FRAUD DETECTION ON CREDIT CARDS

MILESTONE: PROJECT REPORT
GROUP 17

Student 1: PREETHAM REDDY GOLLAPALLI
Student 2: TANAY PARIKH

857(246)-8433(Tel of student 1)
857(313)-4587(Tel of student 2)

gollapalli.p@northeastern.edu(mail id of student 1)
parikh.t@northeastern.edu(mail id of student 2)

Percentage of Effort Contributed by Student 1: 50%
Percentage of Effort Contributed by Student 2: 50%

Signature of Student 1: Preetham Reddy Gollapalli
Signature of Student 2: Tanay Parikh

Submission Date: 04/23/2023

ABSTRACT

A rising issue, credit card theft costs financial institutions and customers billions of dollars annually. We created a fraud detection system that employs machine learning algorithms to spot possibly fraudulent transactions in real-time in order to solve this issue. Our method is built to increase the identification of true fraud while minimizing the number of false positives (legal transactions marked as fraudulent). We used a publicly accessible dataset of credit card transactions, which includes both valid and fraudulent transactions, to train the algorithm. For the purpose of locating the top-performing machine learning models, we carried out exploratory data analysis, feature engineering, and model training. To improve performance, we also tested several various model architectures and hyperparameters.

Our findings demonstrate that our fraud detection technology has a low false positive rate and can accurately identify fraudulent transactions in real-time. Our technology is also adaptive to shifting fraud tendencies and continues to function well over time thanks to ongoing oversight and input from analysts and investigators. Our proposed solution involves utilizing various machine learning techniques such as Random Forest, Naive Bayes, Logistic Regression, Neural Nets, and Support Vector Machines to model and classify the given dataset. By training a well-generalized model, we can achieve reasonable accuracy in website classification. Additionally, we will incorporate dimension reduction methods such as Recursive feature elimination, and other methods such as correlation matrix, and cardinality of categorical variables to simplify the dataset. To determine the models' accuracy and robustness, we will employ evaluation metrics like Lift Chart, ROC curve, Precision, Recall, and F-1 score during testing.

Due to the fact that it offers a dependable and efficient method for identifying and combating credit card fraud, our project has significant consequences for financial institutions and cardholders.

I. BACKGROUND AND INTRODUCTION

Problem Setting

Fraud detection is a collection of actions performed to stop the acquisition of money or property under false pretenses. As repeated tactics are frequently used in fraud, looking for patterns is a common strategy for fraud detection. It is a collection of actions performed to identify and thwart fraudsters' attempts to steal money or property through deception. Fraud detection is a common practice in law enforcement organizations as well as the banking, insurance, medical, government, and public sectors. Businesses might include fraud detection in their websites, corporate guidelines, employee development programs, and improved security features. In this project, we'll concentrate on the credit card industry fraud that takes place.

Problem Definition

Using someone else's credit card without their knowledge or agreement to make purchases or apply for cash advances is known as credit card fraud. The card itself may be taken by these criminals through physical theft, but increasingly, fraudsters are using digital methods to acquire the credit card number and the associated personal data to carry out nefarious transactions. We are trying to build a model as data engineers that categorizes if a transaction is fraudulent or not and makes predictions. Now that we've considered credit card information, we can determine whether there has been fraud. We intend to address the two fundamental problems in credit card fraud detection, namely data skewness and cost sensitivity.

Our solution

Our proposed solution involves utilizing various machine learning techniques such as Random Forest, Naive Bayes, Logistic Regression, Neural Nets, and Support Vector Machines to model and classify the given dataset. By training a well-generalized model, we can achieve reasonable accuracy in website classification. Additionally, we will incorporate dimension reduction methods such as Recursive feature elimination, and other methods such as correlation matrix, and cardinality of categorical variables to simplify the dataset. To determine the models' accuracy and robustness, we will employ evaluation metrics like Lift Chart, ROC curve, Precision, Recall, and F-1 score during testing.

II. DATA EXPLORATION AND VISUALIZATION

Data Sources

The dataset used for this research is collected from Kaggle. This is a simulated credit card transaction dataset containing legitimate and fraudulent transactions from the duration of 1st Jan 2019 - 31st Dec 2020. It covers the credit cards of 1000 customers doing transactions with a pool of 800 merchants. The data has already been divided into test and train where the size of the testing dataset is 352 MB whereas for the testing it is 150 MB. The above code tells the shape of the training dataset. So, the number of rows is 1296675, whereas the number of columns is 23. Here our response variable is “is_fraud”. The dataset has categorized the value of fraud transactions as 1 and normal transactions as 0. Now we checked for the null values, and we found out that there are no null values in the dataset.

Data Collection

These are the different columns-

```
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 22 columns):
 #   Column      Non-Null Count    Dtype
---  ------      --------------    -----
 0   Unnamed: 0  1048575 non-null  int64
 1   Unnamed: 1  0 non-null        float64
 2   cc_num      1048575 non-null  float64
 3   merchant    1048575 non-null  object
 4   category    1048575 non-null  object
 5   amt         1048575 non-null  float64
 6   first       1048575 non-null  object
 7   last        1048575 non-null  object
 8   gender      1048575 non-null  object
 9   street      1048575 non-null  object
 10  city        1048575 non-null  object
 11  state       1048575 non-null  object
 12  zip         1048575 non-null  int64
 13  lat         1048575 non-null  float64
 14  long        1048575 non-null  float64
 15  city_pop    1048575 non-null  int64
 16  job         1048575 non-null  object
 17  merch_long  1048575 non-null  float64
 18  trans_num   1048575 non-null  object
 19  unix_time   1048575 non-null  int64
```

20 merch_lat   1048575 non-null float64
21 is_fraud   1048575 non-null int64
dtypes: float64(7), int64(5), object(10)
memory usage: 176.0+ MB

Read Csv file:

```
[8]: df = pd.read_csv(r"/Users/tanayparikh/Downloads/fraudTrain.csv")
df
```

t[8]:

cc_num	merchant	category	amt	first	last	gender	street	...	zip	lat	long	city_pop	job	merch_long
703190e+15	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	561 Perry Cove	...	28654	36.0788	-81.1781	3495	Psychologist, counselling	-82.048315
304230e+11	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	43039 Riley Greens Suite 393	...	99160	48.8878	-118.2105	149	Special educational needs teacher	-118.186462
885950e+13	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 White Dale Suite 530	...	83252	42.1808	-112.2620	4154	Nature conservation officer	-112.154481
534090e+15	fraud_Kutch, Hermiston and	gas_transport	45.00	Jeremy	White	M	9443 Cynthia	...	59632	46.2306	-112.1138	1939	Patent attorney	-112.561071

1)Checked weather there were missing/Duplicate values or not-

```
In [13]: percent_missing = df.isnull().sum() * 100 / len(df)
         missing_value_df = pd.DataFrame({'column_name': df.columns,
                                          'percent_missing': percent_missing})
         print (percent_missing)

Unnamed: 0       0.0
Unnamed: 1     100.0
cc_num           0.0
merchant         0.0
category         0.0
amt              0.0
first            0.0
last             0.0
gender           0.0
street           0.0
city             0.0
state            0.0
zip              0.0
lat              0.0
long             0.0
city_pop         0.0
job              0.0
merch_long       0.0
trans_num        0.0
unix_time        0.0
merch_lat        0.0
is_fraud         0.0
dtype: float64
```

