Kaggle Competition Report:

Name: Rupali Roy
UTID: rur73

0.91583 (Public leaderboard) and 0.91504 (Private leaderboard)

Link : https://www.kaggle.com/c/data-science-lab-kaggle-sp2020

I followed the following steps for the Binary Classification Problem

1. Exploratory Data Analysis
2. Data Cleaning
3. Feature Engineering
4. Algorithm Selection
5. Model Training and Tuning
6. Insights/Conclusion

1. Exploratory Data Analysis

Exploratory Data Analysis is the process in which we perform initial investigations on the data, to discover patterns, spot anomalies,test hypothesis and check assumptions with the help of summary statistics.

Its main purpose is to “get to know the data” . It serves following two important purpose:

- Helps us gain valuable hints for “Data Cleaning”
- And to think of ideas for “Feature Engineering”

Important inferences from the exploratory data analysis step is:

1. Number of Observations present in the dataset?
2. How many features?
3. What are the data types of different features?Are they numeric or categorical?
4. What is the target variable?

Train and Test data can be found at following URL:

```
train_url = "https://raw.githubusercontent.com/rroy1212/DSL_Kaggle_Competition/master/train_final.csv"
```

```
test_url = "https://raw.githubusercontent.com/rroy1212/DSL_Kaggle_Competition/master/test_final.csv"
```

Using df.shape we can get details about the number of rows and columns that is present in train and test dataset.

The training set has: 16383 rows and 26 columns
The training set has: 16385 rows and 25 columns

```
[ ] test_df.head()
```

	Id	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20	f21	f22	f23	f24
0	16384	37733	1	1.77	118603	1	0	118602	118097	1	0	1	1	117888	2.453740	13881	117941	117887	1	117885	1	1	1	1	1
1	16385	312129	1	3.54	118052	1	0	117961	290919	1	4	1	43	118322	-0.012317	14638	118992	290919	1	118321	1	1	1	7	1
2	16386	24884	1	23.01	118300	1	0	117961	302830	1	0	1	1	128231	1.000000	770	119181	4673	1	128230	1	1	1	14	1
3	16387	4674	1	1.77	119091	1	0	119062	118036	1	9	1	1	117908	1.000000	16752	143531	290919	1	117905	1	1	1	81	1
4	16388	68725	1	3.54	118300	1	0	117961	171056	1	0	1	6	118639	-0.503250	4945	118360	118638	1	118636	1	1	1	1	1

```
[ ] train_df.head()
```

	Id	Y	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20	f21	f22	f23	f24
0	1	1	25884	1	33.63	118596	1	0	118595	125738	1	3	1	2	121374	-2.266430	1945	118450	119184	1	121372	1	1	1	2	1
1	2	1	34346	1	10.62	118041	1	0	117902	130913	1	1	1	23	118943	-0.305612	15385	117945	292795	1	259173	1	1	1	1	1
2	3	1	34923	1	1.77	118327	1	0	117961	124402	1	2	1	1	118786	2.015561	7547	118933	290919	1	118784	1	1	1	1	1
3	4	1	80926	1	30.09	118300	1	0	117961	301218	1	0	1	1	118332	-3.172501	4933	118458	118331	1	307024	1	1	1	2	1
4	5	1	4674	1	1.77	119921	1	0	119920	302830	1	0	1	2	128231	0.573767	13836	142145	4673	1	128230	1	1	1	620	1

df.info() gives details about the type pf data present in the dataset. We observe that we do not have any categorical variables, all the variables are pf either integer or float data type. ‘Y’ is the target variable.

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16383 entries, 0 to 16382
Data columns (total 26 columns):
Id     16383 non-null int64
Y      16383 non-null int64
f1     16383 non-null int64
f2     16383 non-null int64
f3     16383 non-null float64
f4     16383 non-null int64
f5     16383 non-null int64
f6     16383 non-null int64
f7     16383 non-null int64
f8     16383 non-null int64
f9     16383 non-null int64
f10    16383 non-null int64
f11    16383 non-null int64
f12    16383 non-null int64
f13    16383 non-null int64
f14    16383 non-null float64
f15    16383 non-null int64
f16    16383 non-null int64
f17    16383 non-null int64
f18    16383 non-null int64
f19    16383 non-null int64
f20    16383 non-null int64
f21    16383 non-null int64
f22    16383 non-null int64
f23    16383 non-null int64
f24    16383 non-null int64
dtypes: float64(2), int64(24)
memory usage: 3.2 MB
```

Missing Values:

Next, we check if we have any missing values present in the dataset. df.isnull().sum() is used to get any missing values.

