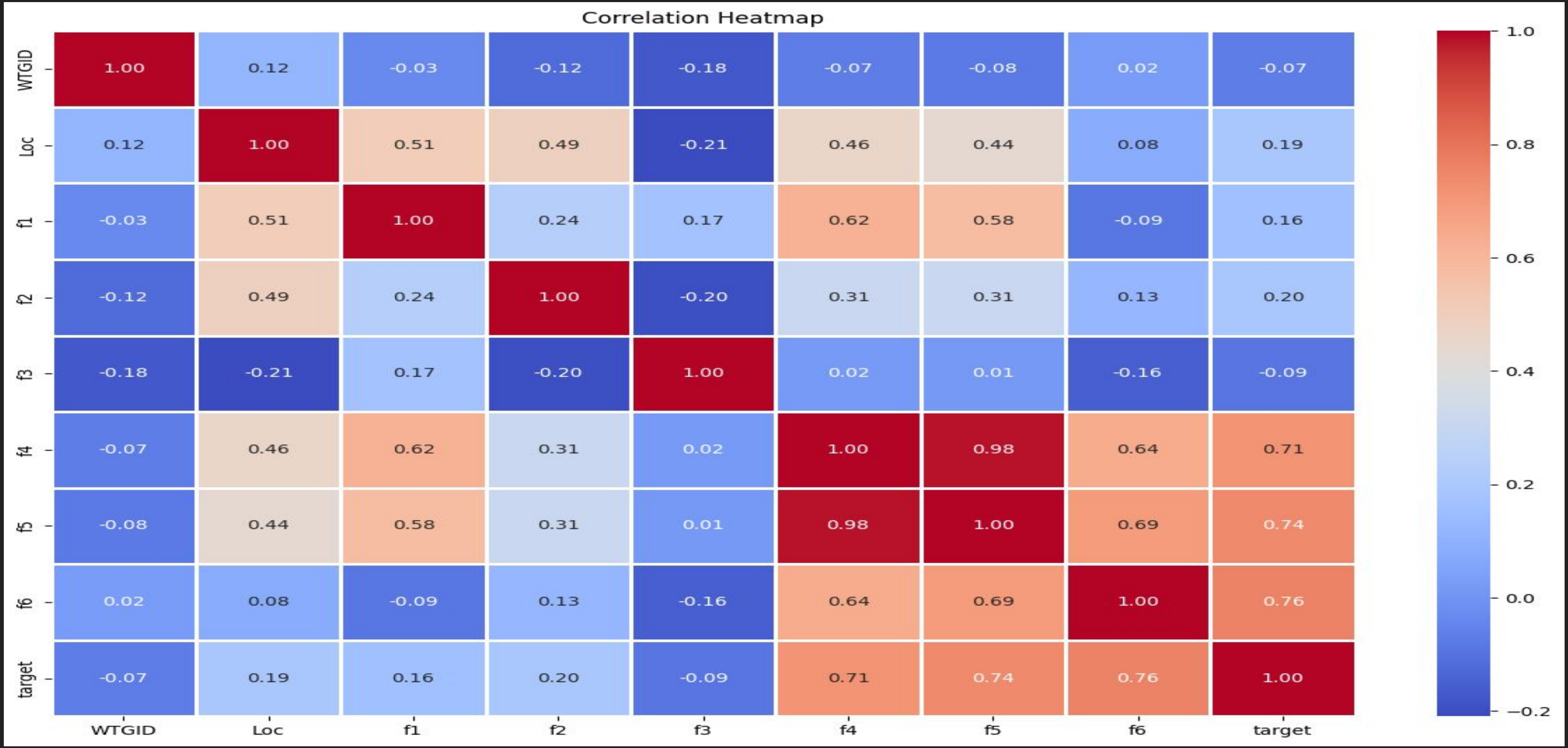
Reduced outlier Dataset



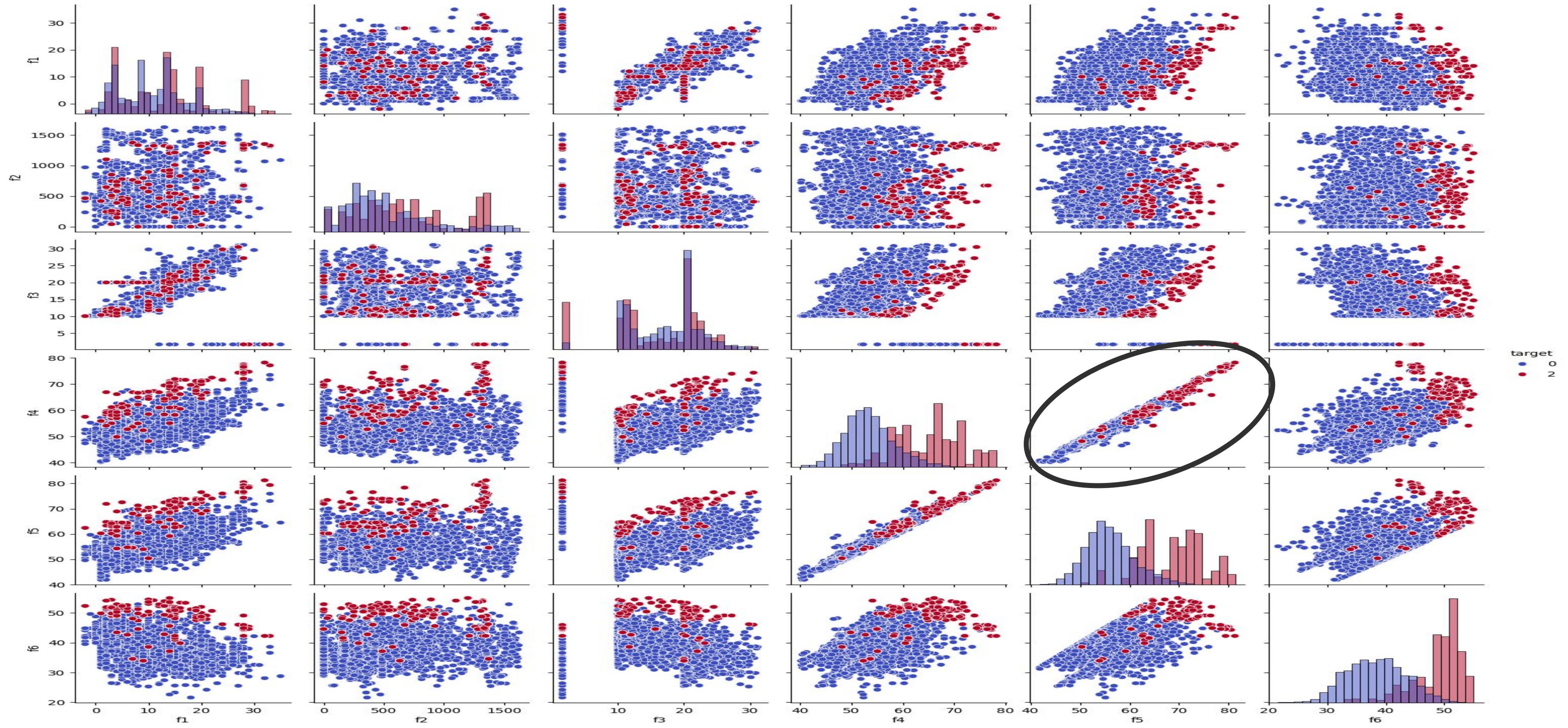
Correlation between Features - post upsampling and outlier treatment

	WTGID	Loc	f1	f2	f3	f4	f5	f6	target
WTGID	1.000000	0.119779	-0.033663	-0.122961	-0.175851	-0.068492	-0.079945	0.024432	-0.069258
Loc	0.119779	1.000000	0.505714	0.488126	-0.209235	0.458206	0.436415	0.081978	0.194332
f1	-0.033663	0.505714	1.000000	0.238537	0.169328	0.620584	0.579564	-0.088441	0.163376
f2	-0.122961	0.488126	0.238537	1.000000	-0.198658	0.310273	0.306571	0.125533	0.196515
f3	-0.175851	-0.209235	0.169328	-0.198658	1.000000	0.017526	0.014959	-0.155739	-0.093408
f4	-0.068492	0.458206	0.620584	0.310273	0.017526	1.000000	0.984306	0.638663	0.714141
f5	-0.079945	0.436415	0.579564	0.306571	0.014959	0.984306	1.000000	0.690346	0.744206
f6	0.024432	0.081978	-0.088441	0.125533	-0.155739	0.638663	0.690346	1.000000	0.762649
target	-0.069258	0.194332	0.163376	0.196515	-0.093408	0.714141	0.744206	0.762649	1.000000