2) Values of Fraud and Genuine

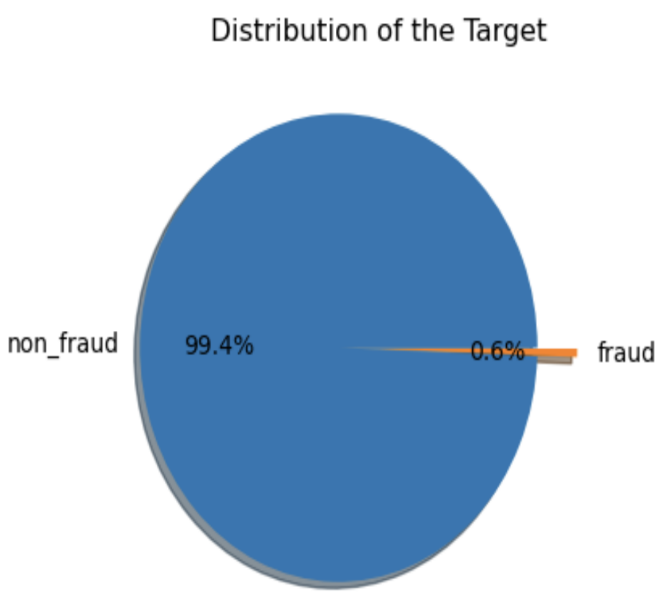
In [16]:
```
a = df['is_fraud'].value_counts().rename('count')
b = (df['is_fraud'].value_counts(normalize = True)*100).rename('distribution')
tem = pd.concat([a,b], axis = 1)
tem.index = ['genuine','fraud']
tem['distribution'].plot(kind = 'bar', figsize = [4,4]);
tem
```

Out[16]:

	count	distribution
genuine	1042569	99.427223
fraud	6006	0.572777

3) Pie Chart of Target Variable

In [21]:
```
plt.figure(figsize = [4,4])
plot_var = df['is_fraud'].value_counts(normalize = True)
plt.pie(plot_var,
        autopct='%1.1f%%',
        labels = ['non_fraud','fraud'],
        explode = [0.2, 0],
        shadow = True)
plt.title('Distribution of the Target');
```



The classes are very unbalanced; 99.5% of them are for legitimate transactions, while just 0.5% of them are used for fraudulent activities.

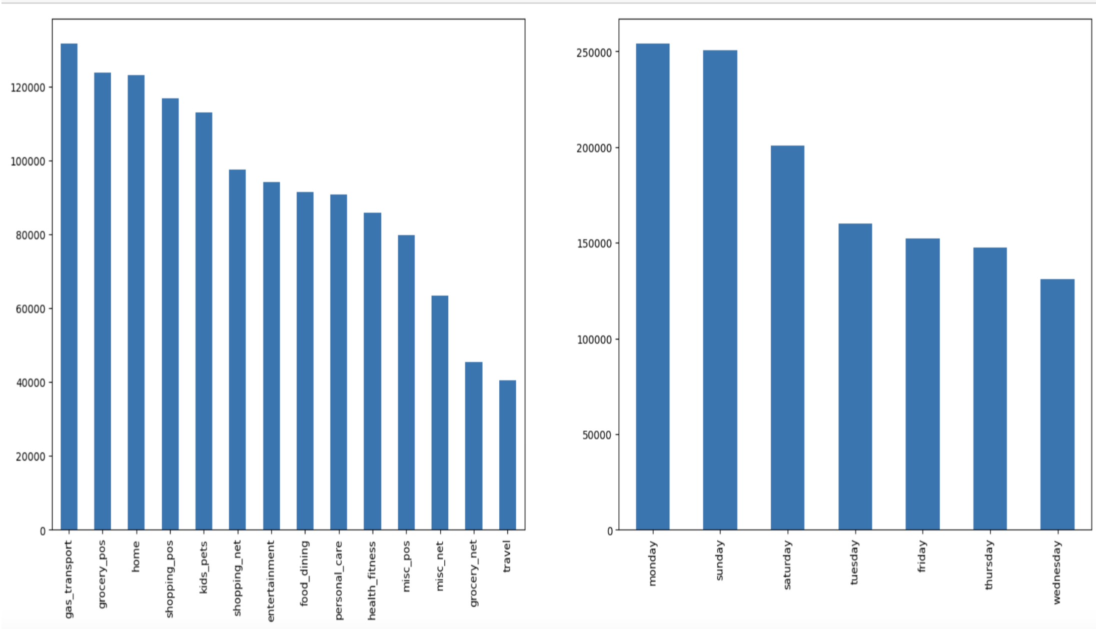
4) Heat Map

•The variable unix_time is positively correlated with the cc_num and has a very high negative correlation with the is_fraud. This suggests that there may be some relationship between the time of the transaction and the credit card number used, and that fraudulent transactions tend to occur at specific times.

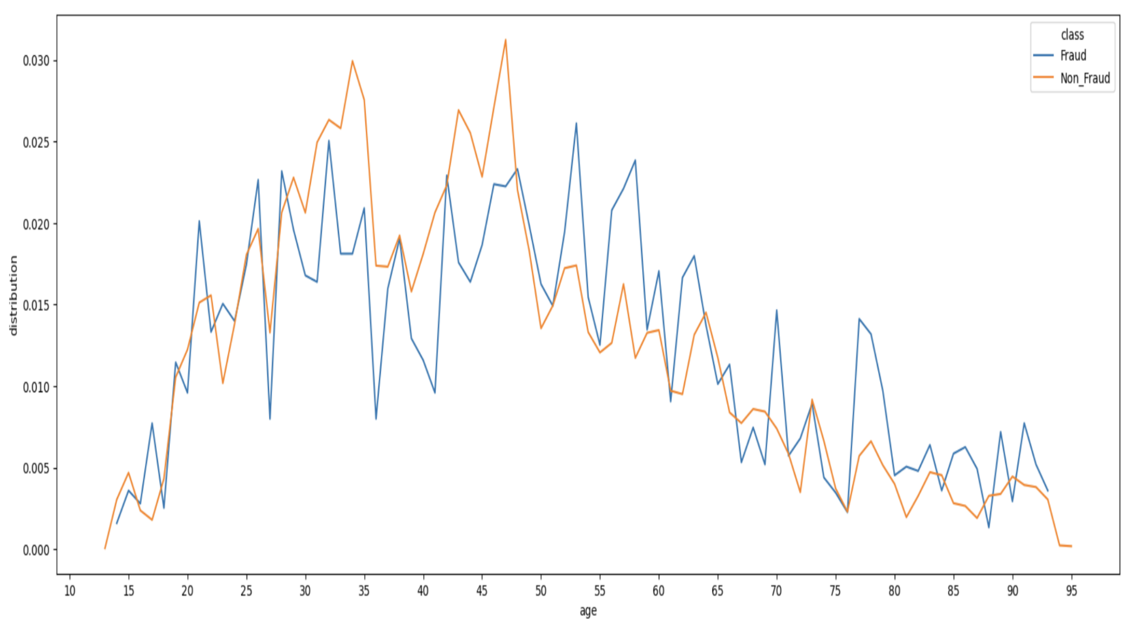
•The amt variable has a moderate positive correlation with the is_fraud variable, which suggests that the amount of the transaction may be a useful predictor of fraudulent activity.

•The lat and long variables have a weak correlation with the is_fraud variable, while they are highly correlated with merch_lat and merch_long. This suggests that the location of the merchant may be more important in predicting fraudulent activity than the location of the transaction.

