Business Analytics Report

Cost Optimization Modeling

Laura Lazarescou, John Rodgers, Maysam Mansor and Mel Schwan

April 12, 2021

Report Scope and Description

This business analytics report analyzes our business partner's reported metrics with the aim of predicting y-values for future reporting periods. This report is based on the data provided by our business partner, which included approximately 170,000 instances and 49 factors. The data included numerical, categorical, and string data types. The primary business objective of this study is to minimize total potential cost due to erroneous predictions.

Model Objective

This "Cost Optimization" report shows which machine learning model resulted in the minimal cost to the business. Our objective for the model is to minimize the dollars lost for every false positive classification of our target as well as the cost of false negative classifications. The target item "y" is positively predicted when the value is 1 and it is predicted to be 1. A negative classification has a value of 0 and is predicted to be 0.

When the model falsely identifies a positive record, the cost to the business is $225. Additionally, when the model falsely identifies a negative target, the business is charged $35. In each model simulation we found a varying balance of false positives and false negatives. Our challenge was to minimize the total number of errors or false predictions, while prioritizing false positives, since the cost impact was almost ten times the cost of a false negative.

Methodology

The machine learning pipeline activities consisted of:

- Data cleansing and normalization
- Exploratory data analysis (EDA)
- Model development
- Model tuning
- Prediction and comparison of model performance

Appendix A includes a complete description of the pipeline process. In this evaluation, given our primary objective is to minimize overall cost, we tuned each model to minimize false positive predictions.

Results

Three types of models were evaluated: Neural Network, XG Boost and Random Forest. These modeling techniques are some of the the most widely used in the analytics community, and are well-suited for large datasets with a number of diverse factors. In this case, we did not have access to a subject matter expert, so we were unable to eliminate factors based on business relevancy. However, the models were able to provide high-quality predictions.

Below in Table 1 is a summary of our tuned model performance. The best financial results were produced by the XGBoost model with the lowest overall cost of $214,280.

Technique	False Positive	False Negative	Total Cost
XGBoost	794	1018	$214,280
Neural Network	671	1859	$216,040
Random Forest	754	1462	$220,820

Model Results (Table 1)

Recommendations

All models may be useful in different circumstances. The XGBoost model provides approximately $2000 of cost savings over the Neural Network model. However, it took more than one hour to process, and the Neural Network model was created in approximately five minutes. Given the cost of labor, and the likelihood that source data will evolve and this model will need to be re-trained, we prefer the Neural Network approach.

It may also be worthwhile to leverage these models and build an ensemble model that could provide even greater cost reduction. The difference in performance between these models is modest, however, the total cost is still more than $200,000. However, if an ensemble model is created, we recommend it be scripted to run independently since the Random Forest model took three hours to train and would be cost prohibitive to maintain otherwise.

Finally, with assistance from subject matter experts, we may improve cost reduction by eliminating unnecessary factors. Simplifying the model frequently improves model performance. We are open to working with your organization to pursue any of these next steps.

Appendix A - Model Design and Development

Data Analytics Approach

1. Define the goal
2. Obtain the data
3. Clean and enrich the data
4. Create and optimize model(s)
5. Report and interpret results

1. Project Goal

Our objective for all models was to minimize dollars lost by minimizing mis-classification of predictions. In this case, the cost of mis-predicting a positive value was almost ten times greater than the cost to mis-predict a negative value. Therefore, our focus was on minimizing false positive results with an eye on false negatives to keep overall cost as low as possible.

When the model falsely identifies a record as a positive, our business is charged $225. When the model falsely identifies a negative target, our business is charged $35.

2. Data Description

The original dataset contained 160,000 records and 49 features with one column to predict = 'y'. This data was split into train and test datasets using an 80/20 split.

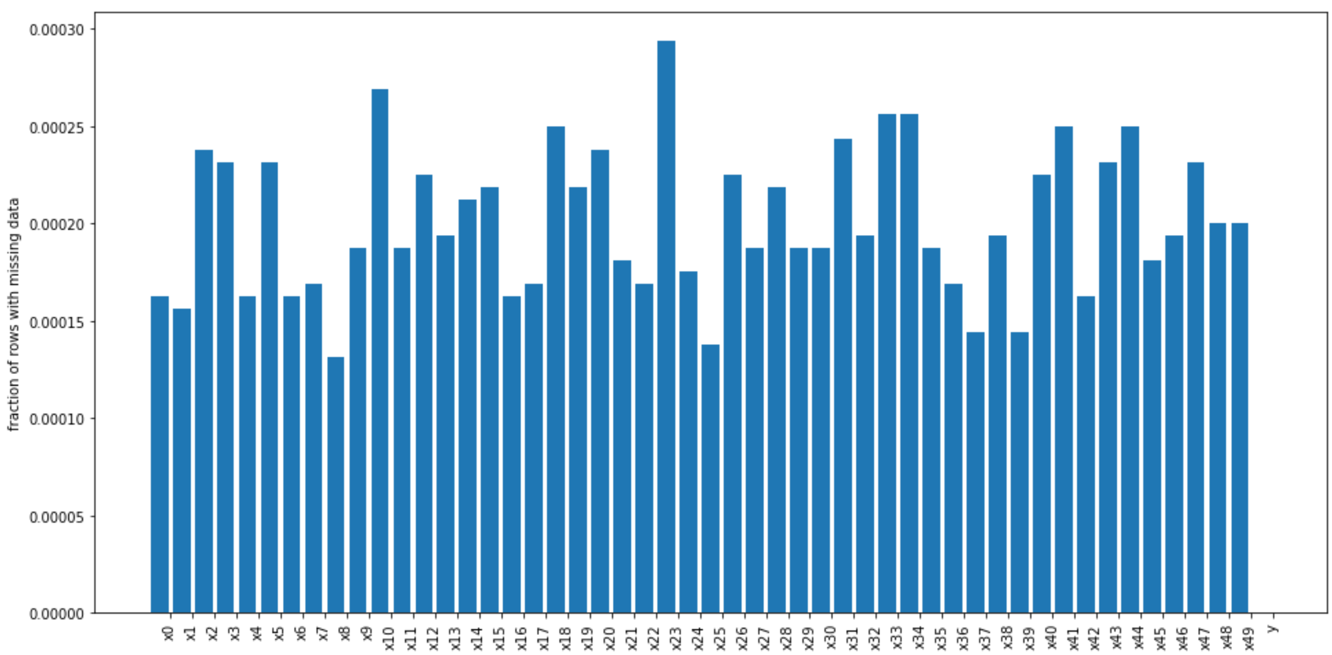
Attribute Information:

The last column, labeled "y" is the class label (1 for positive, 0 for negative). The other column labels are x0 - x49.
These are all float64 type except for columns x24, x29, x30, x32 and x37 which are object type.

3. Clean and Enrich Data Set

Our EDA began by looking at the types of features we would utilize in our models. We initially looked at the dataset shape and feature type (float64, object, ect.)

The next step was to assess any missing data and estimate the impact it may have on the model's accuracy. Figure 1 shows the percentage by feature of missing data. Every factor in the dataset contains some amount of missing data. The quantity of missing data is extremely low for each factor, however it is unusual to see missing data in every factor of a dataset. The labeled target "y" did not contain missing values.

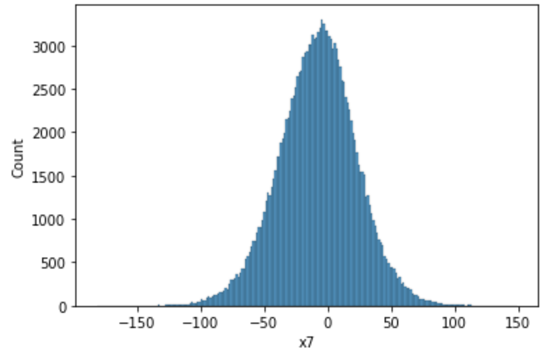


Missing Values by Feature (Figure 1)

Imputation

Imputation of Numerical Features

We found that most of the numerical data is normally distributed as shown in Figure 2 for feature "x7". With this knowledge we used the mean of each factor as the imputed value for any missing values.



Sample Histogram of Feature x7 (Figure 2)

Imputations of Categorical Features

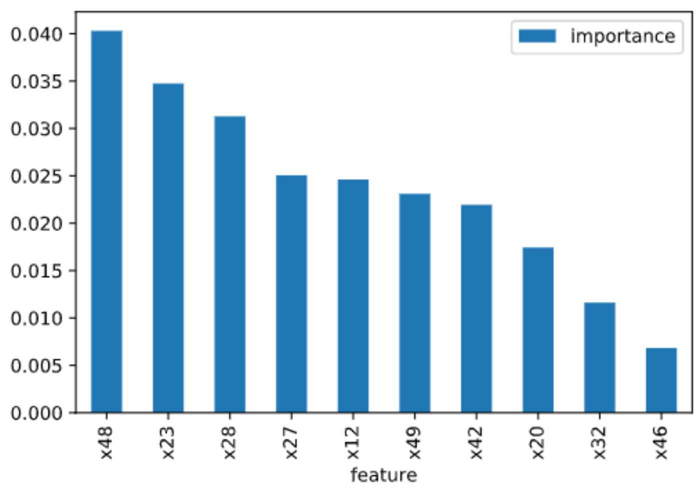
xFeatures x24, x29 and x30 are categorical. x24 is a list of continents x29 is a list of months x30 is a list of days of the week After analyzing these features and the small number of missing values we imputed all missing values to "Unknown".

Once the missing values for object-type factors were imputed, each factor was one-hot encoded which converted it into a number of binary factors indicating the presence or absence of that value.

Approximately twenty new features were created during the process of one-hot encoding. The final dataset that was used for modeling contained 160,000 observations or rows and 70 features or columns.

Feature Importance

After all the data was cleaned and enriched we ran our different models to optimize the cost to the business. We determined that XGBoost was the optimal model, based on minimal cost, so we explored feature importance using the permutation importance module of XGBoost. Figure 3 show the top ten features by their importance to the XGBoost model's results.



Feature Importance for XGBoost Model (Figure 3)

You can see that the x48 feature is the most impactful.

4. Create and Optimize Models

Tensor Flow Neural Network Model

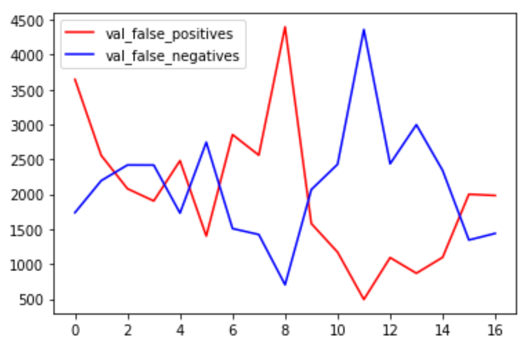
For the Neural Network model we began our study with a significant number of layers and nodes. With experimentation, we found our best model using three layers with 32, 12, 1 nodes per layer. Using more layers and nodes might cause overfitting or overwhelm the model and fewer layers and nodes might consume a lot more time and reduce accuracy. In the neural network hyperparameter tuning approach, we used FalsePositives with Early_Stopping to allow the model to continue through its process until it reached the optimal number of epochs with the lowest number of FalsePositives. Table 2 below shows the final optimization values through 40 epochs for the Neural Network model.

The time to process the neural network model for up to 40 epochs was less than ten minutes.

epoch	loss	accuracy	auc	false_positives	false_negatives	val_loss	val_accuracy	val_auc	val_false_positives	val_false_negatives	Results
37	0.202	0.924	0.9751	4671	5056	0.2135	0.9209	0.9752	671	1859	$216,040
17	0.274464	0.888211	0.952675	6399	7910	0.27974	0.8855	0.954064	958	2706	$310,260
18	0.269613	0.892297	0.954348	6189	7597	0.263502	0.895469	0.957101	1168	2177	$338,995
13	0.297633	0.876063	0.944121	7340	8524	0.300703	0.873812	0.945449	1205	2833	$370,280
11	0.308171	0.871414	0.939989	7878	8581	0.314136	0.868906	0.941766	1187	3008	$372,355
12	0.302265	0.874453	0.942299	7603	8467	0.311723	0.868187	0.94123	1302	2916	$395,010
9	0.316847	0.867016	0.936421	8247	8775	0.31603	0.869125	0.939826	1359	2829	$404,790
16	0.280138	0.88632	0.950641	6613	7938	0.275372	0.890031	0.952921	1611	1908	$429,255
19	0.26427	0.894367	0.956174	6091	7430	0.258374	0.895594	0.958845	1753	1588	$450,005
14	0.291101	0.880047	0.946596	7099	8255	0.283686	0.885156	0.949578	1776	1899	$466,065
15	0.286204	0.882516	0.948423	6880	8158	0.291025	0.881437	0.947788	2025	1769	$517,540
2	0.470354	0.777547	0.849051	11818	16656	0.407747	0.819531	0.895611	1687	4088	$522,655
3	0.392859	0.826688	0.899803	10015	12169	0.370274	0.839688	0.913807	1809	3321	$523,260
10	0.312022	0.86907	0.938427	8083	8676	0.308886	0.871406	0.939748	2106	2009	$544,165
6	0.336008	0.856734	0.928104	8763	9575	0.330614	0.859563	0.930423	2241	2253	$583,080
4	0.361071	0.844078	0.916358	9367	10591	0.352302	0.850719	0.922713	2838	1939	$706,415
5	0.345085	0.852047	0.923943	8962	9976	0.340445	0.853969	0.931267	3091	1582	$750,845
8	0.323067	0.863633	0.933779	8411	9044	0.335042	0.855687	0.936732	3289	1329	$786,540
0	0.625233	0.649609	0.678684	7051	37799	0.574295	0.699781	0.756851	2996	6611	$905,485
7	0.328687	0.861266	0.931351	8547	9211	0.368458	0.835375	0.935659	4328	940	$1,006,700
1	0.565584	0.708648	0.763694	14617	22676	0.553483	0.721969	0.796168	5790	3107	$1,411,495

Neural Network Optimized Results (Table 2)

Figure 4 below shows the false positive and negative count over the model epochs run.



Neural Network Optimize Results (Figure 4)

Extreme Gradient Boosting

For XGBoost, a base model was created using all of the default parameters. This model resulted in a predicted cost value of $1,077,240 on the full dataset as a result of 4,000 False Positives and 2,755 False Negatives.

In an effort to improve performance and develop a model that reduces the value of the cost for prediction errors, RandomizedSearchCV was used to fit 10 combinations of parameters from the options below. The best-performing model based on a 5-fold cross validation of the training dataset was measured based on the value of AUC. The XGBoost model also utilized AUC in evaluating the test dataset for early stopping.

'max_depth': [6, 10, 15, 20],

'learning_rate': [0.001, 0.01, 0.1, 0.2, 0.3],

'subsample': [0.5, 0.6, 0.7, 0.8, 0.9, 1.0],

'colsample_bytree': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],

'colsample_bylevel': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],

'min_child_weight': [0.5, 1.0, 3.0, 5.0, 7.0, 10.0],

'gamma': [0, 0.25, 0.5, 1.0],

'reg_lambda': [0.1, 1.0, 5.0, 10.0, 50.0, 100.0],

'n_estimators': [200]

The hypertuned model resulted in a reduction of $862,960 to the predicted error cost when compared to the untuned model when predicting the target for the full dataset. The predicted error cost of the hypertuned model is $214,280 as a result of 794 False Positives and 1018 False Negatives. The parameters associated with the best performing model, based on an AUC value of 0.9823 are:

'max_depth': 20,

'learning_rate': 0.3,

'subsample': 0.9,

'colsample_bytree': 0.8,

'colsample_bylevel': 0.5,

'min_child_weight': 3.0,

'gamma': 0,

'reg_lambda': 10.0,

'n_estimators': 200

The total time to process the XGBoost model through the 10 iterations of 5-fold cross-validation was approximately 1 hour.