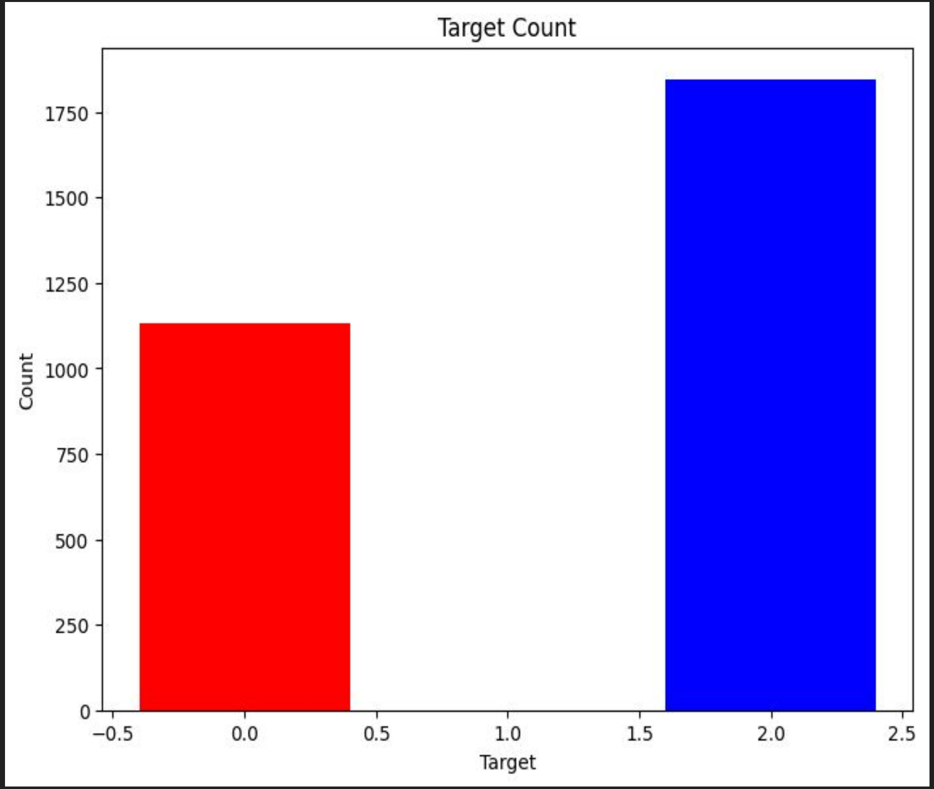
Correlation between Features - post upsampling and outlier treatment



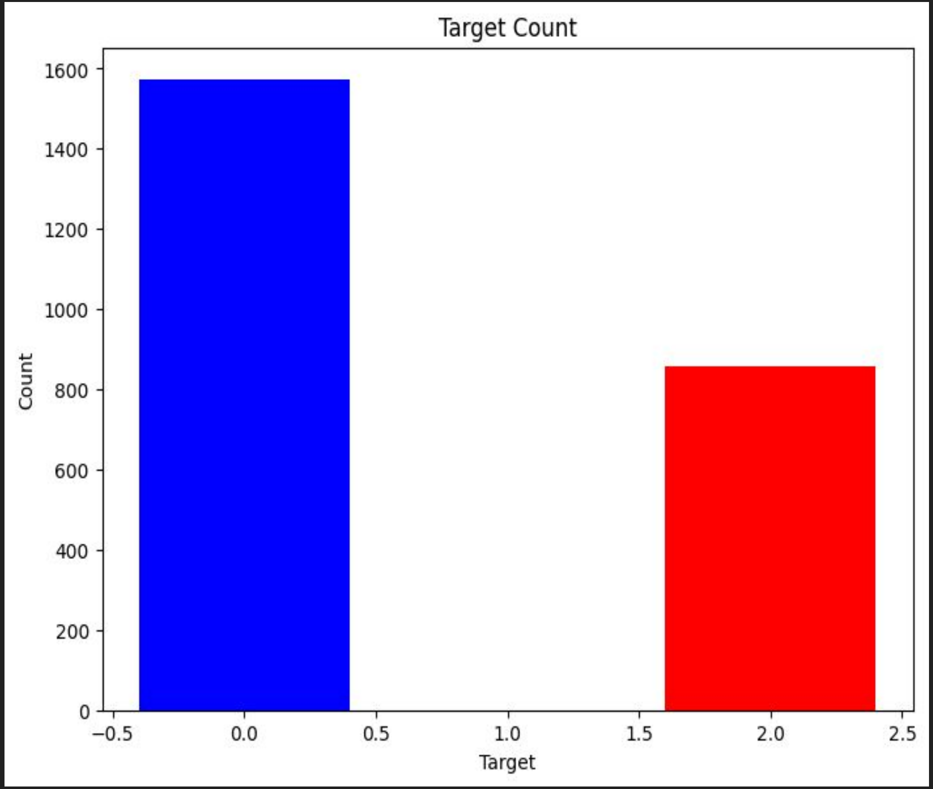
Correlation between features (pairwise) - post upsampling and outlier treatment