5) Transaction vs Categories and Transaction vs on weekdays Bar chart

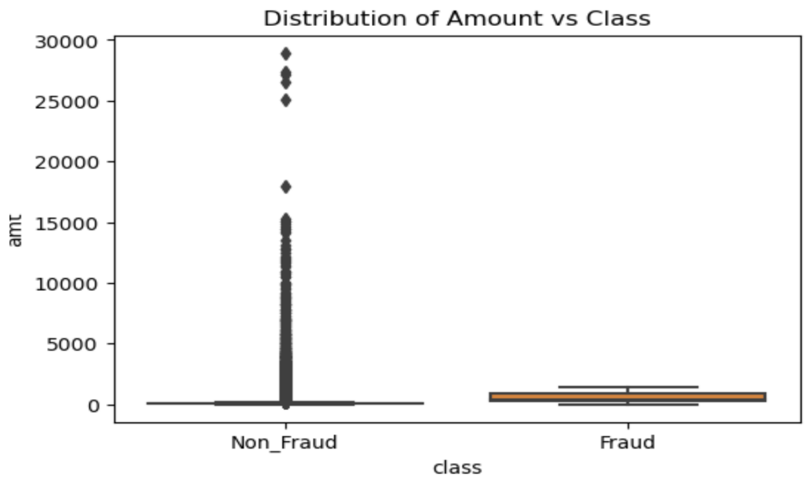


6) Line Plot graph pf Fraud vs Non-Fraud(Genuine)



The majority of the transactions come from people between the ages of 30 and 50. The majority of Fraad transactions are made by cardholders between the ages of 45 and 60.

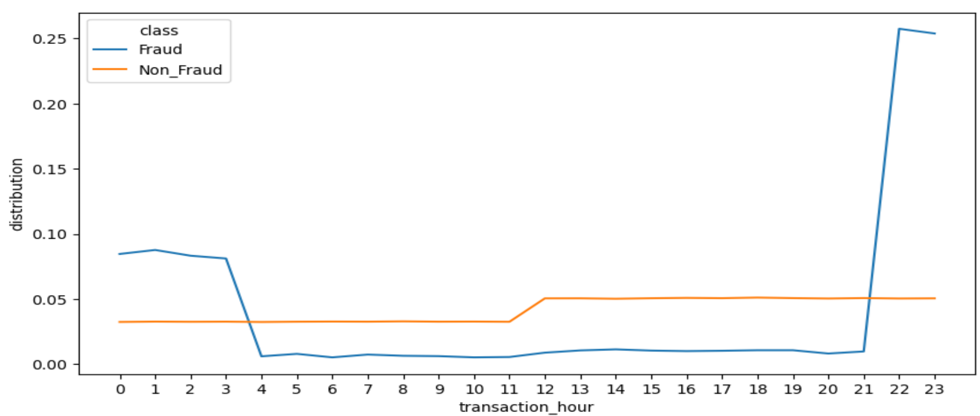
7) Box Plot



The boxplot clearly shows that the fraud transactions do not have outlier amounts, but rather that the majority are heavily concentrated with a median of 390, which is very high compared to the normal transactions, with the mean value of the fraud transactions appearing to be high at 530 dollars.

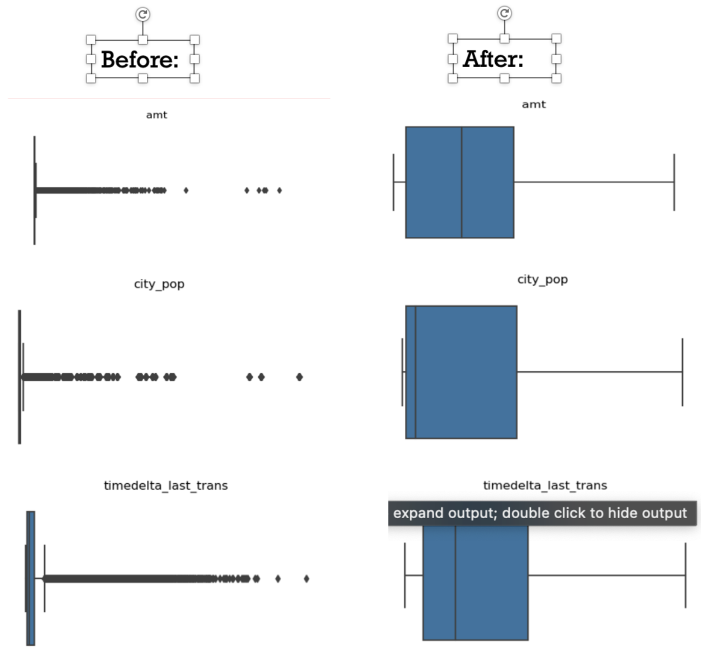
8) Line plot

```
In [57]: plt.figure(figsize = [10,5])

         trans_hour_distribution = all.groupby('class')['transaction_hour'].value_counts(normalize = True).rename('distribution').reset_i

         sns.lineplot(data = trans_hour_distribution, x = 'transaction_hour', y = 'distribution', hue = 'class')
         plt.xticks(np.arange(0,24,1))

         plt.show()
```



The distribution of normal transactions is comparable across the hours, increasing slightly from the eleventh hour and remaining constant until the twenty-third hour. Most fraudulent transactions occur between the hours of 21 and 4. In other words, fraudulent transactions take place at nighttime, while legitimate cardholders are asleep and unable to get notifications of the transactions.

III.Data Preparation And Processing



1) Outliers removal

Outliers exist in the following variables: amt, city_pop, and timedelta_last_trans; some class members may be impacted. Outliers were removed using the Winzorizer class.

2) One Hot Encoding-

```
from feature_engine.encoding import OneHotEncoder

variable = ["category","gender"]
onehot_encod = OneHotEncoder(variables = variable, drop_last = True)
onehot_encod.fit(df)
```

```
OneHotEncoder(drop_last=True, variables=['category', 'gender'])
```

```
df = onehot_encod.transform(df)
df.head()
```

th	transaction_year	...	category_shopping_net	category_food_dining	category_personal_care	category_grocery_pos	category_entertainment	category_shopping_p
1	2019	...	0	0	0	0	0	
1	2019	...	0	0	0	0	0	
1	2019	...	0	0	0	0	0	
1	2019	...	0	0	0	0	0	

We used one hot encoding for category and gender

3) Mean Encoding

```
In [70]: from feature_engine.encoding import MeanEncoder

         variables = ['state','transaction_day','job']
         mean_encod = MeanEncoder(variables = variables)
         mean_encod.fit(df,y = df["is_fraud"])

Out[70]: MeanEncoder(variables=['state', 'transaction_day', 'job'])
```

We have done mean encoding for state, transaction_day, job.