Random Forest

The random forest model offers a good balance of processing speed and performance. In our base model, we used the default model parameters and ten (10) estimators. Without any hypertuning, this simple approach yielded a total cost value of $307,700. This cost was based on 939 False Positives and 2755 False Negatives.

To hypertune the random forest model, we used a randomized approach to selecting from a grid of parameters. The RandomizedSearchCV was called, choosing from the array of options below:

{'bootstrap': [True, False],

'max_depth': [10, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000],

'max_features': ['auto', 'sqrt'],

'min_samples_leaf': [1, 2, 4],

'min_samples_split': [2, 5, 10],

'n_estimators': [10, 100, 500, 1000, 1500]}

With hypertuning, we were able to obtain a cost savings of 86, 880*sinceourtotalcostwas*220,820 based on 754 False Positives and 1462 False Negatives.

The processing time of the full dataset using randomized grid search and these parameters was approximately three (3) hours.

Random Forest is a serious contender in any modeling scenario since it provides good model performance and is easy for business stakeholders to understand. However, it has the highest cost of processing time among all the three models.

6. Results

Model performance was surprisingly similar. The difference in cost due to prediction error between the highest and lowest performing models was $6540 (Table 3). There is a small gain of .027 from our Random Forest results to our XGBoost results.

Technique	False Positive	False Negative	Total Cost
XGBoost	794	1018	$214,280
Neural Network	671	1859	$216,040
Random Forest	754	1462	$220,820

Model Results (Table 3)

The confusion matrix for XGBoost (Figure 5) shows us that model accuracy is also high. Even though this is not our primary requirement, it is helpful to see that predictions may be trusted in the greater majority of cases.

XGBoost Model Confusion Matrix (Figure 5)

7. Next Steps

Our study confirmed that the Neural Network, XGBoost and Random Forest model results are very similar for this dataset. The choice of model depends on the business priorities. These models could contribute to an ensemble model, however we believe the small improvement in cost reduction may not justify the cost to develop and maintain the model.

Appendix B - Code

In [1]:
```
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
from tensorflow.keras.layers import Input, LSTM, Dense, Concatenate,GRU,Dropout,LeakyReLU
from tensorflow.keras.models import Model
from tensorflow import keras
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import StratifiedShuffleSplit
import pandas as pd
import numpy as np
import gzip
from tensorflow.keras import layers
from tensorflow.keras import initializers
import tensorflow as tf
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics.scorer import make_scorer
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.model_selection import RandomizedSearchCV
import locale
import pickle

print("Tensorflow version " + tf.__version__)
```

```
Tensorflow version 2.4.1

/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: The sklearn.metrics.sco
rer module is  deprecated in version 0.22 and will be removed in version 0.24. The corresponding classes / func
tions should instead be imported from sklearn.metrics. Anything that cannot be imported from sklearn.metrics is
now part of the private API.
  warnings.warn(message, FutureWarning)
```

In [ ]:
```
##########################################################
####         Data Import and Cleaning and Imputation  ####
##########################################################
```

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

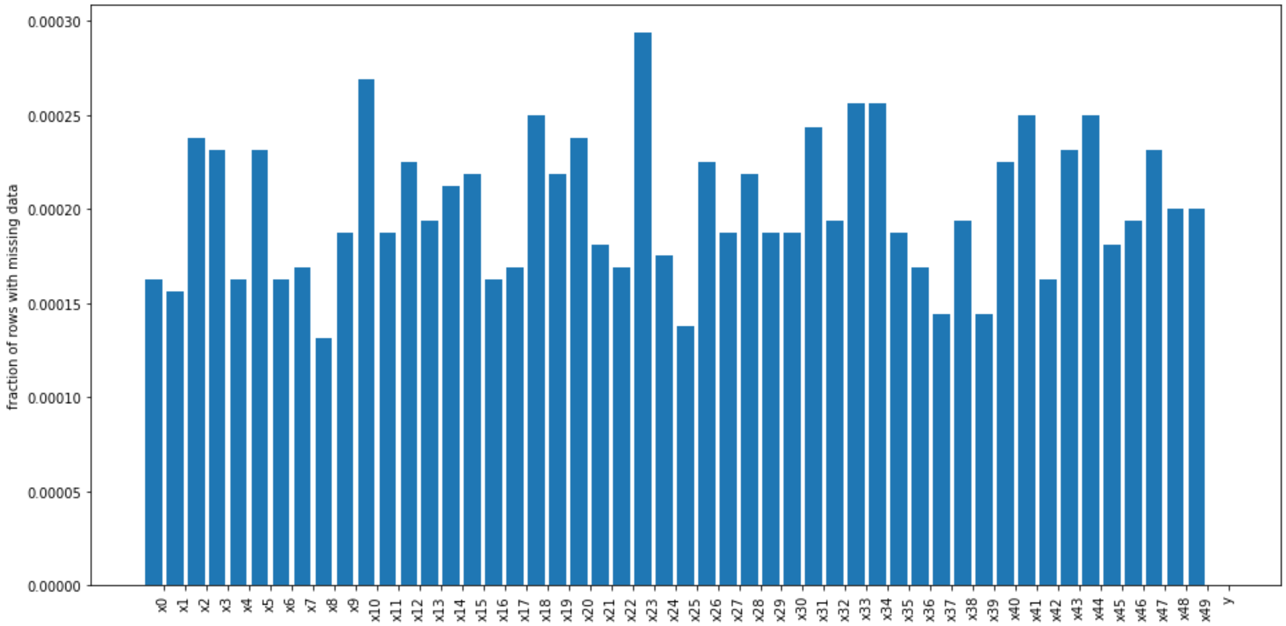
In [7]:
```
# Data Import and cleaning

data=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/final_project.csv")
#data = pd.read_csv("final_project.csv")
```

In [8]:

```python
# Graph the number of missing values per feature
null_counts = data.isnull().sum()/len(data)
plt.figure(figsize=(16,8))
plt.xticks(np.arange(len(null_counts))+0.5,null_counts.index,rotation='vertical')
plt.ylabel('fraction of rows with missing data')
plt.bar(np.arange(len(null_counts)),null_counts)
```

Out[8]:

```
<BarContainer object of 51 artists>
```



In [9]:
```
print("Top 5 lines\n\n", data.head())
# Count empty values
data.isnull().values.any().sum()
data.isna().values.any().sum()
data = data.dropna()
data.isna().values.any().sum()

print("\n\nShape \n\n", data.shape)

print("\n\nData information \n\n", data.info(verbose=True))

print("\n\nColumns \n\n",data.columns)

print("\n\n Description \n\n",data.describe())
# Feature Engineering
# Delete characters "," and "."  in column x37, x32 and change to numeric
data["x37"]=data["x37"].str.replace('[\$\,\.]', '')
data["x32"]=data["x32"].str.replace('[\%\,\.]', '')
(data["x37"],data["x32"])

data['x37']=pd.to_numeric(data['x37'])
data['x32']=pd.to_numeric(data['x32'])
print("\n\nx37 and x32 column \n\n", data["x37"], data["x37"])
print ("\n\n Data information after x37 and x32 column conversion to integer\n\n",data.info(verbose=True))


# Count null values
data.isnull().sum(axis = 0).sum()
# Discover duplicate values
data.columns.has_duplicates
# Count na if they still exist
data.isna().sum().sum()
```

```
Top 5 lines

         x0        x1         x2        x3  ...       x47       x48        x49  y
0 -0.166563 -3.961588   4.621113  2.481908  ... -7.689696  0.151589  -8.040166  0
1 -0.149894 -0.585676  27.839856  4.152333  ... -4.896678 -0.320283  16.719974  0
2 -0.321707 -1.429819  12.251561  6.586874  ... -7.428573 -2.090804  -7.869421  0
3 -0.245594  5.076677 -24.149632  3.637307  ...  5.361375  1.806070  -7.670847  0
4 -0.273366  0.306326 -11.352593  1.676758  ... -0.208351 -0.894942  15.724742  1

[5 rows x 51 columns]


Shape

 (158392, 51)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 158392 entries, 0 to 159999
Data columns (total 51 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   x0      158392 non-null  float64
 1   x1      158392 non-null  float64
 2   x2      158392 non-null  float64
 3   x3      158392 non-null  float64
 4   x4      158392 non-null  float64
 5   x5      158392 non-null  float64
 6   x6      158392 non-null  float64
 7   x7      158392 non-null  float64
 8   x8      158392 non-null  float64
 9   x9      158392 non-null  float64
 10  x10     158392 non-null  float64
 11  x11     158392 non-null  float64
 12  x12     158392 non-null  float64
 13  x13     158392 non-null  float64
 14  x14     158392 non-null  float64
 15  x15     158392 non-null  float64
 16  x16     158392 non-null  float64
 17  x17     158392 non-null  float64
 18  x18     158392 non-null  float64
 19  x19     158392 non-null  float64
 20  x20     158392 non-null  float64
 21  x21     158392 non-null  float64
 22  x22     158392 non-null  float64
 23  x23     158392 non-null  float64
 24  x24     158392 non-null  object 
 25  x25     158392 non-null  float64
 26  x26     158392 non-null  float64
 27  x27     158392 non-null  float64
 28  x28     158392 non-null  float64
 29  x29     158392 non-null  object 
 30  x30     158392 non-null  object 
 31  x31     158392 non-null  float64
 32  x32     158392 non-null  object 
 33  x33     158392 non-null  float64
 34  x34     158392 non-null  float64
 35  x35     158392 non-null  float64
 36  x36     158392 non-null  float64
 37  x37     158392 non-null  object 
 38  x38     158392 non-null  float64
 39  x39     158392 non-null  float64
 40  x40     158392 non-null  float64
 41  x41     158392 non-null  float64
 42  x42     158392 non-null  float64
 43  x43     158392 non-null  float64
 44  x44     158392 non-null  float64
 45  x45     158392 non-null  float64
 46  x46     158392 non-null  float64
 47  x47     158392 non-null  float64
 48  x48     158392 non-null  float64
 49  x49     158392 non-null  float64
 50  y       158392 non-null  int64  
dtypes: float64(45), int64(1), object(5)
memory usage: 62.8+ MB


Data information

 None


Columns

 Index(['x0', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8', 'x9', 'x10',
```

```
       'x11', 'x12', 'x13', 'x14', 'x15', 'x16', 'x17', 'x18', 'x19', 'x20',
       'x21', 'x22', 'x23', 'x24', 'x25', 'x26', 'x27', 'x28', 'x29', 'x30',
       'x31', 'x32', 'x33', 'x34', 'x35', 'x36', 'x37', 'x38', 'x39', 'x40',
       'x41', 'x42', 'x43', 'x44', 'x45', 'x46', 'x47', 'x48', 'x49', 'y'],
      dtype='object')


 Description

                  x0             x1  ...            x49              y
count  158392.000000  158392.000000  ...  158392.000000  158392.000000
mean       -0.000808       0.003705  ...      -0.672052       0.401195
std         0.371064       6.340297  ...      15.033134       0.490142
min        -1.592635     -26.278302  ...     -65.791191       0.000000
25%        -0.251246      -4.259016  ...     -10.929046       0.000000
50%        -0.001818       0.010023  ...      -0.569139       0.000000
75%         0.248622       4.286606  ...       9.649839       1.000000
max         1.600849      27.988178  ...      66.877604       1.000000

[8 rows x 46 columns]


x37 and x32 column

 0         131396
1         196278
2          43047
3        -236629
4         -62066
           ...
159995    -89196
159996    158865
159997     68746
159998     43921
159999   -122934
Name: x37, Length: 158392, dtype: int64 0         131396
1         196278
2          43047
3        -236629
4         -62066
           ...
159995    -89196
159996    158865
159997     68746
159998     43921
159999   -122934
Name: x37, Length: 158392, dtype: int64
<class 'pandas.core.frame.DataFrame'>
Int64Index: 158392 entries, 0 to 159999
Data columns (total 51 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   x0      158392 non-null  float64
 1   x1      158392 non-null  float64
 2   x2      158392 non-null  float64
 3   x3      158392 non-null  float64
 4   x4      158392 non-null  float64
 5   x5      158392 non-null  float64
 6   x6      158392 non-null  float64
 7   x7      158392 non-null  float64
 8   x8      158392 non-null  float64
 9   x9      158392 non-null  float64
 10  x10     158392 non-null  float64
 11  x11     158392 non-null  float64
 12  x12     158392 non-null  float64
 13  x13     158392 non-null  float64
 14  x14     158392 non-null  float64
 15  x15     158392 non-null  float64
 16  x16     158392 non-null  float64
 17  x17     158392 non-null  float64
 18  x18     158392 non-null  float64
 19  x19     158392 non-null  float64
 20  x20     158392 non-null  float64
 21  x21     158392 non-null  float64
 22  x22     158392 non-null  float64
 23  x23     158392 non-null  float64
 24  x24     158392 non-null  object
 25  x25     158392 non-null  float64
 26  x26     158392 non-null  float64
 27  x27     158392 non-null  float64
 28  x28     158392 non-null  float64
 29  x29     158392 non-null  object
 30  x30     158392 non-null  object
```

```
 31  x31     158392 non-null  float64
 32  x32     158392 non-null  int64  
 33  x33     158392 non-null  float64
 34  x34     158392 non-null  float64
 35  x35     158392 non-null  float64
 36  x36     158392 non-null  float64
 37  x37     158392 non-null  int64  
 38  x38     158392 non-null  float64
 39  x39     158392 non-null  float64
 40  x40     158392 non-null  float64
 41  x41     158392 non-null  float64
 42  x42     158392 non-null  float64
 43  x43     158392 non-null  float64
 44  x44     158392 non-null  float64
 45  x45     158392 non-null  float64
 46  x46     158392 non-null  float64
 47  x47     158392 non-null  float64
 48  x48     158392 non-null  float64
 49  x49     158392 non-null  float64
 50  y       158392 non-null  int64  
dtypes: float64(45), int64(3), object(3)
memory usage: 62.8+ MB


 Data information after x37 and x32 column conversion to integer

 None
```

Out[9]: 0

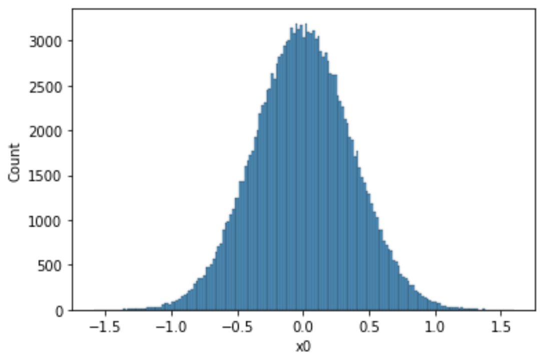
In [10]:
```
# Separate the categoriacla columns from the data and process them later but here just trying to plotting numerical columns
visdf=data[data.columns.difference(['x24','x29','x30'])]
```

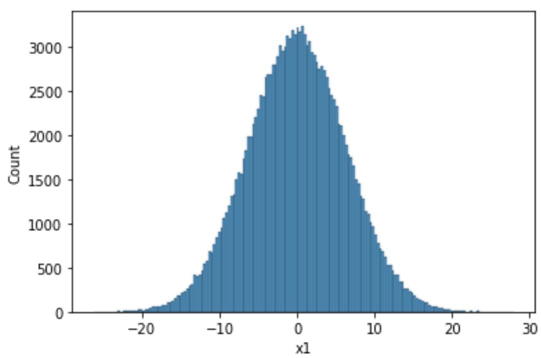
In [11]:

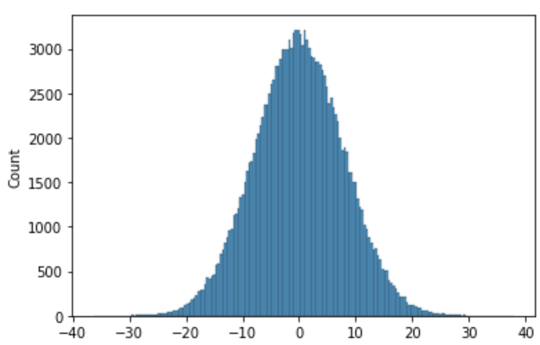
```
# Visualizations
# Finding out which columns are notmal data and which are not
import seaborn as sns


for i, col in enumerate(visdf.columns):
    plt.figure(i)
    sns.histplot(visdf[col])
```

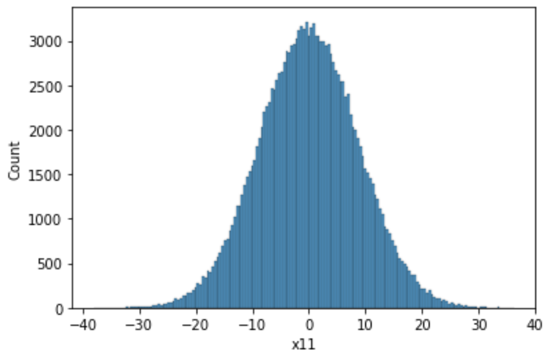
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
```

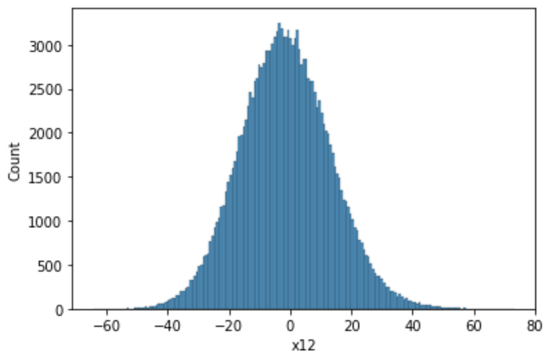
```
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  import sys
```

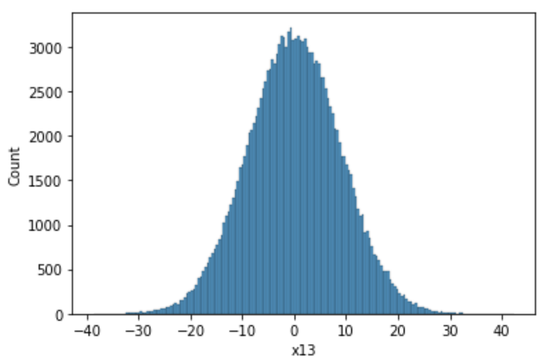


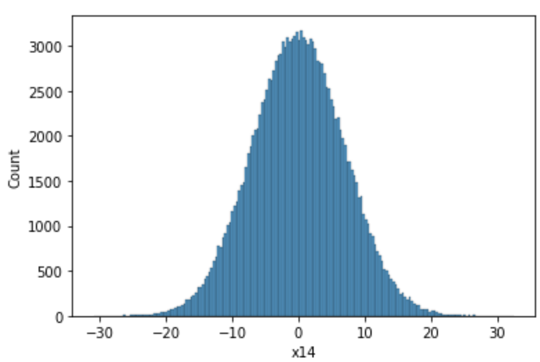


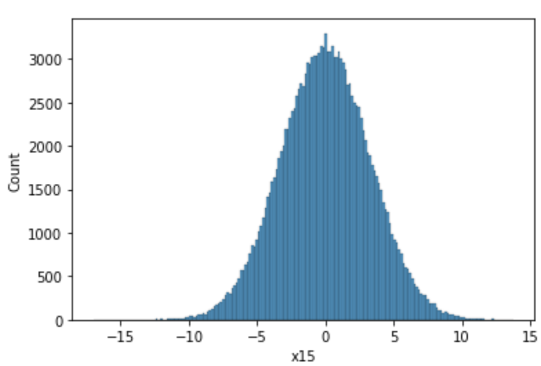


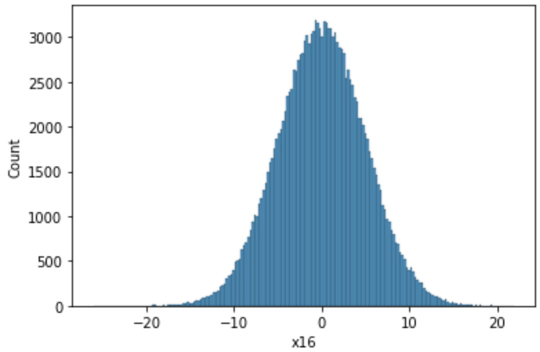

Count
0
500
1000
1500
2000
2500
3000
−40 −30 −20 −10 0 10 20 30 40
x11

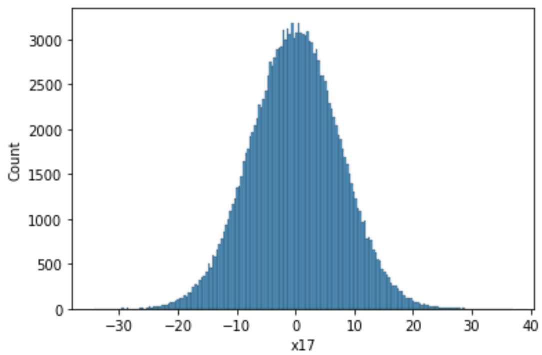

Count
0
500
1000
1500
2000
2500
3000
−60 −40 −20 0 20 40 60 80
x12

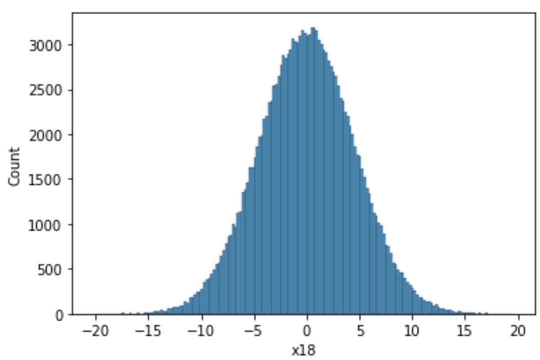

Count
0
500
1000
1500
2000
2500
3000
−40 −30 −20 −10 0 10 20 30 40
x13

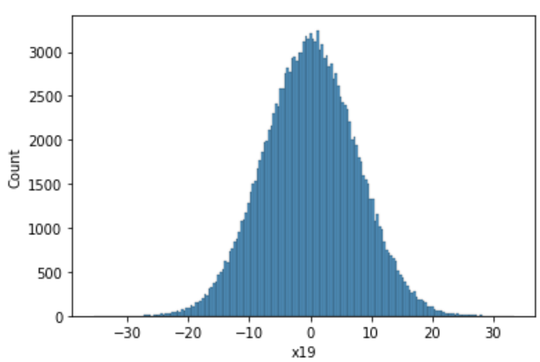

Count
0
500
1000
1500
2000
2500
3000
−30 −20 −10 0 10 20 30
x14

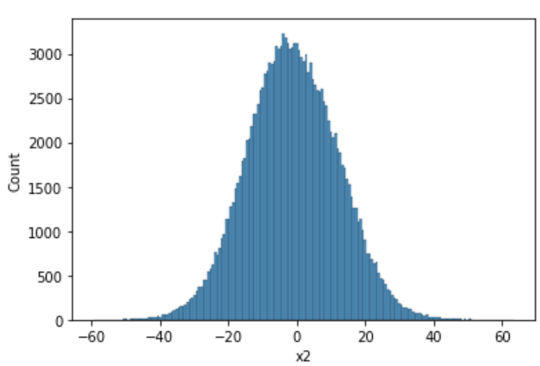

Count
0
500
1000
1500
2000
2500
3000
−15 −10 −5 0 5 10 15
x15

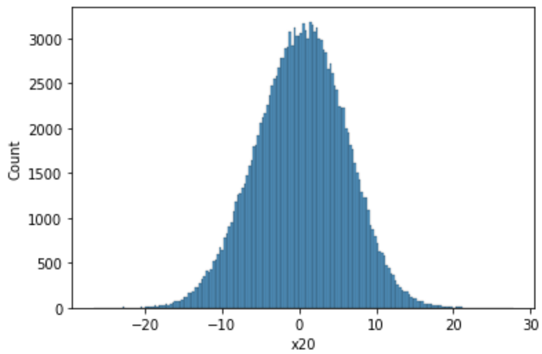

Count
x16

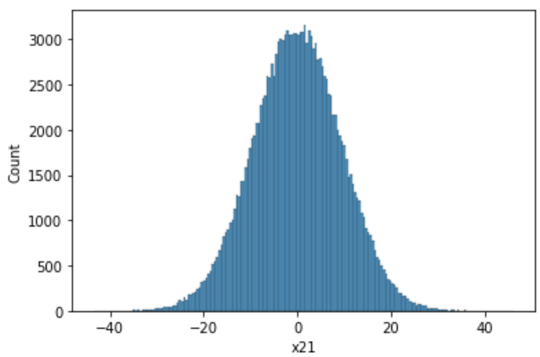

Count
x17

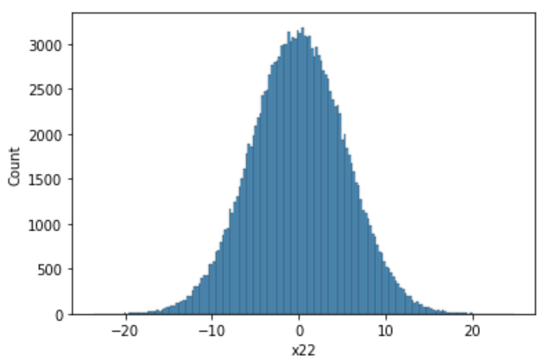

Count
x18

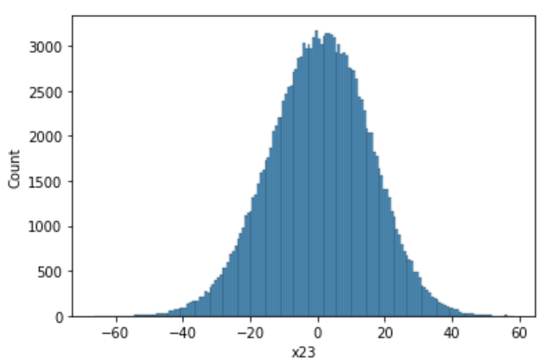

Count
x19

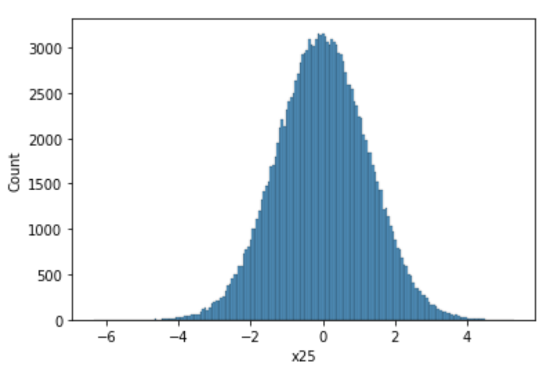

Count
x2

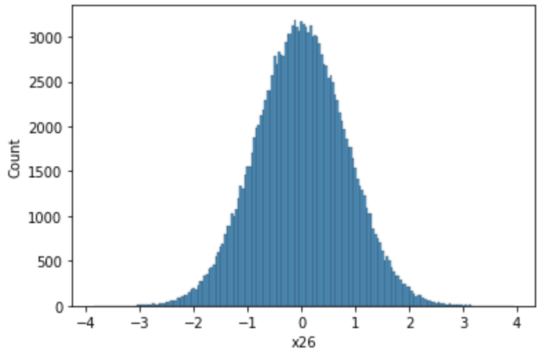

Count
0
500
1000
1500
2000
2500
3000
−20
−10
0
10
20
30
x20

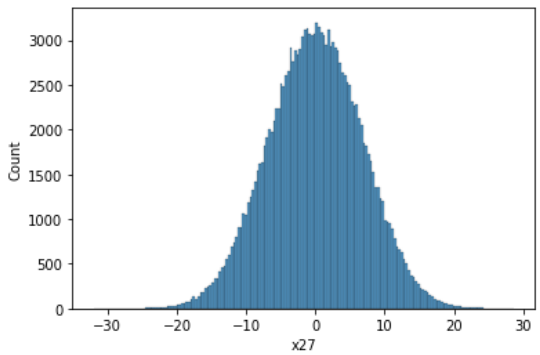

Count
0
500
1000
1500
2000
2500
3000
−40
−20
0
20
40
x21

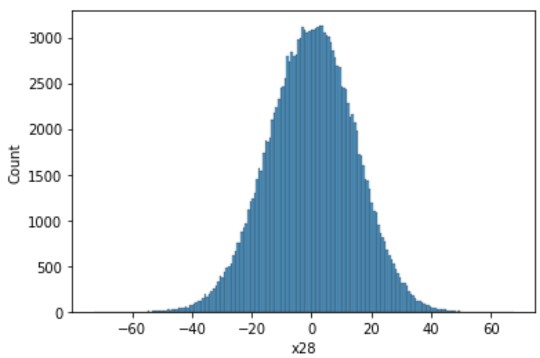

Count
0
500
1000
1500
2000
2500
3000
−20
−10
0
10
20
x22

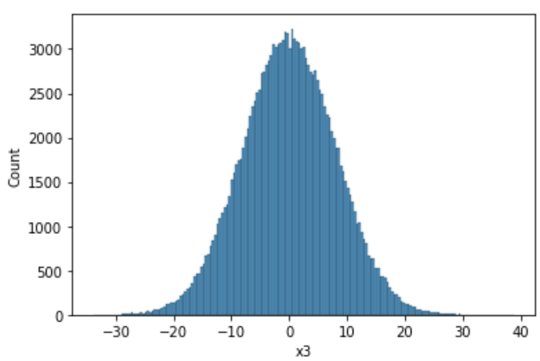

Count
0
500
1000
1500
2000
2500
3000
−60
−40
−20
0
20
40
60
x23

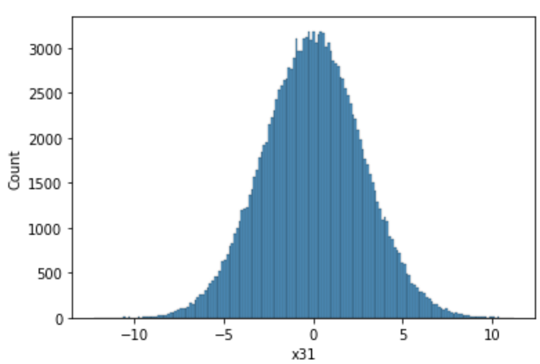

Count
0
500
1000
1500
2000
2500
3000
−6
−4
−2
0
2
4
x25

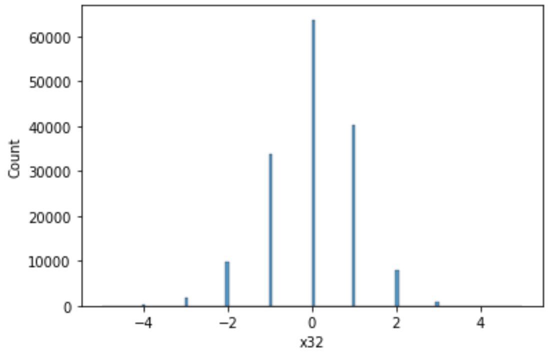

Count
x26

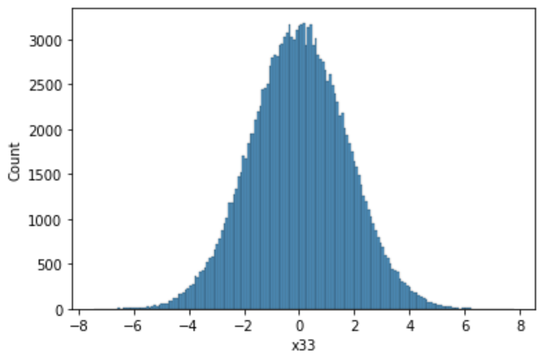

Count
x27

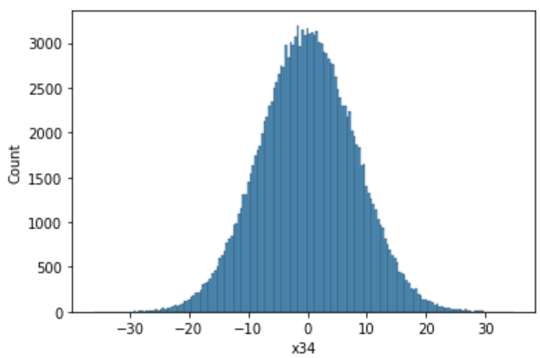

Count
x28

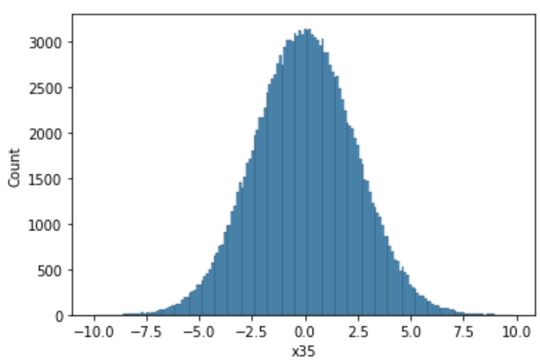

Count
x3

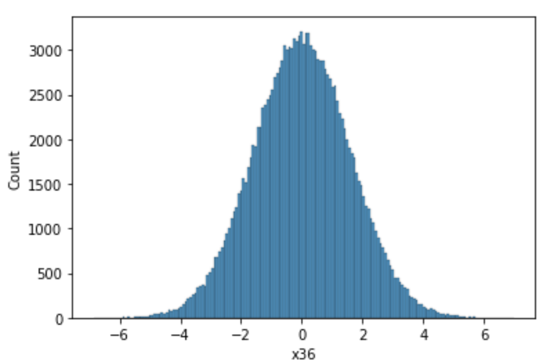

Count
x31

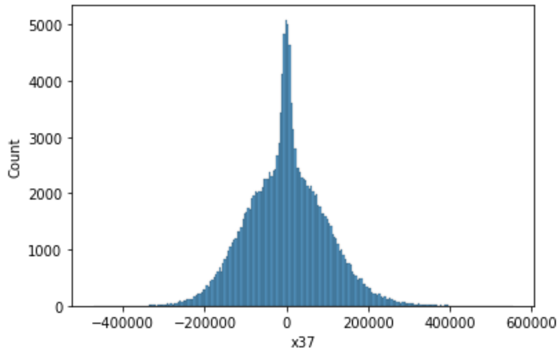

Count
0
10000
20000
30000
40000
50000
60000
−4
−2
0
2
4
x32

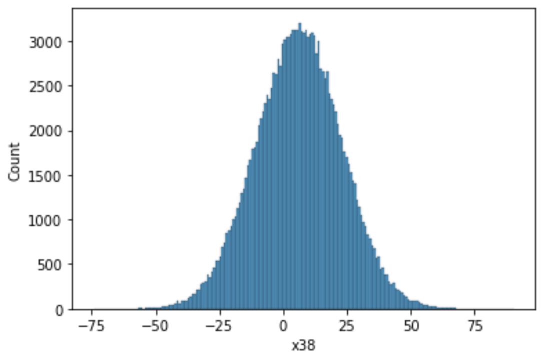

Count
0
500
1000
1500
2000
2500
3000
−8
−6
−4
−2
0
2
4
6
8
x33

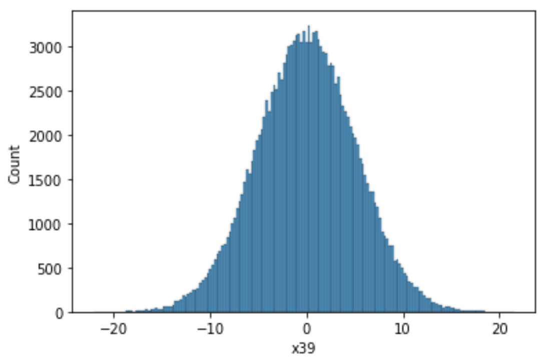

Count
0
500
1000
1500
2000
2500
3000
−30
−20