```
# check null values
train_df.isnull().sum()
```

```
Id     0
Y      0
f1     0
f2     0
f3     0
f4     0
f5     0
f6     0
f7     0
f8     0
f9     0
f10    0
f11    0
f12    0
f13    0
f14    0
f15    0
f16    0
f17    0
f18    0
f19    0
f20    0
f21    0
f22    0
f23    0
f24    0
dtype: int64
```

```
# check null values
test_df.isnull().sum()
```

```
Id     0
f1     0
f2     0
f3     0
f4     0
f5     0
f6     0
f7     0
f8     0
f9     0
f10    0
f11    0
f12    0
f13    0
f14    0
f15    0
f16    0
f17    0
f18    0
f19    0
f20    0
f21    0
f22    0
f23    0
f24    0
dtype: int64
```

We notice that we do not have any missing data in the given dataset. In case we had missing data then, we might have to handle it using various imputation techniques.

Many real-world datasets may contain missing values for various reasons. They are often encoded as NaNs, blanks or any other placeholders. Training a model with a dataset that has a lot of missing values can drastically impact the machine learning model's quality. Some algorithms such as *scikit-learn estimators* assume that all values are numerical and have and hold meaningful value.

There are three main types of missing data:

- Missing completely at random (MCAR)
- Missing at random (MAR)
- Not missing at random (NMAR)

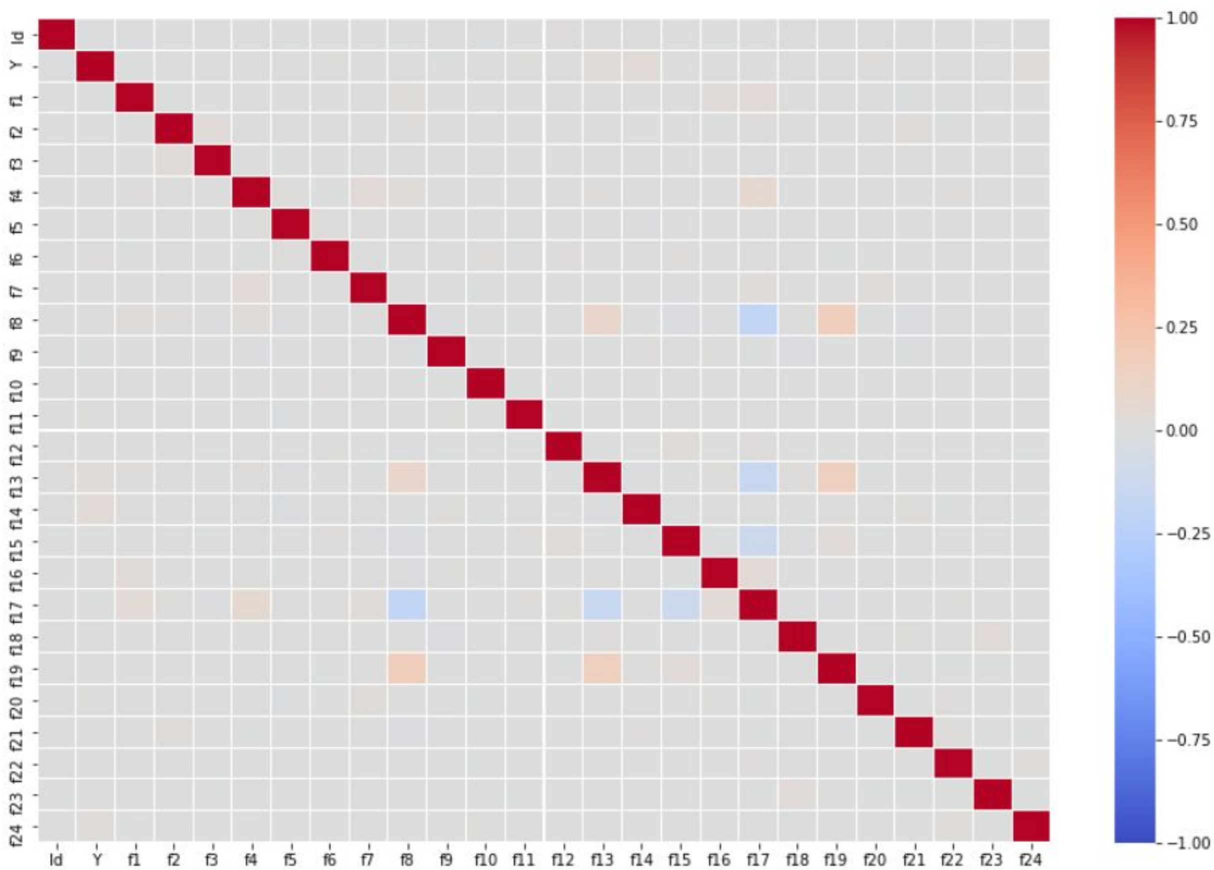
Different ways to handle missing data:

- Dropping observations that have missing values
- Imputing the missing values based on other observations
- Interpolation and Extrapolation
- Using KNN
- Mean/ Median Imputation
- Regression Imputation
- Stochastic regression imputation
- Hot-deck imputation

Correlation:

Correlations allow you to look at the relationships between numeric features and other numeric features. It is a value between -1 and 1 that represents how closely two features move in unison. It has following intuition:

1. **Positive** correlation means that as one feature increases, the other increases. E.g. a child's age and her height.
2. **Negative** correlation means that as one feature increases, the other decreases. E.g. hours spent studying and a number of parties attended.