4) Recursive feature extraction

```
In [73]: from sklearn.feature_selection import RFE
         from sklearn.linear_model import LogisticRegression
```

```
In [74]: model = LogisticRegression()

         # Create an RFE selector and fit to the data
         rfe = RFE(model, n_features_to_select=23)
         rfe.fit(df, df['is_fraud'])


         # Print the selected features
         print("Selected Features: ")
         print(df.columns[rfe.support_])
```

```
Selected Features:
Index(['state', 'job', 'is_fraud', 'transaction_hour', 'transaction_month',
       'lat_dist_cust_merch', 'long_dist_cust_merch', 'lat_dist_prev_merch',
       'long_dist_prev_merch', 'category_misc_net', 'category_gas_transport',
       'category_kids_pets', 'category_home', 'category_shopping_net',
       'category_food_dining', 'category_personal_care',
       'category_grocery_pos', 'category_entertainment',
       'category_shopping_pos', 'category_misc_pos', 'category_travel',
       'category_health_fitness', 'gender_f'],
      dtype='object')
```

IV. Data Mining Techniques and Implementation

We then use the following algorithms for classification –

1. Random Forest

2. Decision Tree Classifier

3. Logistic Regression

4. Naïve Bayes Classifier

5. LinearDiscriminantAnalysis

6. Artificial Neural Network

7. Support Vector Machine

In [92]:
```
from sklearn.neighbors import KNeighborsClassifier
def confusion(X_train, y_train, X_valid, y_valid, model):
    model.fit(X_train, y_train)
    pred = model.predict(X_valid)

    cm = confusion_matrix(y_valid, pred)
    plt.figure(figsize=(10,6))
    sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
    print(classification_report(y_valid, pred), '\n')

    RocCurveDisplay.from_estimator(estimator = model, X = X_valid, y = y_valid)

    f_1 = dict([("f1_score_binary", f1_score(y_valid, pred, average="binary")),
                 ("f1_score_micro", f1_score(y_valid, pred, average="micro")),
                ("f1_score_macro", f1_score(y_valid, pred, average="macro")),
                ("f1_score_weighted", f1_score(y_valid, pred, average="weighted"))
                 ])
    mcc = matthews_corrcoef(y_valid, pred)

    tn, fp, fn, tp = confusion_matrix(y_valid, pred).ravel()
    sensitivity = tp / (tp + fn)
    specificity = tn / (tn + fp)
    print("Sensitivity: ", sensitivity)
    print("Specificity: ", specificity)

    try:
        auc = roc_auc_score(y_valid, model.predict_proba(X_valid)[:,1])
        skplt.metrics.plot_cumulative_gain(y_valid, model.predict_proba(X_valid))
    except AttributeError:
        auc = None
        print("predict_proba is not available when probability=False")
    except Exception as e:
        auc = None
        print(e)

    plt.show()
    return f_1, mcc, auc
```

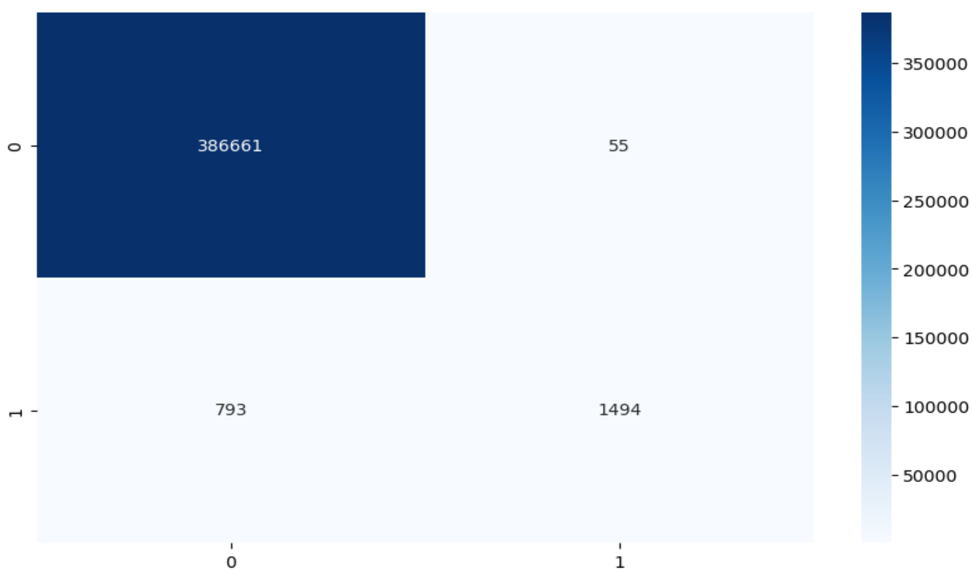
```
In [95]: algorithms = [RandomForestClassifier(), DecisionTreeClassifier(), LogisticRegression(), MultinomialNB(), discriminant_analysis.L

         algo = []
         for algorithm in algorithms:
             print(type(algorithm).__name__)

             f_score, MCC, AUC = confusion(X_train, y_train, X_valid, y_valid,model= algorithm)
             algo.append([type(algorithm).__name__, f_score, MCC, AUC])
         print(algo)
```

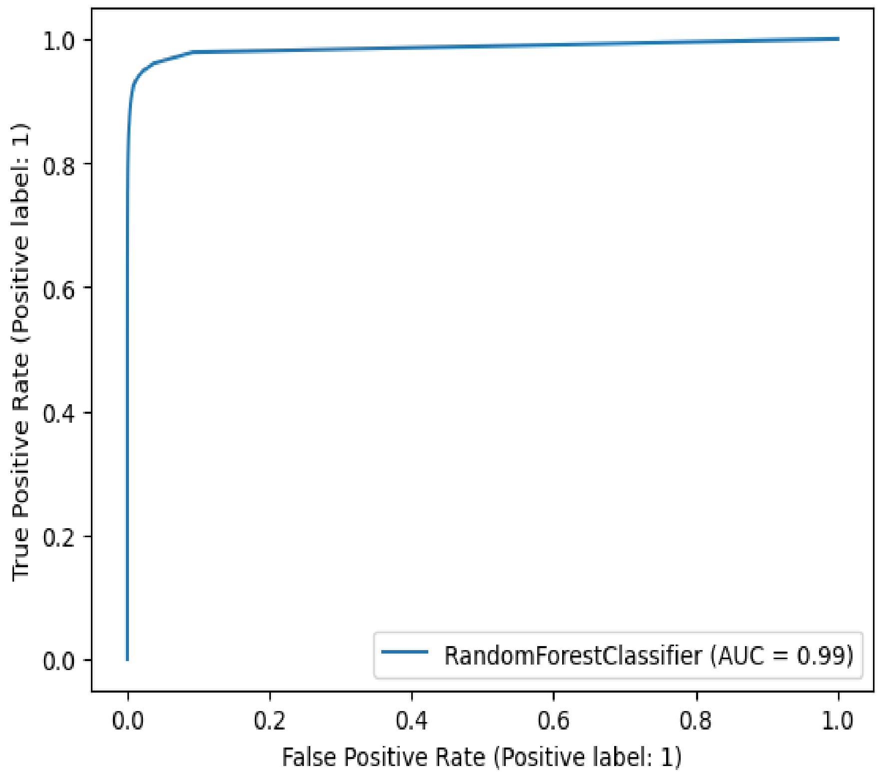
V. PERFORMANCE EVALUATION

We used the mentioned algorithms on the dataset and assessed their performance using the validation set's F1-score, Precision, Recall, Precision Under Curve Values, and Confusion Matrix