−10
0
10
20
30
x34

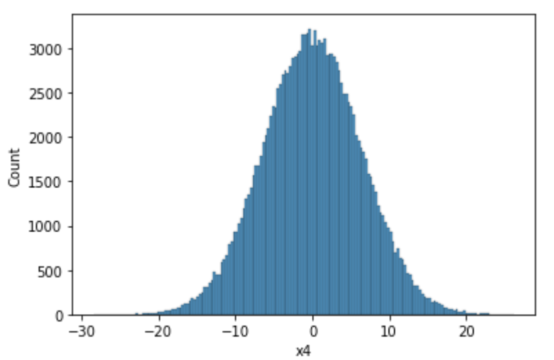

Count
0
500
1000
1500
2000
2500
3000
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
x35

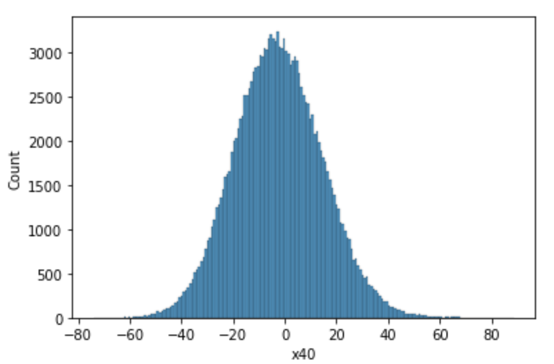

Count
0
500
1000
1500
2000
2500
3000
−6
−4
−2
0
2
4
6
x36

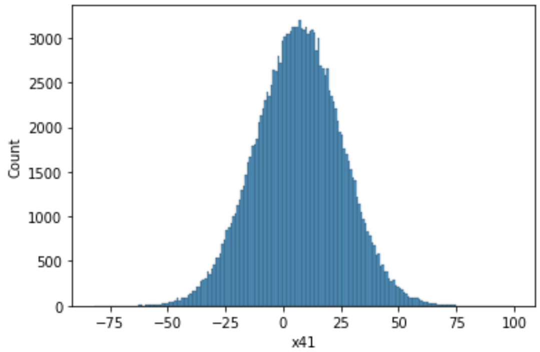

Count
5000
4000
3000
2000
1000
0
−400000
−200000
0
200000
400000
600000
x37

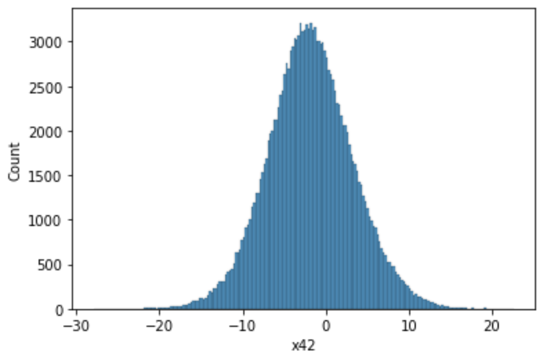

Count
3000
2500
2000
1500
1000
500
0
−75
−50
−25
0
25
50
75
x38

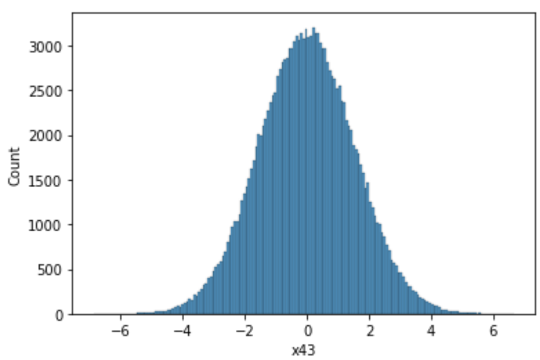

Count
3000
2500
2000
1500
1000
500
0
−20
−10
0
10
20
x39

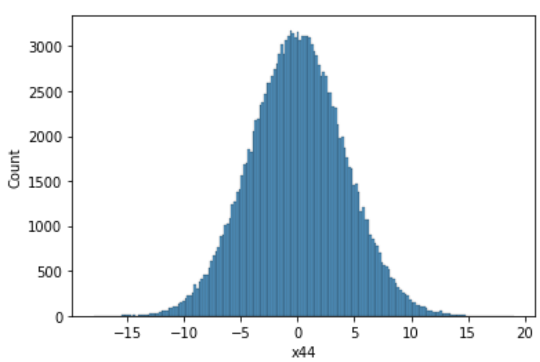

Count
3000
2500
2000
1500
1000
500
0
−30
−20
−10
0
10
20
x4

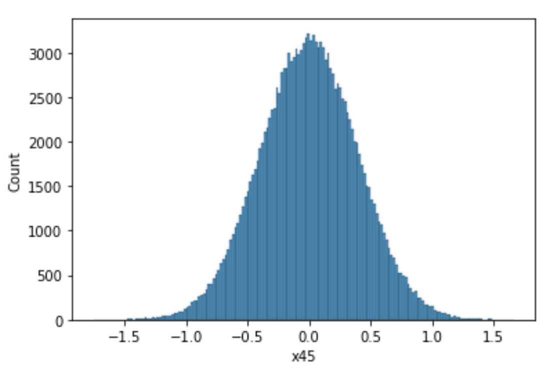

Count
3000
2500
2000
1500
1000
500
0
−80
−60
−40
−20
0
20
40
60
80
x40

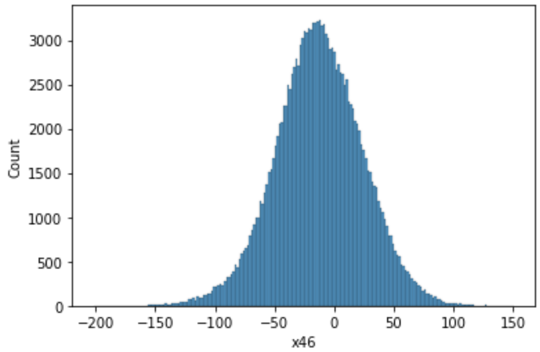

Count
3000
2500
2000
1500
1000
500
0
−75 −50 −25 0 25 50 75 100
x41


Count
3000
2500
2000
1500
1000
500
0
−30 −20 −10 0 10 20
x42

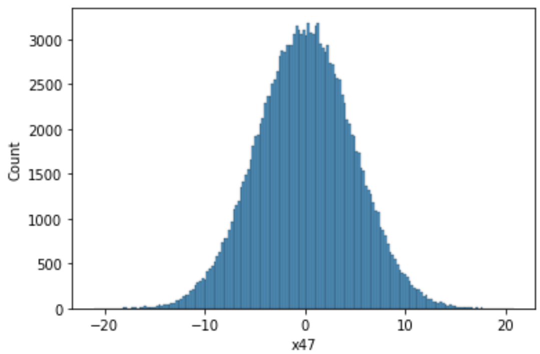

Count
3000
2500
2000
1500
1000
500
0
−6 −4 −2 0 2 4 6
x43


Count
3000
2500
2000
1500
1000
500
0
−15 −10 −5 0 5 10 15 20
x44

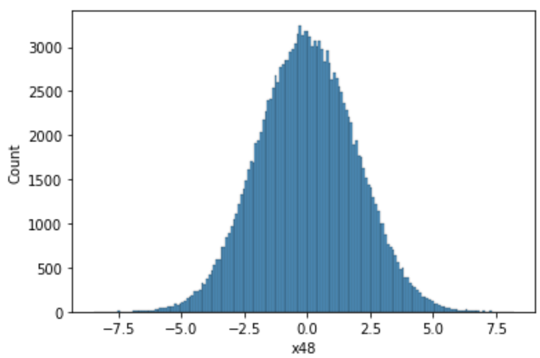

Count
3000
2500
2000
1500
1000
500
0
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
x45



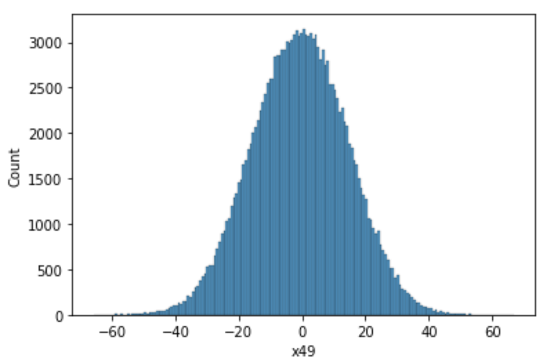
Count
0
500
1000
1500
2000
2500
3000
−200
−150
−100
−50
0
50
100
150
x46



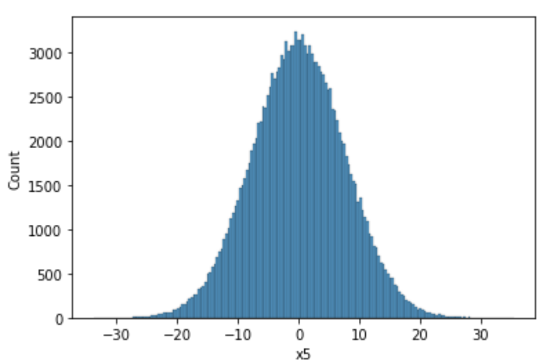
Count
0
500
1000
1500
2000
2500
3000
−20
−10
0
10
20
x47



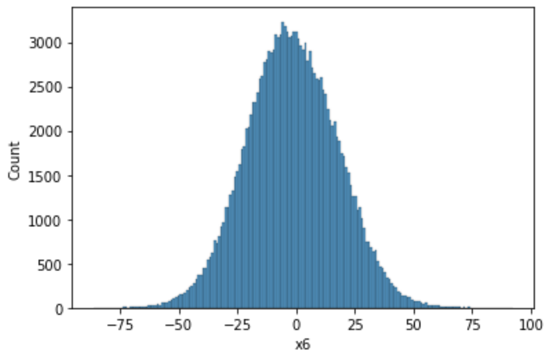
Count
0
500
1000
1500
2000
2500
3000
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
x48

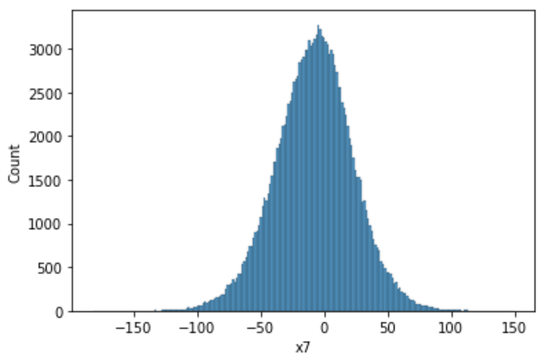


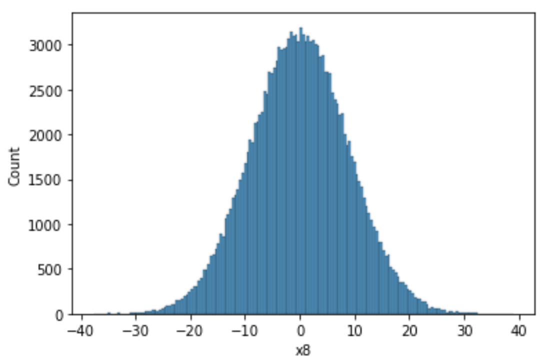
Count
0
500
1000
1500
2000
2500
3000
−60
−40
−20
0
20
40
60
x49

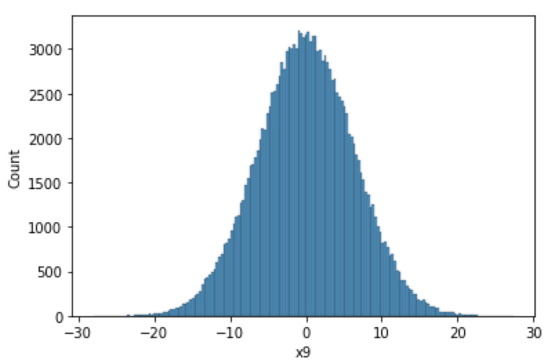


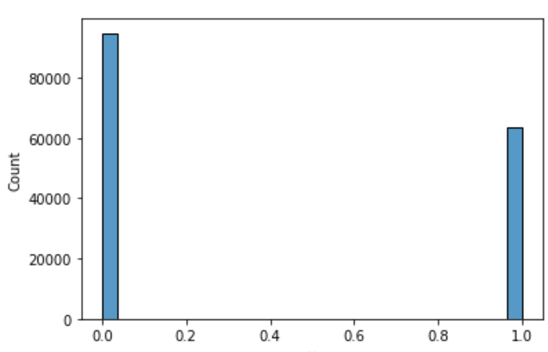
Count
0
500
1000
1500
2000
2500
3000
−30
−20
−10
0
10
20
30
x5


Count
3000
2500
2000
1500
1000
500
0
−75 −50 −25 0 25 50 75 100
x6


Count
3000
2500
2000
1500
1000
500
0
−150 −100 −50 0 50 100 150
x7


Count
3000
2500
2000
1500
1000
500
0
−40 −30 −20 −10 0 10 20 30 40
x8


Count
3000
2500
2000
1500
1000
500
0
−30 −20 −10 0 10 20 30
x9


Count
80000
60000
40000
20000
0
0.0 0.2 0.4 0.6 0.8 1.0
y

In [12]:
```
# ** Imputations of Numerical Columns **
# *****. From the graphs it seems we can use **mean** to impute the missing data from the numerical columns ****
*
df=visdf.fillna(visdf.mean())