3. Correlations near -1 or 1 indicate a **strong relationship**.
4. Those closer to 0 indicate a **weak relationship**.
5. 0 indicates **no relationship**



If we find correlation between two variables/features then we may perform any of the following step:

- Remove some of the highly correlated independent variables.
- Linearly combine the independent variables, such as adding them together.
- Principal components analysis

In our case, we can conclude from the above plot that no two columns are highly correlated.

Data Distribution:

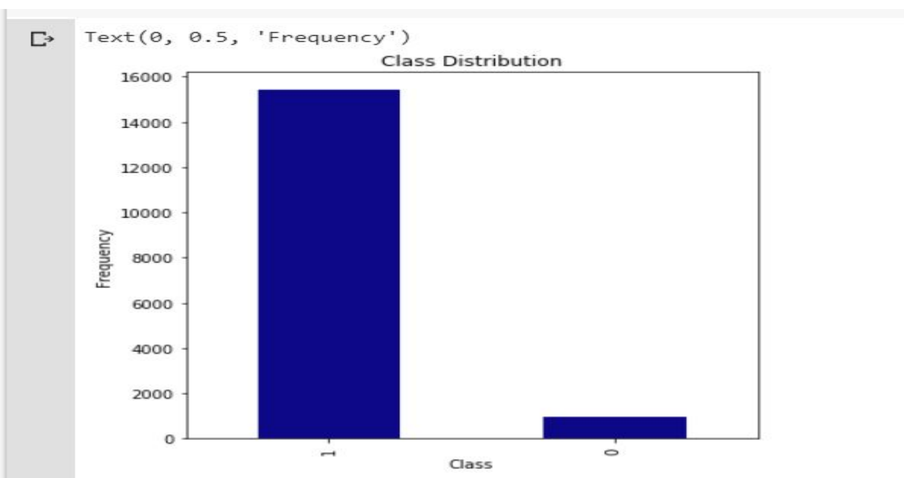
In this step we observe that distribution of all the features with respect to target variable.

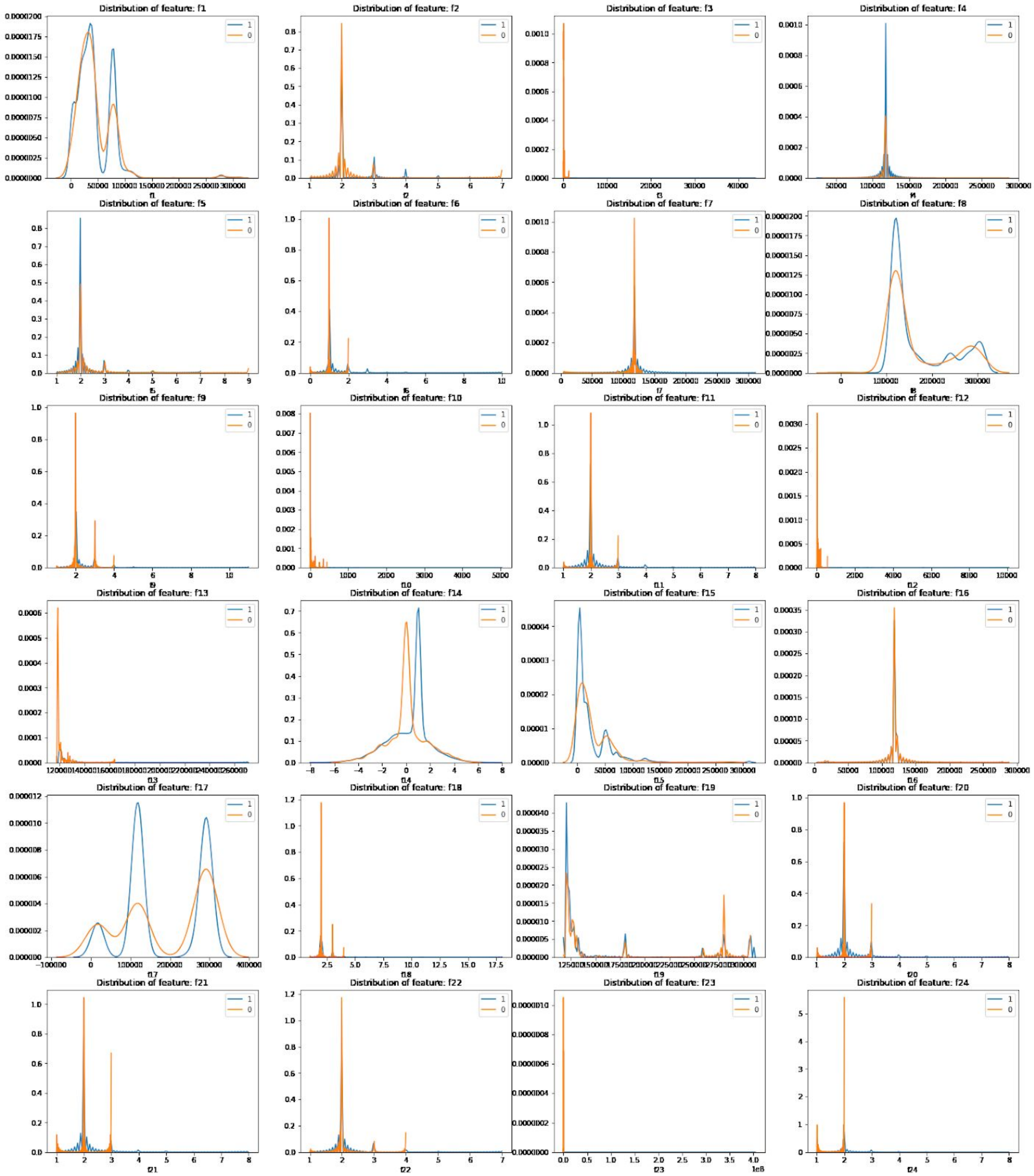
From the below plot we observe that the features f24,f22,f21,f20,f9,f18,f9,f6,f5,f11 have almost same distribution for both class = 0 and class = 1. Therefore, the classification algorithm might face difficulty to separate 1's and 0's from these features.

We also check the data distribution with respect to target variables. As shown below:

```
Number of records that belong to class 0: 948

Number of records that belong to class 1: 15435
```

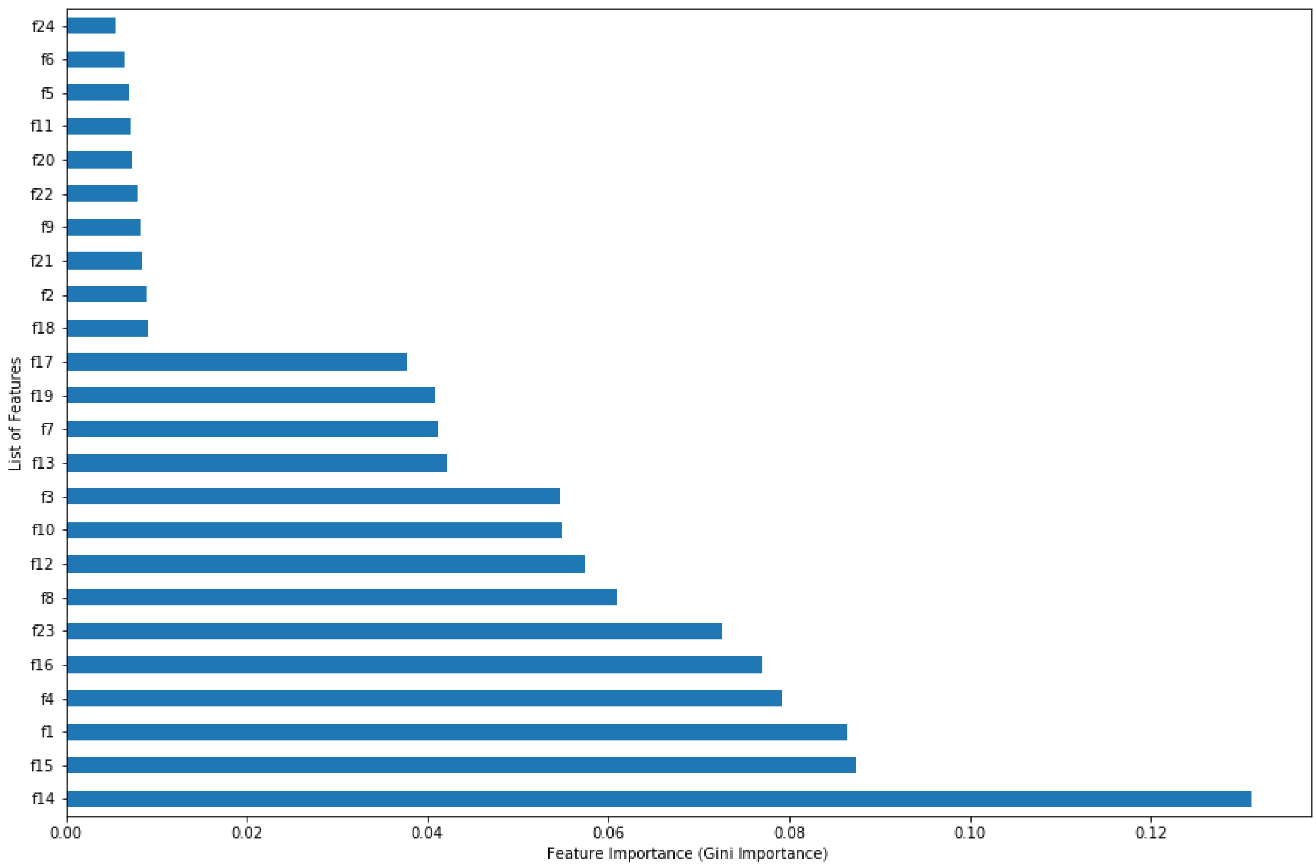



Distribution of feature: f1
Distribution of feature: f2
Distribution of feature: f3
Distribution of feature: f4
Distribution of feature: f5
Distribution of feature: f6
Distribution of feature: f7
Distribution of feature: f8
Distribution of feature: f9
Distribution of feature: f10
Distribution of feature: f11
Distribution of feature: f12
Distribution of feature: f13
Distribution of feature: f14
Distribution of feature: f15
Distribution of feature: f16
Distribution of feature: f17
Distribution of feature: f18
Distribution of feature: f19
Distribution of feature: f20
Distribution of feature: f21
Distribution of feature: f22
Distribution of feature: f23
Distribution of feature: f24

Feature Selection:

I have used ExtraTreesClassifier to get feature importance. From the below feature importance plot we can observe that these 10 features have least values for gini importance: f24,f5,f6,f11,f20,f22,f9,f2,f18,f21