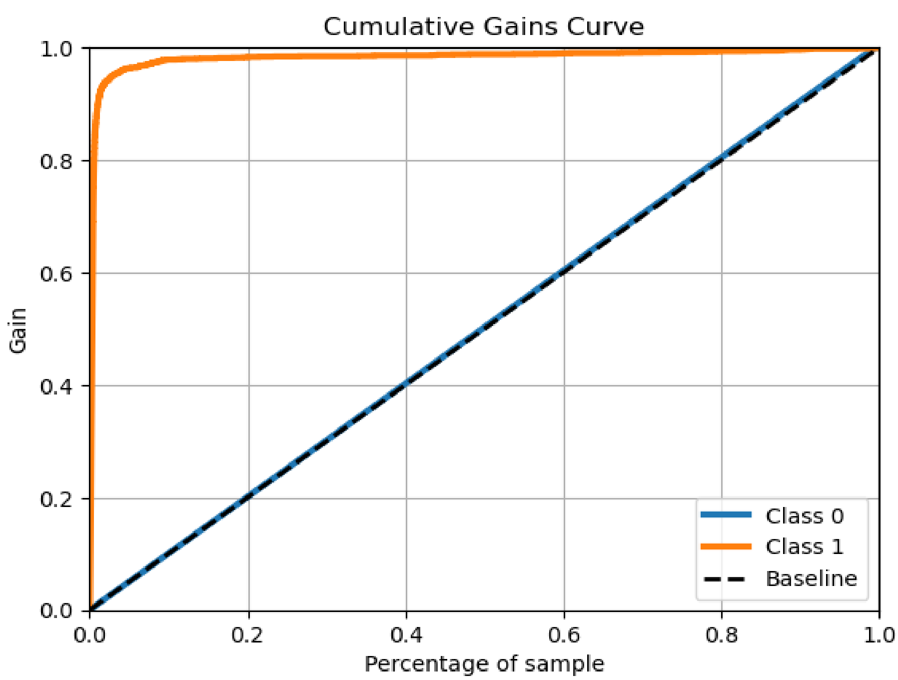
1) Random Forest Classifier

```
RandomForestClassifier
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    386716
           1       0.96      0.65      0.78      2287

    accuracy                           1.00    389003
   macro avg       0.98      0.83      0.89    389003
weighted avg       1.00      1.00      1.00    389003


Sensitivity:  0.6532575426322693
Specificity:  0.9998577767664125
```





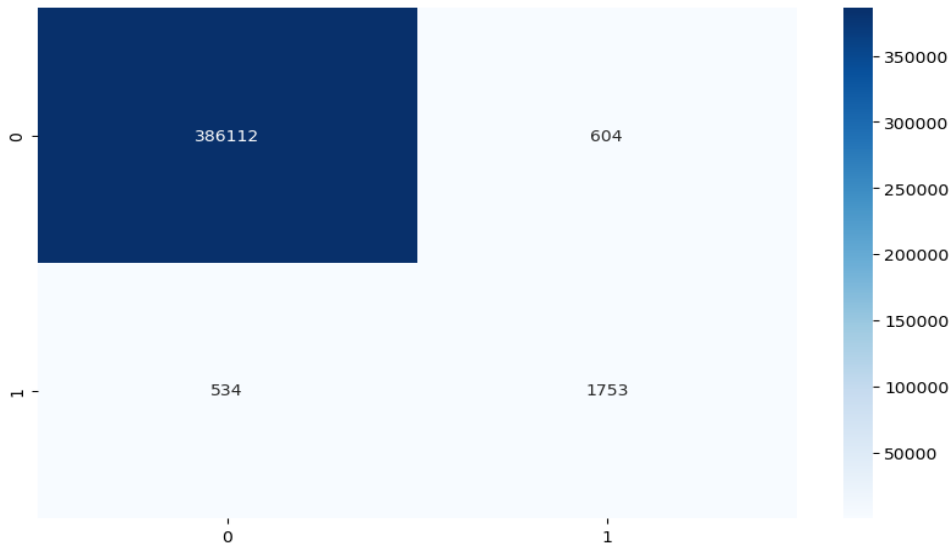
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.99)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

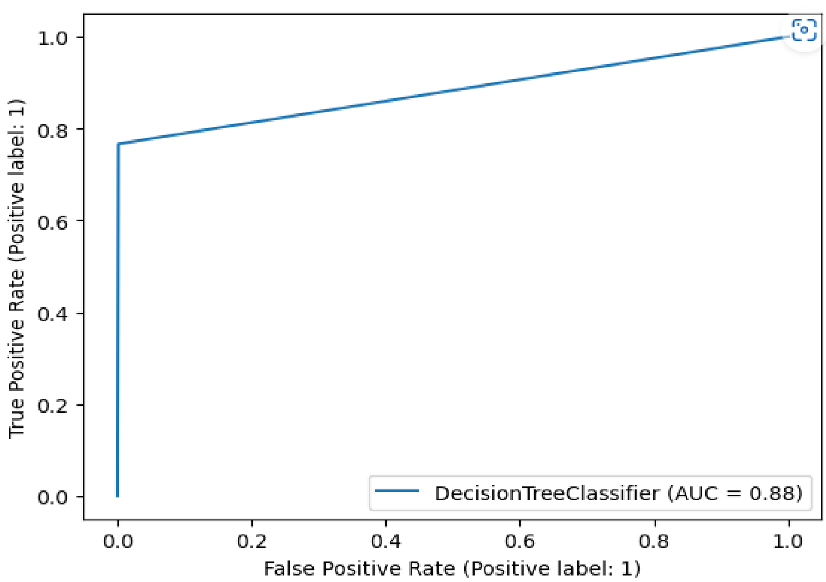


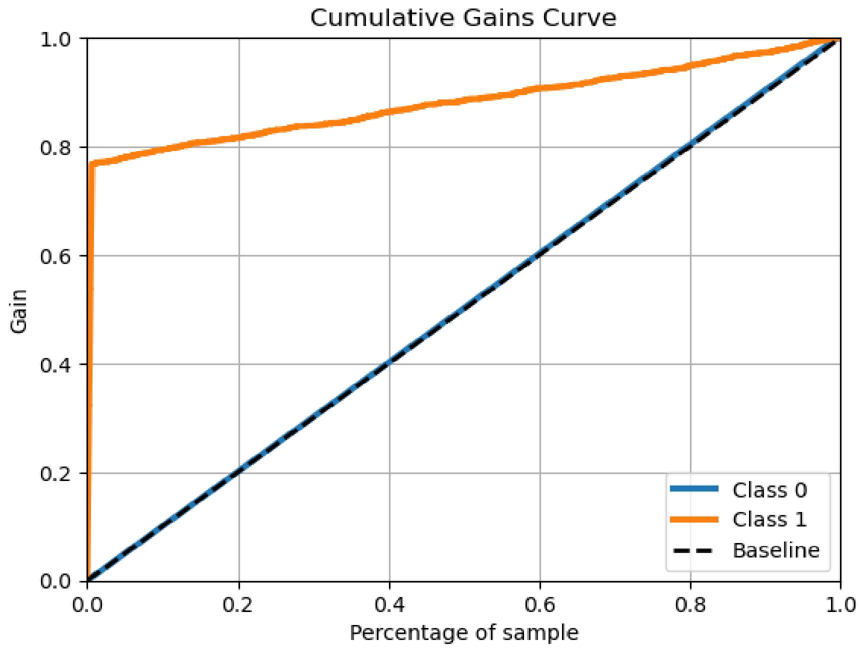
Cumulative Gains Curve
1.0
0.8
0.6
0.4
0.2
0.0
Gain
Class 0
Class 1
Baseline
0.0
0.2
0.4
0.6
0.8
1.0
Percentage of sample