#Testing if there are anymore missing data left behind
df.isna().sum().sum()

#how data looks lik after imputations
df.info(verbose=True)
data.columns
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 158392 entries, 0 to 159999
Data columns (total 48 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   x0      158392 non-null  float64
 1   x1      158392 non-null  float64
 2   x10     158392 non-null  float64
 3   x11     158392 non-null  float64
 4   x12     158392 non-null  float64
 5   x13     158392 non-null  float64
 6   x14     158392 non-null  float64
 7   x15     158392 non-null  float64
 8   x16     158392 non-null  float64
 9   x17     158392 non-null  float64
 10  x18     158392 non-null  float64
 11  x19     158392 non-null  float64
 12  x2      158392 non-null  float64
 13  x20     158392 non-null  float64
 14  x21     158392 non-null  float64
 15  x22     158392 non-null  float64
 16  x23     158392 non-null  float64
 17  x25     158392 non-null  float64
 18  x26     158392 non-null  float64
 19  x27     158392 non-null  float64
 20  x28     158392 non-null  float64
 21  x3      158392 non-null  float64
 22  x31     158392 non-null  float64
 23  x32     158392 non-null  int64  
 24  x33     158392 non-null  float64
 25  x34     158392 non-null  float64
 26  x35     158392 non-null  float64
 27  x36     158392 non-null  float64
 28  x37     158392 non-null  int64  
 29  x38     158392 non-null  float64
 30  x39     158392 non-null  float64
 31  x4      158392 non-null  float64
 32  x40     158392 non-null  float64
 33  x41     158392 non-null  float64
 34  x42     158392 non-null  float64
 35  x43     158392 non-null  float64
 36  x44     158392 non-null  float64
 37  x45     158392 non-null  float64
 38  x46     158392 non-null  float64
 39  x47     158392 non-null  float64
 40  x48     158392 non-null  float64
 41  x49     158392 non-null  float64
 42  x5      158392 non-null  float64
 43  x6      158392 non-null  float64
 44  x7      158392 non-null  float64
 45  x8      158392 non-null  float64
 46  x9      158392 non-null  float64
 47  y       158392 non-null  int64  
dtypes: float64(45), int64(3)
memory usage: 59.2 MB
```

Out[12]:

	x0	x1	x10	x11	x12	x13	x14	x15	x16	
count	158392.000000	158392.000000	158392.000000	158392.000000	158392.000000	158392.000000	158392.000000	158392.000000	158392.000000	1583
mean	-0.000808	0.003705	0.000816	0.030692	-1.337022	0.005699	0.008887	0.002436	0.006746	
std	0.371064	6.340297	7.870963	8.767797	14.752763	8.952626	6.964429	3.271402	4.982869	
min	-1.592635	-26.278302	-36.306571	-38.092869	-64.197967	-38.723514	-30.905214	-17.002359	-26.042983	-
25%	-0.251246	-4.259016	-5.286455	-5.902750	-11.383333	-6.030792	-4.695374	-2.207028	-3.343254	
50%	-0.001818	0.010023	-0.019074	0.013579	-1.627464	-0.004343	0.003644	0.005473	0.012754	
75%	0.248622	4.286606	5.327598	5.933786	8.375380	6.039018	4.702776	2.212473	3.366107	
max	1.600849	27.988178	37.945583	36.360443	73.279354	42.392177	32.546340	13.782559	21.961123	

In [13]:
```
# *******.  Imputations for Categorical columns. *******
# *** Find categorical columns ***

catcolumns=data.select_dtypes(include='object')
catcolumns.head(10)

#put the cat_columns into another data frame to process further
data2=data[['x24','x29','x30']]

data2.info(verbose=True)

#Test it whether it has any missing values
data2.isna().sum()

# Here we put "Unknown" class for imputation of missing categorical values

# Reference:https://jamesrledoux.com/code/imputation


# Validate data one more time before seperating target feature
data2.fillna("Unknown",inplace=True)
data2.isna().sum()
data2.info(verbose=True)
data2.columns.has_duplicates
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 158392 entries, 0 to 159999
Data columns (total 3 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   x24     158392 non-null  object
 1   x29     158392 non-null  object
 2   x30     158392 non-null  object
dtypes: object(3)
memory usage: 4.8+ MB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 158392 entries, 0 to 159999
Data columns (total 3 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   x24     158392 non-null  object
 1   x29     158392 non-null  object
 2   x30     158392 non-null  object
dtypes: object(3)
memory usage: 4.8+ MB

/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4327: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  downcast=downcast,
```

Out[13]: False

In [14]:
```python
# Seperate out your Y value
y=data['y']
print ("\n\n Target value \n\n",y)

df.head()
df.drop(['y'],axis=1,inplace=True)
print ("\n\n Data with target column removed \n\n",data.head())

data_object=data2.select_dtypes(include='object')
data_object.head(5)
print (data_object.head(10))
```

```
 Target value

 0         0
1         0
2         0
3         0
4         1
         ..
159995    1
159996    0
159997    1
159998    0
159999    1
Name: y, Length: 158392, dtype: int64


 Data with target column removed

         x0        x1         x2        x3  ...        x47       x48        x49  y
0 -0.166563 -3.961588   4.621113  2.481908  ... -7.689696  0.151589  -8.040166  0
1 -0.149894 -0.585676  27.839856  4.152333  ... -4.896678 -0.320283  16.719974  0
2 -0.321707 -1.429819  12.251561  6.586874  ... -7.428573 -2.090804  -7.869421  0
3 -0.245594  5.076677 -24.149632  3.637307  ...  5.361375  1.806070  -7.670847  0
4 -0.273366  0.306326 -11.352593  1.676758  ... -0.208351 -0.894942  15.724742  1

[5 rows x 51 columns]
      x24   x29        x30
0  euorpe  July    tuesday
1    asia   Aug  wednesday
2    asia  July  wednesday
3    asia  July  wednesday
4    asia  July    tuesday
5    asia   Aug  wednesday
6    asia   Jun  wednesday
7    asia   Aug  wednesday
8    asia   May  wednesday
9    asia   Jun  wednesday
```

In [15]:
```python
# One hot encode categorical values
data_object.describe(include='all').loc['unique', :]

hot_encoed_df=pd.get_dummies(data_object)
hot_encoed_df.head()

frames = [df,hot_encoed_df]
df2=pd.concat(frames,axis=1)
df2.shape
df2.head()
df2.isna().sum().sum()
df2.columns.has_duplicates
```

Out[15]: False

In [ ]:
```python
##########################################################
####         Model 1 Neural Network.                  ####
##########################################################
```

In [ ]:
```
 # Model

from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.regularizers import l2
from tensorflow.keras import optimizers

scaler = MinMaxScaler(feature_range=(-1, 1))
scaled_train = scaler.fit_transform(df2)
scaled_train
try:
     tpu = tf.distribute.cluster_resolver.TPUClusterResolver.connect() # TPU detection
     strategy = tf.distribute.TPUStrategy(tpu)
except ValueError: # detect GPUs
     strategy = tf.distribute.get_strategy() # default strategy that works on CPU and single GPU
# strategy = tf.distribute.get_strategy() # default strategy that works on CPU and single GPU
print("Number of accelerators: ", strategy.num_replicas_in_sync)

AUTO = tf.data.experimental.AUTOTUNE

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(scaled_train,y, test_size=0.20, random_state=123)

seed = 42
first_layer_init = initializers.RandomNormal(
    mean=0.0, stddev=0.1, seed=seed)
hidden_layer_init = initializers.RandomNormal(
    mean=0.0, stddev=0.05, seed=seed)
output_layer_init = initializers.RandomNormal(
    mean=0.0, stddev=0.001, seed=seed)
weight_decay = 1 * 10**-5
model = tf.keras.Sequential()
model.add(layers.Dense(32,activation='tanh',kernel_initializer=hidden_layer_init,kernel_regularizer=l2(weight_de
cay)))  # adds a layer with 32 neurons, tanh activation
model.add(layers.Dense(12,activation='tanh',kernel_initializer=hidden_layer_init,kernel_regularizer=l2(weight_de
cay)))  # adds a layer with 12 neurons, tanh activation
model.add(layers.Dense(1, activation='sigmoid',kernel_initializer=output_layer_init,kernel_regularizer=l2(weight
_decay)))  # adds a layer with 1 neurons, sigmoid activation

auc_score = tf.keras.metrics.AUC() # define AUC score for model output

model.compile(optimizer=optimizers.SGD(lr=.05), loss='binary_crossentropy', metrics=['accuracy',auc_score,tf.ker
as.metrics.FalsePositives(),tf.keras.metrics.FalseNegatives()]) # optimized with SGD with a learning rate of .05
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping

callbacks = [EarlyStopping(patience=5, monitor='val_false_positives', min_delta=0.00001)]  # stop model after 2
 epochs with no improvement based on epoch accuracy

model.fit(x_train, y_train, epochs=40, validation_data=(x_test,y_test), batch_size=30,callbacks=callbacks)
```

```
WARNING:tensorflow:TPU system grpc://10.51.243.98:8470 has already been initialized. Reinitializing the TPU can cause previously created variables on TPU to be lost.

WARNING:tensorflow:TPU system grpc://10.51.243.98:8470 has already been initialized. Reinitializing the TPU can cause previously created variables on TPU to be lost.

INFO:tensorflow:Initializing the TPU system: grpc://10.51.243.98:8470

INFO:tensorflow:Initializing the TPU system: grpc://10.51.243.98:8470

INFO:tensorflow:Clearing out eager caches

INFO:tensorflow:Clearing out eager caches

INFO:tensorflow:Finished initializing TPU system.

INFO:tensorflow:Finished initializing TPU system.

INFO:tensorflow:Found TPU system:

INFO:tensorflow:Found TPU system:

INFO:tensorflow:*** Num TPU Cores: 8

INFO:tensorflow:*** Num TPU Cores: 8

INFO:tensorflow:*** Num TPU Workers: 1

INFO:tensorflow:*** Num TPU Workers: 1

INFO:tensorflow:*** Num TPU Cores Per Worker: 8

INFO:tensorflow:*** Num TPU Cores Per Worker: 8

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:localhost/replica:0/task:0/device:CPU:0, CPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:localhost/replica:0/task:0/device:CPU:0, CPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:CPU:0, CPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:CPU:0, CPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:0, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:0, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:1, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:1, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:2, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:2, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:3, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:3, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:4, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:4, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:5, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:5, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:6, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:6, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:7, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:7, TPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU_SYSTEM:0, TPU_SYSTEM, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU_SYSTEM:0, TPU_SYSTEM, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:XLA_CPU:0, XLA_CPU, 0, 0)

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:XLA_CPU:0, XLA_CPU, 0, 0)
```

```
Number of accelerators:  8
Epoch 1/40
4224/4224 [==============================] - 109s 26ms/step - loss: 0.6597 - accuracy: 0.6125 - auc_1: 0.5760 -
false_positives_1: 1672.0821 - false_negatives_1: 22324.3297 - val_loss: 0.5739 - val_accuracy: 0.7049 - val_au
c_1: 0.7570 - val_false_positives_1: 3523.0000 - val_false_negatives_1: 5827.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 2/40
4224/4224 [==============================] - 107s 25ms/step - loss: 0.5719 - accuracy: 0.7039 - auc_1: 0.7567 -
false_positives_1: 7413.5344 - false_negatives_1: 11307.5950 - val_loss: 0.5455 - val_accuracy: 0.7126 - val_au
c_1: 0.7845 - val_false_positives_1: 3170.0000 - val_false_negatives_1: 5935.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 3/40
4224/4224 [==============================] - 107s 25ms/step - loss: 0.5227 - accuracy: 0.7365 - auc_1: 0.8039 -
false_positives_1: 6333.9877 - false_negatives_1: 9849.9347 - val_loss: 0.4123 - val_accuracy: 0.8185 - val_auc
_1: 0.8918 - val_false_positives_1: 2415.0000 - val_false_negatives_1: 3334.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 4/40
4224/4224 [==============================] - 107s 25ms/step - loss: 0.4131 - accuracy: 0.8166 - auc_1: 0.8878 -
false_positives_1: 4951.9049 - false_negatives_1: 6479.4298 - val_loss: 0.4237 - val_accuracy: 0.7987 - val_auc
_1: 0.9179 - val_false_positives_1: 780.0000 - val_false_negatives_1: 5597.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 5/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.3644 - accuracy: 0.8422 - auc_1: 0.9148 -
false_positives_1: 4530.0750 - false_negatives_1: 5359.3425 - val_loss: 0.3318 - val_accuracy: 0.8587 - val_auc
_1: 0.9304 - val_false_positives_1: 2353.0000 - val_false_negatives_1: 2122.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 6/40
4224/4224 [==============================] - 107s 25ms/step - loss: 0.3401 - accuracy: 0.8528 - auc_1: 0.9261 -
false_positives_1: 4323.5328 - false_negatives_1: 4907.0902 - val_loss: 0.3669 - val_accuracy: 0.8378 - val_auc
_1: 0.9355 - val_false_positives_1: 4217.0000 - val_false_negatives_1: 921.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 7/40
4224/4224 [==============================] - 107s 25ms/step - loss: 0.3223 - accuracy: 0.8628 - auc_1: 0.9340 -
false_positives_1: 4127.5879 - false_negatives_1: 4557.5548 - val_loss: 0.3048 - val_accuracy: 0.8699 - val_auc
_1: 0.9413 - val_false_positives_1: 2078.0000 - val_false_negatives_1: 2044.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1
```