Training Data Target Count



Validation Data Target Count



MODEL SELECTION

Different Classification Models

- → Simple Logistic Regression
- → SVM Classifier
- → KNN Classifier
- → Decision Tree Classifier

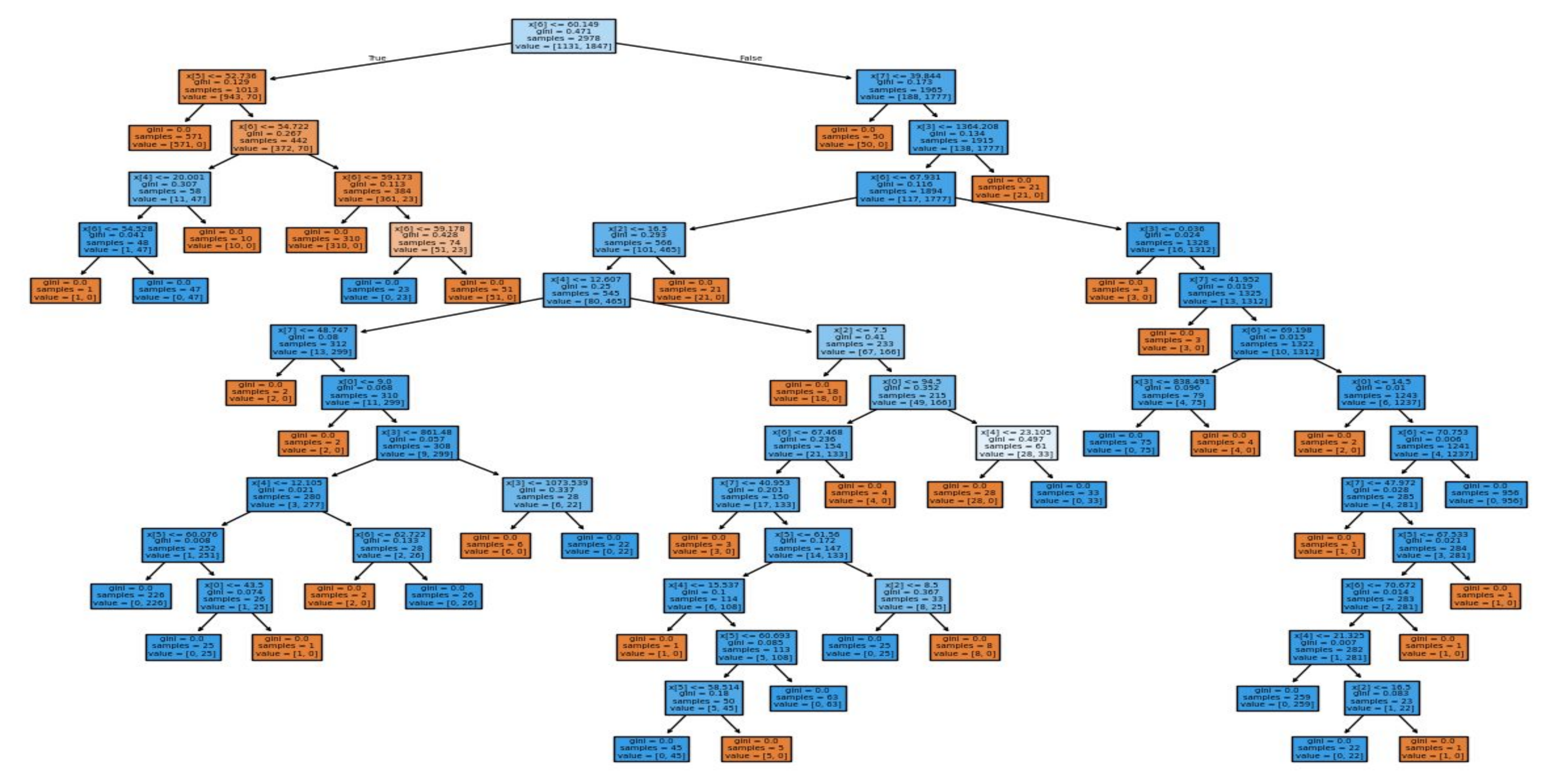
<u>Ensemble Techniques:</u>

- → Random Forest Classifier (Bagging)
- → AdaBoost Classifier (Boosting)
- → Gradient Boost Classifier (Boosting)
- → XGBoost Classifier (Boosting)

Why Gradient Boosting or XGBoost Classifier?