2) Decision Tree Classifier

```
DecisionTreeClassifier
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    386716
           1       0.74      0.77      0.75      2287

    accuracy                           1.00    389003
   macro avg       0.87      0.88      0.88    389003
weighted avg       1.00      1.00      1.00    389003


Sensitivity:  0.7665063401836467
Specificity:  0.9984381303075125
```

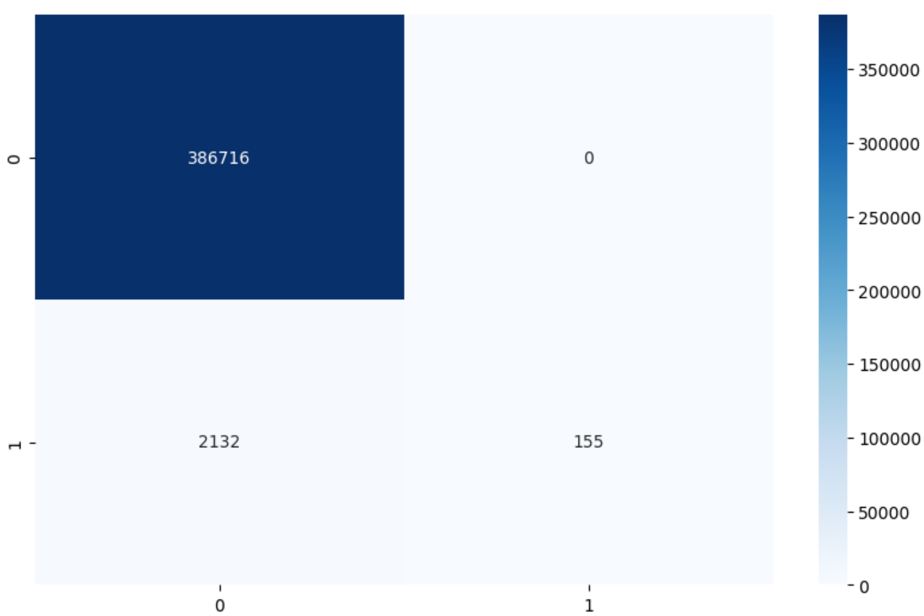


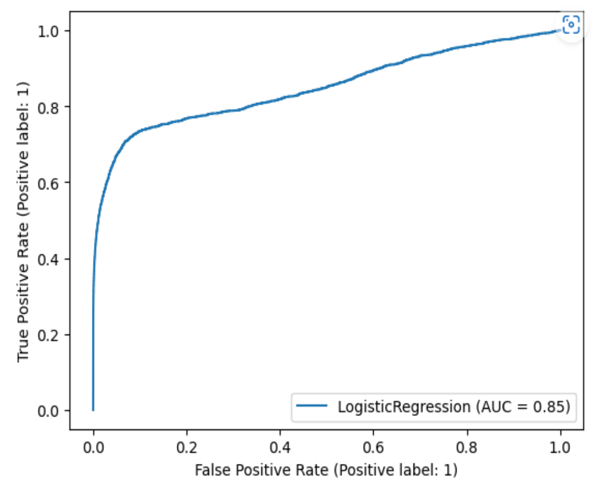


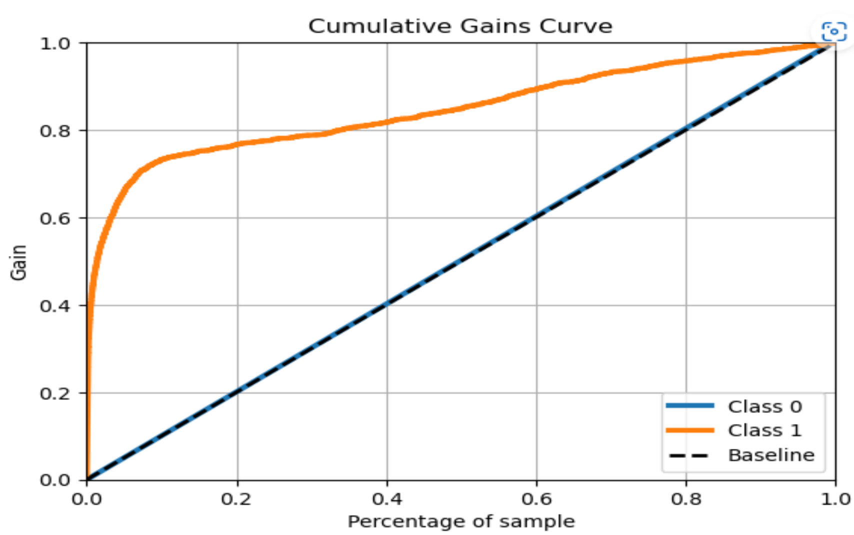


3) Logistic Regression

```
LogisticRegression
              precision    recall  f1-score   support

           0       0.99      1.00      1.00    386716
           1       1.00      0.07      0.13      2287

    accuracy                           0.99    389003
   macro avg       1.00      0.53      0.56    389003
weighted avg       0.99      0.99      0.99    389003


Sensitivity:  0.06777437691298645
Specificity:  1.0
```