```
Epoch 8/40
4224/4224 [==============================] - 107s 25ms/step - loss: 0.3126 - accuracy: 0.8667 - auc_1: 0.9378 -
false_positives_1: 3968.6315 - false_negatives_1: 4403.2949 - val_loss: 0.3056 - val_accuracy: 0.8688 - val_auc
_1: 0.9448 - val_false_positives_1: 1320.0000 - val_false_negatives_1: 2835.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 9/40
4224/4224 [==============================] - 108s 25ms/step - loss: 0.3046 - accuracy: 0.8699 - auc_1: 0.9412 -
false_positives_1: 3922.6618 - false_negatives_1: 4241.1877 - val_loss: 0.2953 - val_accuracy: 0.8751 - val_auc
_1: 0.9470 - val_false_positives_1: 1388.0000 - val_false_negatives_1: 2568.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 10/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.3003 - accuracy: 0.8739 - auc_1: 0.9429 -
false_positives_1: 3818.5967 - false_negatives_1: 4147.4888 - val_loss: 0.2892 - val_accuracy: 0.8787 - val_auc
_1: 0.9478 - val_false_positives_1: 2052.0000 - val_false_negatives_1: 1791.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 11/40
4224/4224 [==============================] - 102s 24ms/step - loss: 0.2909 - accuracy: 0.8783 - auc_1: 0.9466 -
false_positives_1: 3687.1337 - false_negatives_1: 4026.9737 - val_loss: 0.2862 - val_accuracy: 0.8813 - val_auc
_1: 0.9488 - val_false_positives_1: 1911.0000 - val_false_negatives_1: 1849.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 12/40
4224/4224 [==============================] - 103s 24ms/step - loss: 0.2840 - accuracy: 0.8821 - auc_1: 0.9491 -
false_positives_1: 3585.1702 - false_negatives_1: 3908.0845 - val_loss: 0.2932 - val_accuracy: 0.8763 - val_auc
_1: 0.9514 - val_false_positives_1: 2660.0000 - val_false_negatives_1: 1260.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 13/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.2787 - accuracy: 0.8840 - auc_1: 0.9511 -
false_positives_1: 3470.4639 - false_negatives_1: 3872.3200 - val_loss: 0.2684 - val_accuracy: 0.8889 - val_auc
_1: 0.9551 - val_false_positives_1: 1545.0000 - val_false_negatives_1: 1974.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 14/40
4224/4224 [==============================] - 105s 25ms/step - loss: 0.2694 - accuracy: 0.8899 - auc_1: 0.9543 -
false_positives_1: 3305.0824 - false_negatives_1: 3674.5517 - val_loss: 0.2614 - val_accuracy: 0.8927 - val_auc
_1: 0.9580 - val_false_positives_1: 1825.0000 - val_false_negatives_1: 1574.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1
```

```
Epoch 15/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.2646 - accuracy: 0.8932 - auc_1: 0.9560 -
false_positives_1: 3143.5851 - false_negatives_1: 3642.3366 - val_loss: 0.2630 - val_accuracy: 0.8911 - val_auc
_1: 0.9579 - val_false_positives_1: 1286.0000 - val_false_negatives_1: 2164.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 16/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.2615 - accuracy: 0.8955 - auc_1: 0.9571 -
false_positives_1: 3083.6542 - false_negatives_1: 3534.2462 - val_loss: 0.2654 - val_accuracy: 0.8912 - val_auc
_1: 0.9570 - val_false_positives_1: 1206.0000 - val_false_negatives_1: 2240.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 17/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.2568 - accuracy: 0.8970 - auc_1: 0.9588 -
false_positives_1: 3047.4480 - false_negatives_1: 3463.6092 - val_loss: 0.2487 - val_accuracy: 0.9002 - val_auc
_1: 0.9627 - val_false_positives_1: 1080.0000 - val_false_negatives_1: 2080.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 18/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.2520 - accuracy: 0.8990 - auc_1: 0.9602 -
false_positives_1: 2968.6400 - false_negatives_1: 3430.7903 - val_loss: 0.2535 - val_accuracy: 0.8993 - val_auc
_1: 0.9617 - val_false_positives_1: 1888.0000 - val_false_negatives_1: 1303.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 19/40
4224/4224 [==============================] - 105s 25ms/step - loss: 0.2482 - accuracy: 0.9011 - auc_1: 0.9615 -
false_positives_1: 2908.0244 - false_negatives_1: 3381.8821 - val_loss: 0.2461 - val_accuracy: 0.9013 - val_auc
_1: 0.9652 - val_false_positives_1: 2015.0000 - val_false_negatives_1: 1111.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 20/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.2451 - accuracy: 0.9029 - auc_1: 0.9623 -
false_positives_1: 2857.3733 - false_negatives_1: 3256.0587 - val_loss: 0.2464 - val_accuracy: 0.9006 - val_auc
_1: 0.9633 - val_false_positives_1: 1047.0000 - val_false_negatives_1: 2103.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 21/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.2409 - accuracy: 0.9046 - auc_1: 0.9638 -
false_positives_1: 2817.1489 - false_negatives_1: 3193.8587 - val_loss: 0.2360 - val_accuracy: 0.9070 - val_auc
_1: 0.9656 - val_false_positives_1: 1219.0000 - val_false_negatives_1: 1727.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1
```

```
Epoch 22/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.2350 - accuracy: 0.9073 - auc_1: 0.9656 - false_positives_1: 2719.7006 - false_negatives_1: 3144.4909 - val_loss: 0.2272 - val_accuracy: 0.9109 - val_auc_1: 0.9682 - val_false_positives_1: 1333.0000 - val_false_negatives_1: 1489.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

Epoch 23/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.2331 - accuracy: 0.9094 - auc_1: 0.9662 - false_positives_1: 2654.9060 - false_negatives_1: 3106.3808 - val_loss: 0.2389 - val_accuracy: 0.9071 - val_auc_1: 0.9681 - val_false_positives_1: 1974.0000 - val_false_negatives_1: 970.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

Epoch 24/40
4224/4224 [==============================] - 102s 24ms/step - loss: 0.2303 - accuracy: 0.9103 - auc_1: 0.9671 - false_positives_1: 2640.9063 - false_negatives_1: 3027.8310 - val_loss: 0.2514 - val_accuracy: 0.8968 - val_auc_1: 0.9691 - val_false_positives_1: 583.0000 - val_false_negatives_1: 2685.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

Epoch 25/40
4224/4224 [==============================] - 101s 24ms/step - loss: 0.2257 - accuracy: 0.9124 - auc_1: 0.9685 - false_positives_1: 2566.4918 - false_negatives_1: 2978.0791 - val_loss: 0.2172 - val_accuracy: 0.9169 - val_auc_1: 0.9713 - val_false_positives_1: 987.0000 - val_false_negatives_1: 1646.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

Epoch 26/40
4224/4224 [==============================] - 101s 24ms/step - loss: 0.2192 - accuracy: 0.9149 - auc_1: 0.9704 - false_positives_1: 2469.7122 - false_negatives_1: 2944.2717 - val_loss: 0.2488 - val_accuracy: 0.8998 - val_auc_1: 0.9717 - val_false_positives_1: 469.0000 - val_false_negatives_1: 2706.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

Epoch 27/40
4224/4224 [==============================] - 101s 24ms/step - loss: 0.2193 - accuracy: 0.9163 - auc_1: 0.9703 - false_positives_1: 2424.8847 - false_negatives_1: 2886.3917 - val_loss: 0.2197 - val_accuracy: 0.9156 - val_auc_1: 0.9707 - val_false_positives_1: 1512.0000 - val_false_negatives_1: 1162.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

Epoch 28/40
4224/4224 [==============================] - 101s 24ms/step - loss: 0.2138 - accuracy: 0.9184 - auc_1: 0.9720 - false_positives_1: 2367.0838 - false_negatives_1: 2800.8244 - val_loss: 0.2116 - val_accuracy: 0.9204 - val_auc_1: 0.9732 - val_false_positives_1: 1475.0000 - val_false_negatives_1: 1046.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1
```

```
Epoch 29/40
4224/4224 [==============================] - 102s 24ms/step - loss: 0.2089 - accuracy: 0.9211 - auc_1: 0.9731 -
false_positives_1: 2308.4963 - false_negatives_1: 2676.2561 - val_loss: 0.2135 - val_accuracy: 0.9199 - val_auc
_1: 0.9724 - val_false_positives_1: 972.0000 - val_false_negatives_1: 1566.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 30/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.2074 - accuracy: 0.9212 - auc_1: 0.9737 -
false_positives_1: 2289.9529 - false_negatives_1: 2698.6400 - val_loss: 0.2048 - val_accuracy: 0.9227 - val_auc
_1: 0.9746 - val_false_positives_1: 1320.0000 - val_false_negatives_1: 1130.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 31/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.2038 - accuracy: 0.9237 - auc_1: 0.9744 -
false_positives_1: 2222.0251 - false_negatives_1: 2626.9311 - val_loss: 0.2093 - val_accuracy: 0.9205 - val_auc
_1: 0.9759 - val_false_positives_1: 1702.0000 - val_false_negatives_1: 817.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 32/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.2006 - accuracy: 0.9244 - auc_1: 0.9753 -
false_positives_1: 2175.8959 - false_negatives_1: 2577.8542 - val_loss: 0.2039 - val_accuracy: 0.9215 - val_auc
_1: 0.9754 - val_false_positives_1: 827.0000 - val_false_negatives_1: 1659.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 33/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.1964 - accuracy: 0.9275 - auc_1: 0.9761 -
false_positives_1: 2113.1964 - false_negatives_1: 2494.2873 - val_loss: 0.1915 - val_accuracy: 0.9286 - val_auc
_1: 0.9774 - val_false_positives_1: 983.0000 - val_false_negatives_1: 1280.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 34/40
4224/4224 [==============================] - 107s 25ms/step - loss: 0.1961 - accuracy: 0.9273 - auc_1: 0.9761 -
false_positives_1: 2080.2133 - false_negatives_1: 2500.6551 - val_loss: 0.1955 - val_accuracy: 0.9275 - val_auc
_1: 0.9768 - val_false_positives_1: 1190.0000 - val_false_negatives_1: 1107.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

Epoch 35/40
4224/4224 [==============================] - 104s 25ms/step - loss: 0.1952 - accuracy: 0.9273 - auc_1: 0.9766 -
false_positives_1: 2085.1893 - false_negatives_1: 2475.1337 - val_loss: 0.1903 - val_accuracy: 0.9304 - val_auc
_1: 0.9777 - val_false_positives_1: 953.0000 - val_false_negatives_1: 1251.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available
metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_
positives_1,val_false_negatives_1
```

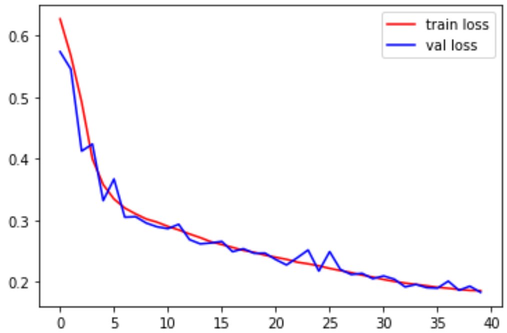
```
Epoch 36/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.1902 - accuracy: 0.9295 - auc_1: 0.9778 - false_positives_1: 2055.5792 - false_negatives_1: 2405.4514 - val_loss: 0.1894 - val_accuracy: 0.9322 - val_auc_1: 0.9779 - val_false_positives_1: 934.0000 - val_false_negatives_1: 1214.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

Epoch 37/40
4224/4224 [==============================] - 107s 25ms/step - loss: 0.1888 - accuracy: 0.9313 - auc_1: 0.9781 - false_positives_1: 2025.2159 - false_negatives_1: 2362.9778 - val_loss: 0.2008 - val_accuracy: 0.9244 - val_auc_1: 0.9778 - val_false_positives_1: 635.0000 - val_false_negatives_1: 1760.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

Epoch 38/40
4224/4224 [==============================] - 105s 25ms/step - loss: 0.1877 - accuracy: 0.9310 - auc_1: 0.9783 - false_positives_1: 1992.3794 - false_negatives_1: 2365.1082 - val_loss: 0.1859 - val_accuracy: 0.9322 - val_auc_1: 0.9795 - val_false_positives_1: 693.0000 - val_false_negatives_1: 1456.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

Epoch 39/40
4224/4224 [==============================] - 106s 25ms/step - loss: 0.1831 - accuracy: 0.9333 - auc_1: 0.9793 - false_positives_1: 1955.2826 - false_negatives_1: 2299.5938 - val_loss: 0.1926 - val_accuracy: 0.9287 - val_auc_1: 0.9797 - val_false_positives_1: 547.0000 - val_false_negatives_1: 1712.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

Epoch 40/40
4224/4224 [==============================] - 107s 25ms/step - loss: 0.1822 - accuracy: 0.9350 - auc_1: 0.9796 - false_positives_1: 1917.4911 - false_negatives_1: 2256.8551 - val_loss: 0.1826 - val_accuracy: 0.9338 - val_auc_1: 0.9798 - val_false_positives_1: 1011.0000 - val_false_negatives_1: 1087.0000
WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1

WARNING:tensorflow:Early stopping conditioned on metric `val_false_positives` which is not available. Available metrics are: loss,accuracy,auc_1,false_positives_1,false_negatives_1,val_loss,val_accuracy,val_auc_1,val_false_positives_1,val_false_negatives_1
```

Out[ ]:

```
<tensorflow.python.keras.callbacks.History at 0x7f64eb53f850>
```

In [ ]:

```
train_loss = model.history.history['loss']
val_loss = model.history.history['val_loss']
plt.plot(train_loss,color='red', label='train loss')
plt.plot(val_loss,color='blue', label='val loss')
plt.legend()
plt.show()

model.summary()

pd.DataFrame(model.history.history)
```



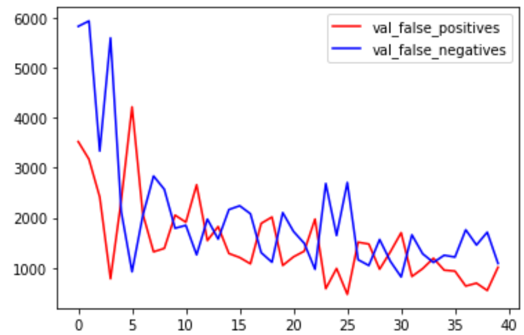

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 32)                2176
_________________________________________________________________
dense_4 (Dense)              (None, 12)                396
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 13
=================================================================
Total params: 2,585
Trainable params: 2,585
Non-trainable params: 0
_________________________________________________________________
```