```
# so we can try removing it from the test and train sets
```



Outliers Detection:

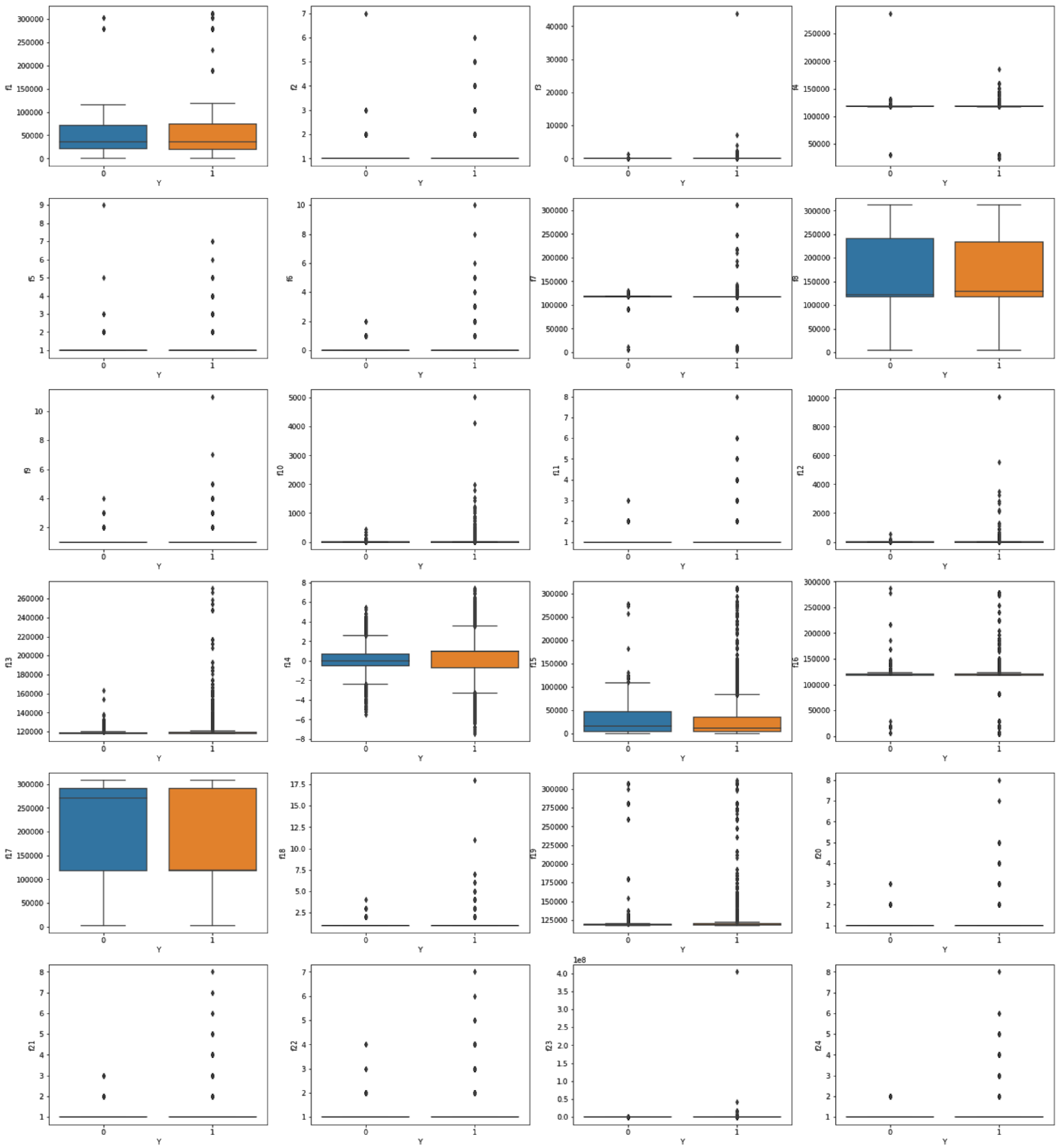
Detection methods:

- Scatter plot
- IQR
- Boxplot
- Extreme Value Analysis
- Z-score method
- K Means clustering-based approach
- Visualizing the data

Outlier Treatment Methods:

- Mean/Median or random Imputation
- Trimming
- Top, Bottom and Zero Coding
- Discretization

I have used boxplot method . Dataset had lot of outliers .Removed them using z-score: § 99.6% data is within 3 standard deviation . For values outside 3 standard deviation are considered outliers . After I used z-score method, all the zeroes were removed from the dataset which means all the zeroes were outliers . Therefore, we cannot remove the outliers. THIS IS AN ANOMALY DETECTION PROBLEM .



f1
f2
f3
f4
f5
f6
f7
f8
f9
f10
f11
f12
f13
f14
f15
f16
f17
f18
f19
f20
f21
f22
f23
f24
1e8
Y
0
1

Model Selection:

Things I tried-

My approach was to do a trial run on various models listed below and then try to tune the model that was performing better amongst all.

Models that I tried:

1. Naive Bayes-GaussianNB

   ROC-AUC Value for Naive-Bayes: 0.564 (0.019)

   0.5638321086538236

2. Stochastic Gradient Descent Classifier

   ROC-AUC Value for SGDClassifier: 0.506 (0.043)
   0.5064947191913308

3. K- Nearest Neighbours

   ROC-AUC Value for KNN_Classifier: 0.650 (0.017)
   0.6496439369304

4. Logistic Regression

   ROC-AUC Value for Logistic Regression: 0.506 (0.038)

   0.5057835037932855

5. Gradient Boosting Classifier

   ROC-AUC Value for gradient_boost: 0.866 (0.020)

   0.8657215325717396

6. Decision Trees

   ROC-AUC Value for Decision_Tree: 0.795 (0.017)

0.7949129341434995

7. Random Forest Classifier

   ROC-AUC Value for Random_Forest: 0.838 (0.019)
   0.837695534741191
   After performing hyper parameter tuning using bayesian optimization for random forest, I obtained roc of : ROC-AUC Value for Random_Forest_Tuned: 0.873 (0.019)
   0.8732638912670329

8. Light GBM Classifier

   After hyper parameter tuning we get: ROC-AUC Value for LGB_Tuned: 0.880 (0.014)