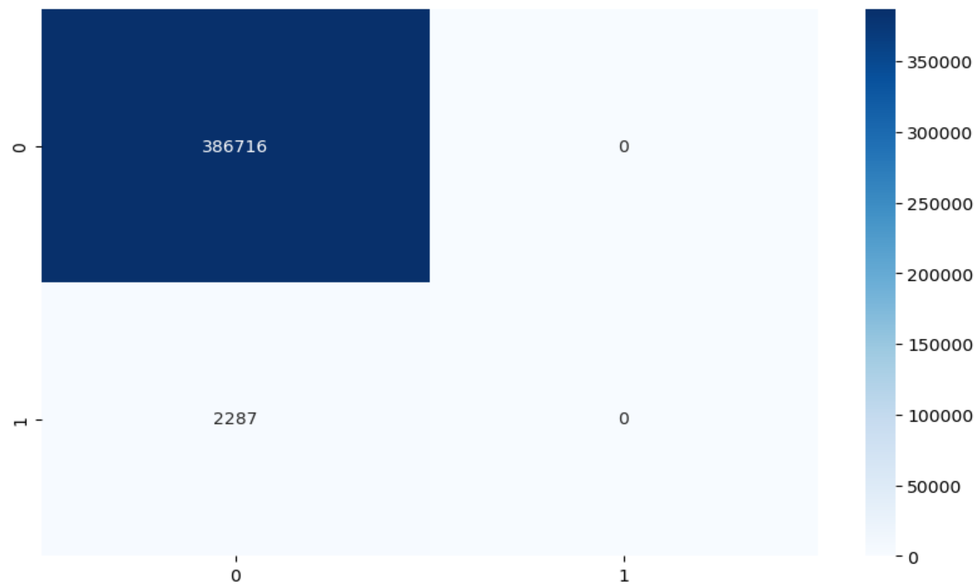


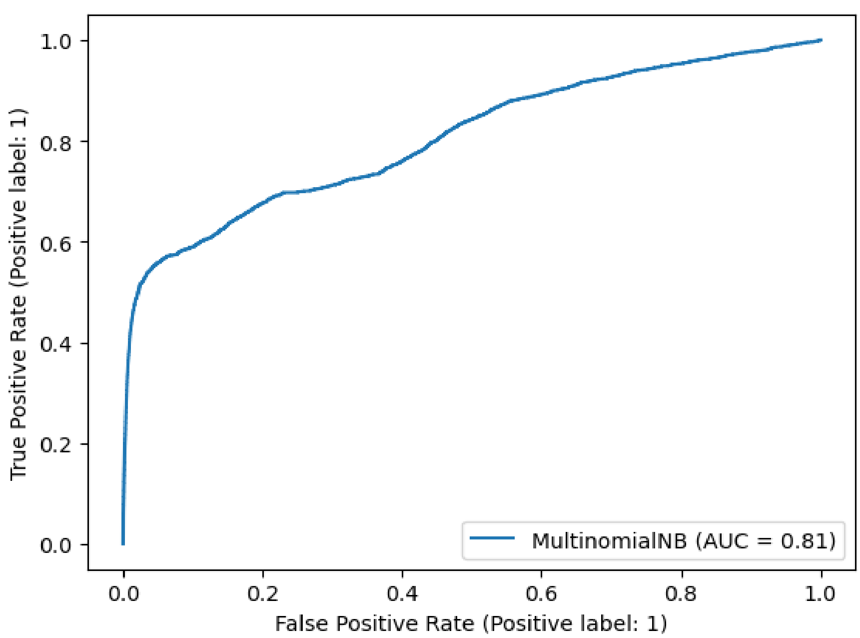

1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)
LogisticRegression (AUC = 0.85)

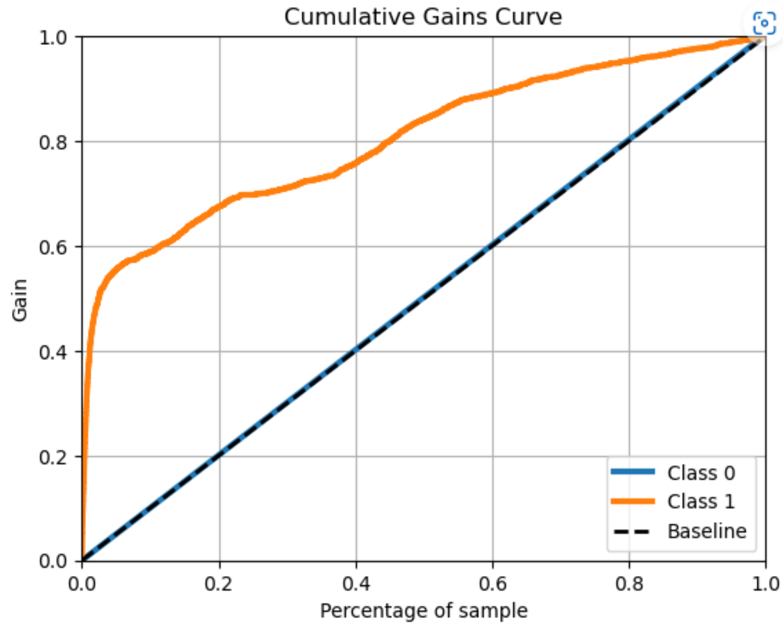

Cumulative Gains Curve
1.0
0.8
0.6
0.4
0.2
0.0
Gain
0.0
0.2
0.4
0.6
0.8
1.0
Percentage of sample
Class 0
Class 1
Baseline

4) Multinomial Naïve Baeye’s

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00    386716
           1       0.00      0.00      0.00      2287

    accuracy                           0.99    389003
   macro avg       0.50      0.50      0.50    389003
weighted avg       0.99      0.99      0.99    389003


Sensitivity:  0.0
Specificity:  1.0
```

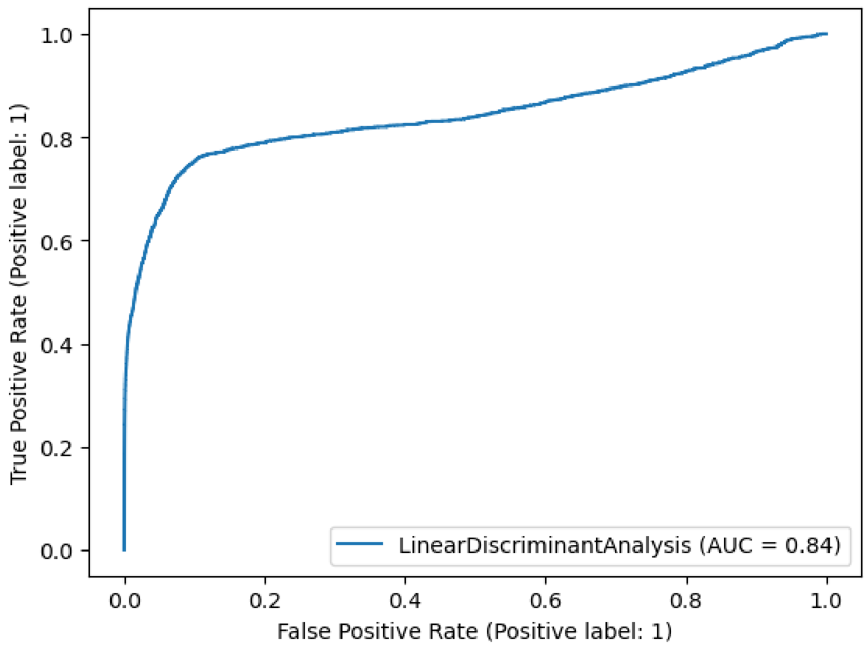




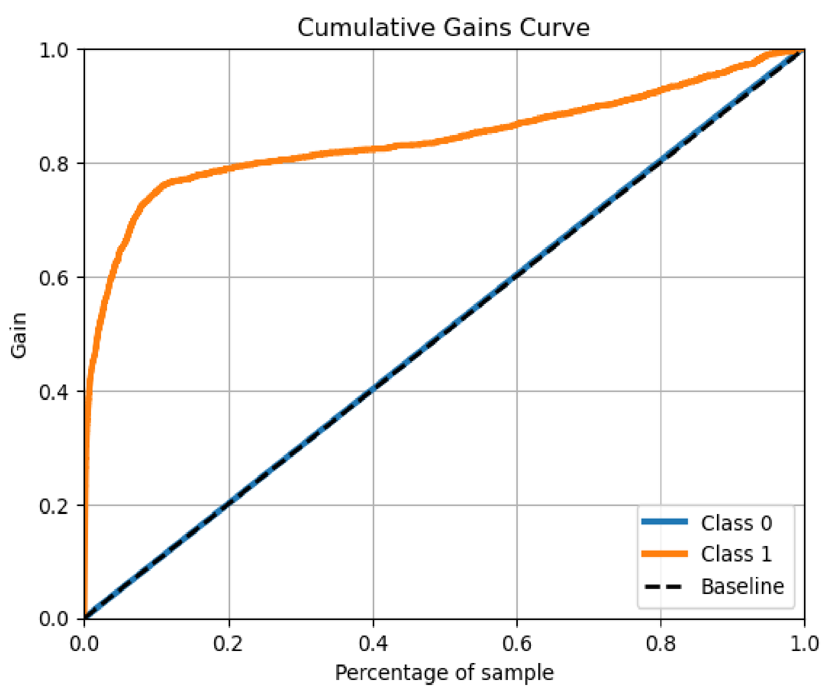


5) Linear Discriminant Analysis

```
LinearDiscriminantAnalysis
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    386716
           1       0.85      0.15      0.26      2287

    accuracy                           0.99    389003
   macro avg       0.92      0.58      0.63    389003
weighted avg       0.99      0.99      0.99    389003