Out[ ]:

	loss	accuracy	auc_1	false_positives_1	false_negatives_1	val_loss	val_accuracy	val_auc_1	val_false_positives_1	val_false_negatives_1
0	0.627154	0.646051	0.673559	6830.0	38020.0	0.573894	0.704852	0.757008	3523.0	5827.0
1	0.567375	0.705815	0.761462	14818.0	22459.0	0.545456	0.712586	0.784453	3170.0	5935.0
2	0.492175	0.760214	0.831897	12000.0	18384.0	0.412275	0.818523	0.891757	2415.0	3334.0
3	0.399346	0.824035	0.896188	9796.0	12501.0	0.423732	0.798699	0.917932	780.0	5597.0
4	0.357771	0.846117	0.918035	8997.0	10502.0	0.331792	0.858739	0.930372	2353.0	2122.0
5	0.334351	0.856763	0.928747	8512.0	9638.0	0.366871	0.837811	0.935511	4217.0	921.0
6	0.319715	0.863447	0.934992	8225.0	9078.0	0.304823	0.869882	0.941340	2078.0	2044.0
7	0.310161	0.867606	0.938951	8010.0	8766.0	0.305638	0.868841	0.944787	1320.0	2835.0
8	0.301833	0.872168	0.942309	7771.0	8427.0	0.295271	0.875122	0.947024	1388.0	2568.0
9	0.296685	0.875459	0.944346	7545.0	8236.0	0.289192	0.878689	0.947795	2052.0	1791.0
10	0.289830	0.878805	0.947030	7337.0	8020.0	0.286201	0.881309	0.948778	1911.0	1849.0
11	0.284133	0.881386	0.949149	7131.0	7899.0	0.293235	0.876259	0.951403	2660.0	1260.0
12	0.277563	0.884905	0.951501	6902.0	7682.0	0.268415	0.888917	0.955104	1545.0	1974.0
13	0.271549	0.889727	0.953672	6590.0	7383.0	0.261399	0.892705	0.958007	1825.0	1574.0
14	0.264829	0.893215	0.955989	6279.0	7252.0	0.262957	0.891095	0.957903	1286.0	2164.0
15	0.260377	0.895504	0.957489	6173.0	7068.0	0.265413	0.891221	0.957010	1206.0	2240.0
16	0.255691	0.897019	0.959088	6079.0	6970.0	0.248703	0.900249	0.962724	1080.0	2080.0
17	0.250819	0.900507	0.960663	5855.0	6752.0	0.253494	0.899271	0.961689	1888.0	1303.0
18	0.247675	0.901139	0.961662	5783.0	6744.0	0.246134	0.901323	0.965220	2015.0	1111.0
19	0.243039	0.903830	0.963109	5687.0	6499.0	0.246427	0.900565	0.963315	1047.0	2103.0
20	0.239595	0.905574	0.964215	5587.0	6378.0	0.235966	0.907005	0.965565	1219.0	1727.0
21	0.236240	0.907097	0.965254	5458.0	6314.0	0.227156	0.910919	0.968233	1333.0	1489.0
22	0.231644	0.909307	0.966720	5304.0	6188.0	0.238887	0.907068	0.968056	1974.0	970.0
23	0.228979	0.910893	0.967546	5247.0	6044.0	0.251441	0.896840	0.969101	583.0	2685.0
24	0.225721	0.912124	0.968498	5158.0	5977.0	0.217224	0.916885	0.971350	987.0	1646.0
25	0.221540	0.914736	0.969702	4934.0	5870.0	0.248807	0.899776	0.971749	469.0	2706.0
26	0.218021	0.916291	0.970618	4854.0	5753.0	0.219686	0.915591	0.970745	1512.0	1162.0
27	0.214706	0.918177	0.971556	4732.0	5636.0	0.211614	0.920420	0.973179	1475.0	1046.0
28	0.210709	0.920853	0.972610	4608.0	5421.0	0.213481	0.919884	0.972355	972.0	1566.0
29	0.207430	0.921571	0.973557	4559.0	5379.0	0.204826	0.922662	0.974608	1320.0	1130.0
30	0.203542	0.923457	0.974470	4447.0	5252.0	0.209288	0.920484	0.975866	1702.0	817.0
31	0.200395	0.925162	0.975211	4328.0	5155.0	0.203934	0.921525	0.975422	827.0	1659.0
32	0.197657	0.926464	0.975914	4288.0	5030.0	0.191484	0.928565	0.977429	983.0	1280.0
33	0.195498	0.927900	0.976446	4157.0	4979.0	0.195467	0.927491	0.976788	1190.0	1107.0
34	0.193236	0.928358	0.977018	4160.0	4918.0	0.190293	0.930427	0.977663	953.0	1251.0
35	0.190753	0.929715	0.977658	4078.0	4828.0	0.189418	0.932195	0.977852	934.0	1214.0
36	0.189223	0.930899	0.977946	4048.0	4708.0	0.200816	0.924398	0.977828	635.0	1760.0
37	0.187308	0.931949	0.978356	3926.0	4697.0	0.185920	0.932163	0.979516	693.0	1456.0
38	0.185758	0.932706	0.978778	3937.0	4590.0	0.192632	0.928691	0.979736	547.0	1712.0
39	0.185373	0.933385	0.978905	3895.0	4546.0	0.182593	0.933773	0.979763	1011.0	1087.0

In [ ]:

```
val_FP = model.history.history['val_false_positives_1']
val_FN = model.history.history['val_false_negatives_1']
plt.plot(val_FP,color='red',label='val_false_positives')
plt.plot(val_FN,color='blue',label='val_false_negatives')
plt.legend()
plt.show()
```



In [ ]:

```
###########################################################
####                  XGBoost                          ####
###########################################################
```

In [19]:

```
# Create test/train split of data
final_df = df2 # Use cleaned and imputated data from data section
x_train, x_test, y_train, y_test = train_test_split(final_df,y, test_size=0.20, random_state=123)
```

In [20]:

```
## XGB Parameter grid
param_grid = {
        'max_depth': [6, 10, 15, 20],
        'learning_rate': [0.001, 0.01, 0.1, 0.2, 0.3],
        'subsample': [0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
        'colsample_bytree': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
        'colsample_bylevel': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
        'min_child_weight': [0.5, 1.0, 3.0, 5.0, 7.0, 10.0],
        'gamma': [0, 0.25, 0.5, 1.0],
        'reg_lambda': [0.1, 1.0, 5.0, 10.0, 50.0, 100.0],
        'n_estimators': [200]}
```

In [21]:

```
## XGB Model building
# define function to do randomsearch
def xgb_randomsearch(xgb_df, xgb_y, xgb_x_test, xgb_y_test, n_iter, param_grid):
    # intiate XGBoost Classifier
    clf = xgb.XGBClassifier(use_label_encoder=False)
    # RandomSearchCV with custom scoring function
    xgb_rs_clf = RandomizedSearchCV(clf, param_grid, n_iter=n_iter, n_jobs=-1, verbose=2,cv=5, refit=True, rando
m_state=123, scoring='roc_auc')
    # return fit model
    fit_params = {'early_stopping_rounds':50, 'eval_set':[[xgb_x_test,xgb_y_test]], 'eval_metric':'auc'}
    return xgb_rs_clf.fit(xgb_df, xgb_y, **fit_params)
```

In [22]:

```
## XGB Model Fit
from datetime import datetime
start = datetime.now()
print(f'start time: {start}')
xgb_clf = xgb_randomsearch(xgb_df=x_train, xgb_y=y_train, xgb_x_test=x_test, xgb_y_test=y_test, n_iter=10, param
_grid=param_grid)
end = datetime.now()
print(f'end time: {end}')
print(f'total time(m): {end-start}')
```

```
start time: 2021-04-12 12:17:54.811640
Fitting 5 folds for each of 10 candidates, totalling 50 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed: 94.0min
[Parallel(n_jobs=-1)]: Done  50 out of  50 | elapsed: 109.8min finished
```

```
[0]	validation_0-auc:0.892483
Will train until validation_0-auc hasn't improved in 50 rounds.
[1]	validation_0-auc:0.932696
[2]	validation_0-auc:0.949706
[3]	validation_0-auc:0.956148
[4]	validation_0-auc:0.960744
[5]	validation_0-auc:0.964273
[6]	validation_0-auc:0.966017
[7]	validation_0-auc:0.96825
[8]	validation_0-auc:0.970722
[9]	validation_0-auc:0.972401
[10]	validation_0-auc:0.973797
[11]	validation_0-auc:0.974928
[12]	validation_0-auc:0.975898
[13]	validation_0-auc:0.976998
[14]	validation_0-auc:0.977888
[15]	validation_0-auc:0.97843
[16]	validation_0-auc:0.979111
[17]	validation_0-auc:0.979393
[18]	validation_0-auc:0.9797
[19]	validation_0-auc:0.980013
[20]	validation_0-auc:0.980227
[21]	validation_0-auc:0.980468
[22]	validation_0-auc:0.980698
[23]	validation_0-auc:0.980758
[24]	validation_0-auc:0.980715
[25]	validation_0-auc:0.98067
[26]	validation_0-auc:0.980968
[27]	validation_0-auc:0.980958
[28]	validation_0-auc:0.98117
[29]	validation_0-auc:0.981319
[30]	validation_0-auc:0.981232
[31]	validation_0-auc:0.981428
[32]	validation_0-auc:0.981542
[33]	validation_0-auc:0.981536
[34]	validation_0-auc:0.981674
[35]	validation_0-auc:0.981688
[36]	validation_0-auc:0.981661
[37]	validation_0-auc:0.981687
[38]	validation_0-auc:0.981905
[39]	validation_0-auc:0.982034
[40]	validation_0-auc:0.98213
[41]	validation_0-auc:0.982101
[42]	validation_0-auc:0.982147
[43]	validation_0-auc:0.98227
[44]	validation_0-auc:0.9823
[45]	validation_0-auc:0.982307
[46]	validation_0-auc:0.982308
[47]	validation_0-auc:0.982264
[48]	validation_0-auc:0.982296
[49]	validation_0-auc:0.982359
[50]	validation_0-auc:0.982358
[51]	validation_0-auc:0.982483
[52]	validation_0-auc:0.982536
[53]	validation_0-auc:0.982514
[54]	validation_0-auc:0.982565
[55]	validation_0-auc:0.982587
[56]	validation_0-auc:0.982594
[57]	validation_0-auc:0.982559
[58]	validation_0-auc:0.982604
[59]	validation_0-auc:0.982693
[60]	validation_0-auc:0.982673
[61]	validation_0-auc:0.982664
[62]	validation_0-auc:0.982667
[63]	validation_0-auc:0.982641
[64]	validation_0-auc:0.982676
[65]	validation_0-auc:0.98269
[66]	validation_0-auc:0.982717
[67]	validation_0-auc:0.982719
[68]	validation_0-auc:0.982711
[69]	validation_0-auc:0.982714
[70]	validation_0-auc:0.982767
[71]	validation_0-auc:0.982852
[72]	validation_0-auc:0.982925
[73]	validation_0-auc:0.982952
[74]	validation_0-auc:0.982993
[75]	validation_0-auc:0.982972
[76]	validation_0-auc:0.982983
[77]	validation_0-auc:0.983008
[78]	validation_0-auc:0.982981
[79]	validation_0-auc:0.983002
[80]	validation_0-auc:0.983006
[81]	validation_0-auc:0.982999
```

```
[82]	validation_0-auc:0.983039
[83]	validation_0-auc:0.983112
[84]	validation_0-auc:0.983167
[85]	validation_0-auc:0.983161
[86]	validation_0-auc:0.983154
[87]	validation_0-auc:0.98315
[88]	validation_0-auc:0.983129
[89]	validation_0-auc:0.983169
[90]	validation_0-auc:0.983154
[91]	validation_0-auc:0.983155
[92]	validation_0-auc:0.983148
[93]	validation_0-auc:0.983142
[94]	validation_0-auc:0.983105
[95]	validation_0-auc:0.983152
[96]	validation_0-auc:0.983152
[97]	validation_0-auc:0.98318
[98]	validation_0-auc:0.983148
[99]	validation_0-auc:0.98319
[100]	validation_0-auc:0.983226
[101]	validation_0-auc:0.983241
[102]	validation_0-auc:0.983267
[103]	validation_0-auc:0.983344
[104]	validation_0-auc:0.983369
[105]	validation_0-auc:0.983363
[106]	validation_0-auc:0.983392
[107]	validation_0-auc:0.983377
[108]	validation_0-auc:0.983429
[109]	validation_0-auc:0.983503
[110]	validation_0-auc:0.983535
[111]	validation_0-auc:0.983556
[112]	validation_0-auc:0.983551
[113]	validation_0-auc:0.983558
[114]	validation_0-auc:0.983548
[115]	validation_0-auc:0.983585
[116]	validation_0-auc:0.983573
[117]	validation_0-auc:0.983576
[118]	validation_0-auc:0.983589
[119]	validation_0-auc:0.98363
[120]	validation_0-auc:0.983588
[121]	validation_0-auc:0.983601
[122]	validation_0-auc:0.98362
[123]	validation_0-auc:0.983612
[124]	validation_0-auc:0.983647
[125]	validation_0-auc:0.983627
[126]	validation_0-auc:0.983638
[127]	validation_0-auc:0.983682
[128]	validation_0-auc:0.983729
[129]	validation_0-auc:0.983745
[130]	validation_0-auc:0.983786
[131]	validation_0-auc:0.983811
[132]	validation_0-auc:0.983805
[133]	validation_0-auc:0.983822
[134]	validation_0-auc:0.983823
[135]	validation_0-auc:0.983827
[136]	validation_0-auc:0.983817
[137]	validation_0-auc:0.983829
[138]	validation_0-auc:0.983824
[139]	validation_0-auc:0.983859
[140]	validation_0-auc:0.983835
[141]	validation_0-auc:0.983834
[142]	validation_0-auc:0.983843
[143]	validation_0-auc:0.983839
[144]	validation_0-auc:0.983836
[145]	validation_0-auc:0.983853
[146]	validation_0-auc:0.983858
[147]	validation_0-auc:0.983865
[148]	validation_0-auc:0.983857
[149]	validation_0-auc:0.98384
[150]	validation_0-auc:0.983841
[151]	validation_0-auc:0.983828
[152]	validation_0-auc:0.98381
[153]	validation_0-auc:0.983807
[154]	validation_0-auc:0.983829
[155]	validation_0-auc:0.983829
[156]	validation_0-auc:0.983862
[157]	validation_0-auc:0.983879
[158]	validation_0-auc:0.983904
[159]	validation_0-auc:0.983899
[160]	validation_0-auc:0.983893
[161]	validation_0-auc:0.983917
[162]	validation_0-auc:0.983914
[163]	validation_0-auc:0.983913
[164]	validation_0-auc:0.983914
```

```
[165]   validation_0-auc:0.983898
[166]   validation_0-auc:0.983886
[167]   validation_0-auc:0.983881
[168]   validation_0-auc:0.983874
[169]   validation_0-auc:0.983876
[170]   validation_0-auc:0.98387
[171]   validation_0-auc:0.983877
[172]   validation_0-auc:0.983887
[173]   validation_0-auc:0.983888
[174]   validation_0-auc:0.983887
[175]   validation_0-auc:0.983904
[176]   validation_0-auc:0.983909
[177]   validation_0-auc:0.98389
[178]   validation_0-auc:0.983879
[179]   validation_0-auc:0.983885
[180]   validation_0-auc:0.983878
[181]   validation_0-auc:0.983873
[182]   validation_0-auc:0.983859
[183]   validation_0-auc:0.983872
[184]   validation_0-auc:0.983904
[185]   validation_0-auc:0.983897
[186]   validation_0-auc:0.98389
[187]   validation_0-auc:0.983911
[188]   validation_0-auc:0.98393
[189]   validation_0-auc:0.983939
[190]   validation_0-auc:0.983954
[191]   validation_0-auc:0.98394
[192]   validation_0-auc:0.983948
[193]   validation_0-auc:0.983942
[194]   validation_0-auc:0.983946
[195]   validation_0-auc:0.983945
[196]   validation_0-auc:0.983949
[197]   validation_0-auc:0.983944
[198]   validation_0-auc:0.983942
[199]   validation_0-auc:0.98394
end time: 2021-04-12 14:12:59.570916
total time(m): 1:55:04.759276
```