   0.8803523342826285

9. XGB Classifier

   Initially as well, without doing any hyper parameter tuning the XGB performed best amongst all the models. After tuning the XGB I could get the cross validation value of 0.9041665596893562. ROC-AUC Value for XGB: 0.904 (0.015) - 0.9041665596893562

10. Stacking:

    ROC-AUC Value for Stacked_XGB_LGB: 0.899 (0.015)

    0.8993940713756263

I spent a lot of time tuning a single XGB Classifier and getting the parameters that helped me get the best possible AUC score.

Things I tried that didn't work well.

- I tried all these algorithms, without doing any feature selection, but could not gain higher AUC.
- I tried to stack tuned XGB and LGB but it didn't work well on the leaderboard.

  ROC-AUC Value for Stacked_XGB_LGB: 0.899 (0.015)

  Cross val stack : 0.8993940713756263

  Leader board: 0.866

Conclusion:

XGBoost performed the best after hyperparameter tuning and cross validation with training AUC of 0.904 and testing score of 0.91583 (Public leaderboard) and 0.91504 (Private leaderboard)

Public Leader Board:

#	Team Name	Notebook	Team Members	Score	Entries	Last
1	Joseph Dean jd45664			0.95059	20	4d
2	Vidur Sinha			0.94288	27	2d
3	Brian Tsai			0.93101	13	3d
4	Sami Khandker			0.93074	34	1d
5	YashRLad			0.92212	5	1d
6	Aastha Goyal			0.91663	24	1d
7	Isabel J Li			0.91630	18	2d
8	erickli4224			0.91593	33	1d
9	Rupali Roy			0.91583	34	1d

Your Best Entry ↑

Private Leader Board:

#	△pub	Team Name	Notebook	Team Members	Score	Entries	Last
1	—	Joseph Dean jd45664			0.93998	20	4d
2	—	Vidur Sinha			0.93713	27	2d
3	—	Brian Tsai			0.92872	13	3d
4	—	Sami Khandker			0.92628	34	1d
5	—	YashRLad			0.92407	5	1d
6	—	Aastha Goyal			0.91834	24	1d
7	▲ 3	Harshitnainwani			0.91824	26	1d
8	▲ 1	Rupali Roy			0.91504	34	1d
9	▲ 13	Zander Tedjo			0.91492	31	1d

2. Data Cleaning

Check if

5. You’ll gain valuable hints for Data Cleaning (which can make or break your models).

6. You’ll think of ideas for Feature Engineering (which can take your models from good to great).