- ➔ Handles mixed data well :
  Works well with categorical and numerical features well
- ➔ High Performance :
  Generally outperforms logistic regression, decision trees, and even random forest for structured data
- ➔ Automatically Handles Feature Interactions :
  Captures non-linear relationships between data
- ➔ Feature Importance :
  Easy to extract importance scores for feature selection or model interpretation
- ➔ Robust to Multicollinearity and Missing Data :
  XGBoost can handle missing values natively
- ➔ Fast and Scalable :
  Optimized for speed and memory; scalable to larger datasets

Important Features in Target Classification



Evaluation Metrics

- Accuracy
- F1 Score
- Precision
- Recall
- ROC AUC Score

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

EVALUATION METRICS

Gradient Boost Classifier

Before Hyper parameter Tuning	After Hyper parameter Tuning
GradientBoost	GradientBoost
Model performance for Training set	Model performance for Training set
- Accuracy: 0.9940	- Accuracy: 1.0000
- F1 Score: 0.9939	- F1 score: 1.0000
- Precision: 0.9903	- Precision: 1.0000
- Recall: 1.0000	- Recall: 1.0000
- ROC AUC Score: 0.9920	- Roc Auc Score: 1.0000
----------------------------------	----------------------------------
Model performance for Test set	Model performance for Test set
- Accuracy: 0.9342	- Accuracy: 0.8771
- F1 Score: 0.9336	- F1 score: 0.8714
- Precision: 0.9373	- Precision: 0.9530
- Recall: 0.8718	- Recall: 0.6853
- ROC AUC Score: 0.9200	- Roc Auc Score: 0.8334
===================================	===================================

Hyper Parameter Tuning for GB Classifier

- ➔ Loss : The loss function to be optimized. 'log_loss' refers to binomial and multinomial deviance, the same as used in logistic regression
- ➔ Criterion : The function to measure the quality of a split
- ➔ Min_samples_split : The minimum number of samples required to split an internal node
- ➔ Max_Depth : maximum depth of the Decision Tree
- ➔ N_estimators : the number of boosting rounds or the number of gradient-boosted trees in the model

```
{'loss': ['log_loss', 'deviance', 'exponential'],
 'criterion': ['friedman_mse', 'squared_error', 'mse'],
 'min_samples_split': [2, 8, 15, 20],
 'n_estimators': [100, 200, 500, 1000],
 'max_depth': [5, 8, 15, None, 10]}
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
--------------- Best Params for GB ------------------
{'n_estimators': 100, 'min_samples_split': 20, 'max_depth': 8, 'loss': 'exponential', 'criterion': 'squared_error'}
```

ML Model Comparisons

S.NO	MODEL NAME	ACCURACY	F1-SCORE	PRECISION	RECALL	ROC_AUC SCORE
1	LOGISTIC REG	0.63	0.64	0.74	0.63	0.95
2	LOGIC REG (HT)	0.90	0.90	0.91	0.90	0.95
3	SVM	0.76	0.76	0.75	0.76	-
4	SVM (HT)	0.66	0.53	0.78	0.66	-
5	KNN	0.68	0.62	0.67	0.68	-
6	KNN(HT)	0.72	0.67	0.78	0.72	-
7	DT	0.84	0.82	0.85	0.84	-
8	DT (HT)	0.92	0.92	0.92	0.92	-
9	RF	0.86	0.85	0.97	0.62	0.80
10	RF (HT)	0.88	0.87	0.97	0.69	0.84

ML Model Comparisons

S.NO	MODEL NAME	ACCURACY	F1-SCORE	PRECISION	RECALL	ROC_AUC SCORE
11	AdaBoost	0.90	0.90	0.83	0.91	0.90
12	AdaBoost (HT)	0.83	0.84	0.71	0.91	0.85
13	Gradient Boost	0.93	0.93	0.93	0.87	0.92
14	Gradient Boost(HT)	0.87	0.87	0.95	0.68	0.83
15	XGBoost	0.89	0.88	0.95	0.73	0.94
16	XGBoost (HT)	0.91	0.91	0.96	0.79	0.88

Observation:

- ➔ Boosting performs slightly better than Bagging ensemble Technique
- ➔ Gradient Boost has the best accuracy among all the classifier models

THANK YOU…