In [23]:

```
xgb_clf.cv_results_
```

```
Out[23]: {'mean_fit_time': array([259.62258763, 318.32708993, 122.6055253 , 149.1768312 ,
                486.81900897, 390.34630032, 353.35170884, 280.56320696,
                146.02923722, 102.53373694]),
          'mean_score_time': array([0.79429202, 1.91519499, 1.10696845, 1.40457764, 3.05394487,
                1.76340699, 1.2852448 , 1.24809475, 1.11367536, 0.37092886]),
          'mean_test_score': array([0.93871262, 0.97167068, 0.97734763, 0.95452596, 0.98199754,
                0.98202057, 0.96821098, 0.96535591, 0.97841496, 0.90395791]),
          'param_colsample_bylevel': masked_array(data=[1.0, 0.6, 0.4, 0.4, 0.5, 0.5, 0.9, 0.9, 0.4, 0.5],
                      mask=[False, False, False, False, False, False, False, False,
                            False, False],
                fill_value='?',
                     dtype=object),
          'param_colsample_bytree': masked_array(data=[0.6, 0.6, 0.5, 0.5, 0.8, 0.8, 0.7, 0.6, 0.6, 0.9],
                      mask=[False, False, False, False, False, False, False, False,
                            False, False],
                fill_value='?',
                     dtype=object),
          'param_gamma': masked_array(data=[1.0, 0, 0.5, 0, 0, 0, 1.0, 1.0, 0.5, 1.0],
                      mask=[False, False, False, False, False, False, False, False,
                            False, False],
                fill_value='?',
                     dtype=object),
          'param_learning_rate': masked_array(data=[0.001, 0.01, 0.1, 0.001, 0.1, 0.3, 0.01, 0.01, 0.2,
                            0.001],
                      mask=[False, False, False, False, False, False, False, False,
                            False, False],
                fill_value='?',
                     dtype=object),
          'param_max_depth': masked_array(data=[10, 15, 10, 15, 20, 20, 10, 10, 10, 6],
                      mask=[False, False, False, False, False, False, False, False,
                            False, False],
                fill_value='?',
                     dtype=object),
          'param_min_child_weight': masked_array(data=[5.0, 3.0, 7.0, 0.5, 0.5, 3.0, 1.0, 0.5, 10.0, 1.0],
                      mask=[False, False, False, False, False, False, False, False,
                            False, False],
                fill_value='?',
                     dtype=object),
          'param_n_estimators': masked_array(data=[200, 200, 200, 200, 200, 200, 200, 200, 200, 200],
                      mask=[False, False, False, False, False, False, False, False,
                            False, False],
                fill_value='?',
                     dtype=object),
          'param_reg_lambda': masked_array(data=[50.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 5.0,
                            10.0],
                      mask=[False, False, False, False, False, False, False, False,
                            False, False],
                fill_value='?',
                     dtype=object),
          'param_subsample': masked_array(data=[0.7, 0.8, 0.9, 0.5, 0.7, 0.9, 0.8, 0.5, 0.8, 0.5],
                      mask=[False, False, False, False, False, False, False, False,
                            False, False],
                fill_value='?',
                     dtype=object),
          'params': [{'colsample_bylevel': 1.0,
            'colsample_bytree': 0.6,
            'gamma': 1.0,
            'learning_rate': 0.001,
            'max_depth': 10,
            'min_child_weight': 5.0,
            'n_estimators': 200,
            'reg_lambda': 50.0,
            'subsample': 0.7},
           {'colsample_bylevel': 0.6,
            'colsample_bytree': 0.6,
            'gamma': 0,
            'learning_rate': 0.01,
            'max_depth': 15,
            'min_child_weight': 3.0,
            'n_estimators': 200,
            'reg_lambda': 10.0,
            'subsample': 0.8},
           {'colsample_bylevel': 0.4,
            'colsample_bytree': 0.5,
            'gamma': 0.5,
            'learning_rate': 0.1,
            'max_depth': 10,
            'min_child_weight': 7.0,
            'n_estimators': 200,
            'reg_lambda': 10.0,
            'subsample': 0.9},
           {'colsample_bylevel': 0.4,
            'colsample_bytree': 0.5,
```

```
  'gamma': 0,
  'learning_rate': 0.001,
  'max_depth': 15,
  'min_child_weight': 0.5,
  'n_estimators': 200,
  'reg_lambda': 10.0,
  'subsample': 0.5},
 {'colsample_bylevel': 0.5,
  'colsample_bytree': 0.8,
  'gamma': 0,
  'learning_rate': 0.1,
  'max_depth': 20,
  'min_child_weight': 0.5,
  'n_estimators': 200,
  'reg_lambda': 10.0,
  'subsample': 0.7},
 {'colsample_bylevel': 0.5,
  'colsample_bytree': 0.8,
  'gamma': 0,
  'learning_rate': 0.3,
  'max_depth': 20,
  'min_child_weight': 3.0,
  'n_estimators': 200,
  'reg_lambda': 10.0,
  'subsample': 0.9},
 {'colsample_bylevel': 0.9,
  'colsample_bytree': 0.7,
  'gamma': 1.0,
  'learning_rate': 0.01,
  'max_depth': 10,
  'min_child_weight': 1.0,
  'n_estimators': 200,
  'reg_lambda': 10.0,
  'subsample': 0.8},
 {'colsample_bylevel': 0.9,
  'colsample_bytree': 0.6,
  'gamma': 1.0,
  'learning_rate': 0.01,
  'max_depth': 10,
  'min_child_weight': 0.5,
  'n_estimators': 200,
  'reg_lambda': 10.0,
  'subsample': 0.5},
 {'colsample_bylevel': 0.4,
  'colsample_bytree': 0.6,
  'gamma': 0.5,
  'learning_rate': 0.2,
  'max_depth': 10,
  'min_child_weight': 10.0,
  'n_estimators': 200,
  'reg_lambda': 5.0,
  'subsample': 0.8},
 {'colsample_bylevel': 0.5,
  'colsample_bytree': 0.9,
  'gamma': 1.0,
  'learning_rate': 0.001,
  'max_depth': 6,
  'min_child_weight': 1.0,
  'n_estimators': 200,
  'reg_lambda': 10.0,
  'subsample': 0.5}],
'rank_test_score': array([ 9,  5,  4,  8,  2,  1,  6,  7,  3, 10], dtype=int32),
'split0_test_score': array([0.94000969, 0.97229672, 0.97876297, 0.95646026, 0.98280674,
       0.98291807, 0.96947219, 0.96621011, 0.97904023, 0.90711334]),
'split1_test_score': array([0.93767458, 0.97111717, 0.97676745, 0.95436787, 0.98132879,
       0.98159178, 0.96771752, 0.96451019, 0.97822754, 0.90557239]),
'split2_test_score': array([0.93732013, 0.97032286, 0.97665505, 0.95359538, 0.98111854,
       0.98091739, 0.96712405, 0.96391469, 0.97772116, 0.90412108]),
'split3_test_score': array([0.93971891, 0.97300685, 0.97786441, 0.95415279, 0.98317307,
       0.98331237, 0.96896587, 0.96677683, 0.97961933, 0.90035354]),
'split4_test_score': array([0.93883979, 0.97160979, 0.97668825, 0.95405348, 0.98156056,
       0.98136325, 0.96777527, 0.96536775, 0.97746655, 0.9026292 ]),
'std_fit_time': array([80.97005587,  1.57475068,  0.60294241, 29.03005187,  5.53122667,
        1.06234729,  2.28224488,  3.32170027,  1.3462637 , 25.02257013]),
'std_score_time': array([0.3611693 , 0.10242913, 0.01778442, 0.65867111, 0.36131288,
       0.05162279, 0.04496383, 0.06329287, 0.01477368, 0.13184871]),
'std_test_score': array([0.00107031, 0.00092779, 0.0008392 , 0.0009995 , 0.00083036,
       0.00092814, 0.00086894, 0.00105273, 0.00080715, 0.00233867])}
```

In [33]:

```
# predict value for y on whole data set
xgb_pred = xgb_clf.predict(final_df)
```

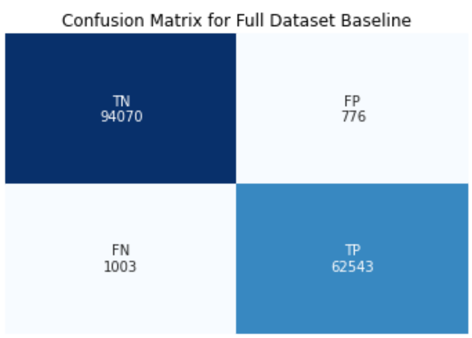
In [34]:
```
### XGB True Cost of FP and FN

# create a custom cost function
def cost_function(y_actual, y_pred):
    CM = confusion_matrix(y_actual, y_pred)
    FP = CM[0][1]
    FN = CM[1][0]
    Cost = (FP*225)+(FN*35)
    return CM, FP, FN, Cost
```

In [35]:
```
# use custom cost function to find total cost
CM, FP, FN, cost = cost_function(y, xgb_pred)
```

In [ ]:
```
import locale
locale.setlocale(locale.LC_ALL, 'en_US')
print(f'False Positives: {FP}')
print(f'False Negatives: {FN}')
print(f'Total Cost: {locale.currency(cost, grouping=True)}')
```

In [36]:
```
### XGB Confusion Matrix
```

In [38]:
```
group_names = ['TN','FP','FN','TP']
group_counts = CM.flatten()
labels = [f'{v1}\n{v2}' for v1, v2 in zip(group_names, group_counts)]
labels = np.asarray(labels).reshape(2,2)
matrix = sns.heatmap(CM, annot=labels, fmt='',cmap='Blues',cbar=False, xticklabels=False, yticklabels=False).set
_title("Confusion Matrix for Full Dataset Baseline")
```



In [40]:
```
from sklearn.metrics import mean_squared_error as MSE
xgb_rmse = np.sqrt(MSE(y, xgb_pred))
xgb_accuracy = accuracy_score(y, xgb_pred)
```

In [41]:
```
print(xgb_rmse)
```

```
0.10597937469146027
```

In [ ]:
```
### Feature Importance
from sklearn.inspection import permutation_importance

# use permutation_importance() to calculate feature importance of fitted model against test dataset
perm_importance = permutation_importance(xgb_clf, x_test, y_test, n_repeats=30, random_state=123)

# Get top 10 features based on importance
feature_importance = pd.DataFrame(perm_importance.importances_mean, index=x_test.columns, columns=['importance'
]).nlargest(10,['importance'])
```

In [ ]:
```
# plot feature importance
feature_importance.plot(kind='bar')
plt.xlabel("feature")
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```
##########################################################
####                 Random Forest                    ####
##########################################################
```

In [ ]:
```python
import pandas as pd
import xgboost as xgb
import os
import time
import numpy as np
from sklearn.metrics import log_loss, accuracy_score
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from tabulate import tabulate
import pickle

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV

from sklearn.tree import export_graphviz
from pprint import pprint

import math
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
from sklearn import model_selection

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.svm import LinearSVC
import sklearn.feature_selection as fs
from sklearn.model_selection import cross_val_score

from sklearn.metrics import confusion_matrix, classification_report
```

In [ ]:
```python
#from sklearn.ensemble import RandomForestClassifier

# Full Dataset used for Base Model.
# Default max_levels is None, so the tree is very large

rf_base = RandomForestClassifier(n_estimators=10, random_state=123 )

start = time.time()
rf_base.fit(x_train, y_train)

end = time.time()
rf_base_time=round((end-start),2)
rf_base_time
```

Out[ ]: 8.99

In [ ]:
```python
rf_base_preds = rf_base.predict_proba(x_test)
```

In [ ]:
```python
rf_base_log_loss = log_loss(y_test,rf_base_preds[:,1]) # each column is class probability,
print(rf_base_log_loss)
rf_base_accuracy = accuracy_score(y_test,np.rint(rf_base_preds[:,1]))
print(rf_base_accuracy)
```

```
0.36447004231209096
0.8957037785283627
```

In [ ]:
```python
# Generate predictions and calculate Confusion Matrix CM
# Set target names for classification report output

y_pred = np.rint(rf_base_preds[:,1])
CM = confusion_matrix(y_test,y_pred)
target_names = ['class 0', 'class 1']
```

In [ ]:
```python
# RF Model Performance Metrics

print(classification_report(y_test, y_pred, target_names = target_names, digits=4))
```

```
              precision    recall  f1-score   support

     class 0     0.8834    0.9511    0.9160     18938
     class 1     0.9180    0.8134    0.8625     12741

    accuracy                         0.8957     31679
   macro avg     0.9007    0.8822    0.8892     31679
weighted avg     0.8973    0.8957    0.8945     31679
```

In [ ]:
```
# Evaluate False Positives, False Negatives and Calculate Total Cost

FP = CM[0][1]
FN = CM[1][0]
total_cost = (FP*225)+(FN*35)
print("False Positives =",FP)
print("False Negatives =",FN)
print("Total Cost =",total_cost)
```

```
False Positives = 926
False Negatives = 2378
Total Cost = 291580
```

In [ ]:
```
#########################################################
#              Begin Hypertuning for Random Forest.     #
#########################################################
```

In [ ]:
```
# Citation:  Thank you to Will Koehrsen and Towards Data Science. RF Hypertuning code is borrowed from his examp
le.
# https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd
74

# Note - different from tutorial, we are using rf=RandomForestClassifier(), not RandomForestRegressor()
from pprint import pprint
# Look at parameters used by our current forest
print('Parameters Used by Base Model:\n')
pprint(rf_base.get_params())
```

```
Parameters Used by Base Model:

{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 10,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 123,
 'verbose': 0,
 'warm_start': False}
```

In [ ]:
```python
# Define Random Grid Parameters and use RandomizedSearchCV to choose
# different combinations of parameters for different sizes of datasets

#from sklearn.model_selection import RandomizedSearchCV
# Number of trees in random forest
n_estimators = [10,100,500,1000,1500]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [10, 100, 200,300,400,500,600,700,800,900,1000]
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
print('Random Grid Parameters:\n')
pprint(random_grid)
```

```
Random Grid Parameters:

{'bootstrap': [True, False],
 'max_depth': [10, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [10, 100, 500, 1000, 1500]}
```

In [ ]:
```python
# Adapt variable names for this example
train_features = x_train
train_labels = y_train
test_features = x_test
test_labels = y_test
print('Training Features Shape:', train_features.shape)
print('Training Labels Shape:', train_labels.shape)
print('Testing Features Shape:', test_features.shape)
print('Testing Labels Shape:', test_labels.shape)
```

```
Training Features Shape: (126713, 67)
Training Labels Shape: (126713,)
Testing Features Shape: (31679, 67)
Testing Labels Shape: (31679,)
```

In [ ]:
```python
# Original Code Commented - Using Pickled Models

# Use the random grid to search for best hyperparameters
# Define estimator - same parameters as base model

rf = RandomForestClassifier(n_estimators=50, random_state=123 )
```

In [ ]:
```python
# Create the FULL model based on the random_grid parameters
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid, n_iter = 5, cv = 5, verbose=2,
random_state=123, n_jobs = -1)
```

In [ ]:
```python
# Commented to prevent running best model - three hour processing time.
# Fit a model on all data using the same random_grid parameters
# rf_random_all = rf_random.fit(train_features, train_labels)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
```

In [ ]:
```python
rf_random_all_preds = rf_random_all.predict_proba(x_test)
```

In [ ]:
```python
y_all_pred = np.rint(rf_random_all_preds[:,1])
CM = confusion_matrix(y_test,y_all_pred)
target_names = ['class 0', 'class 1']
```

In [ ]:
```python
# RF Model Performance Metrics

print(classification_report(y_test, y_all_pred, target_names = target_names, digits=4))
```

In [ ]:

```
# # Evaluate False Positives, False Negatives and Calculate Total Cost

FP = CM[0][1]
FN = CM[1][0]
total_cost = (FP*225)+(FN*35)
print("False Positives =",FP)
print("False Negatives =",FN)
print("Total Cost =",total_cost)
```

In [ ]:

```
## End Random Forest
```