Sensitivity:  0.15085264538696982
Specificity:  0.9998422615045667
```



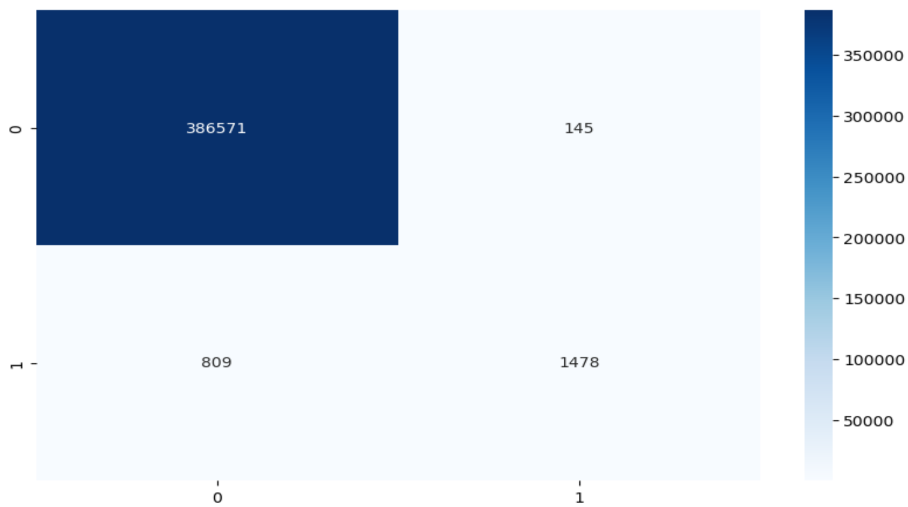
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
LinearDiscriminantAnalysis (AUC = 0.84)
0.0
0.2
0.4
0.6
0.8
1.0

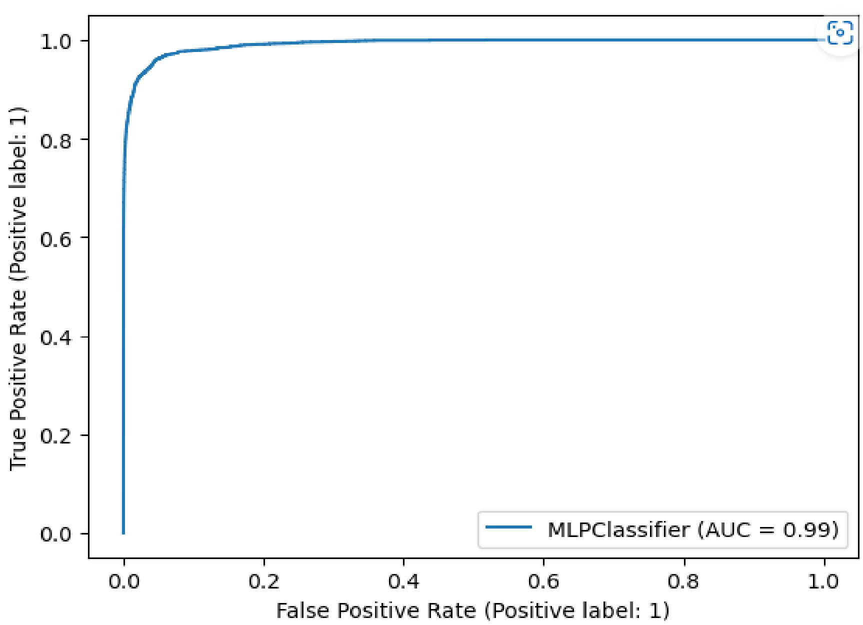


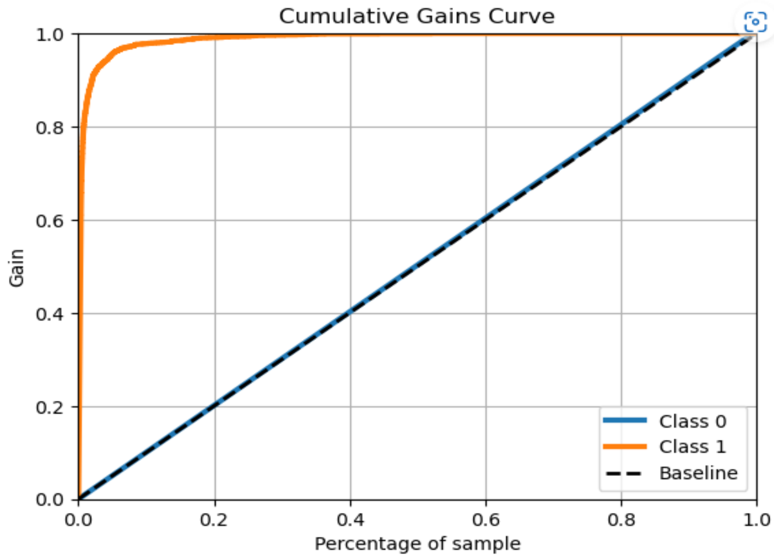
Cumulative Gains Curve
Gain
Percentage of sample
Class 0
Class 1
Baseline
0.0
0.2
0.4
0.6
0.8
1.0

6) MLP Classifier

```
MLPClassifier
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    386716
           1       0.91      0.65      0.76      2287

    accuracy                           1.00    389003
   macro avg       0.95      0.82      0.88    389003
weighted avg       1.00      1.00      1.00    389003


Sensitivity:  0.6462614779186707
Specificity:  0.9996250478387241
```

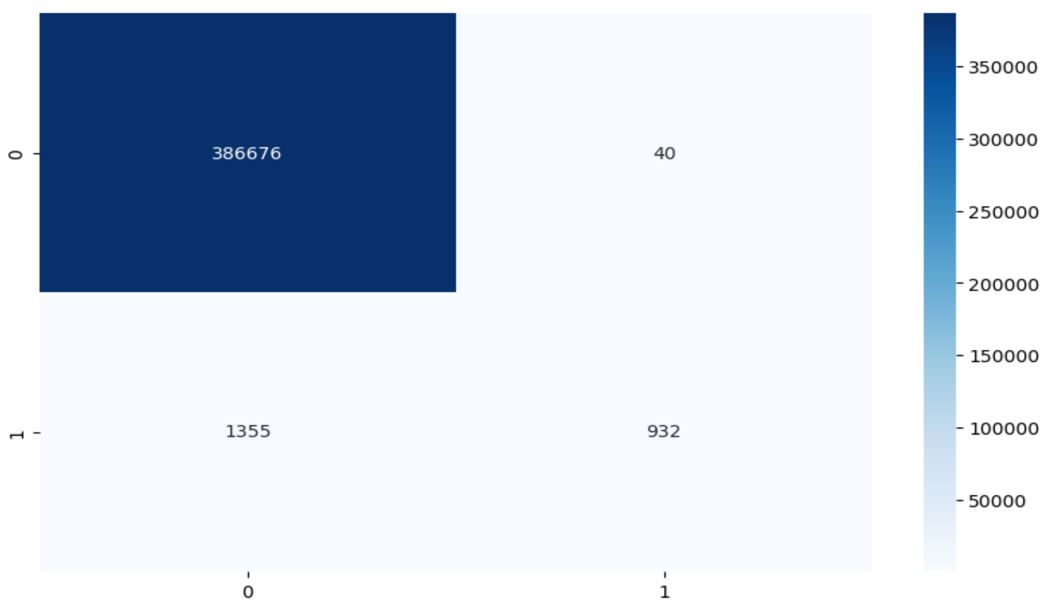


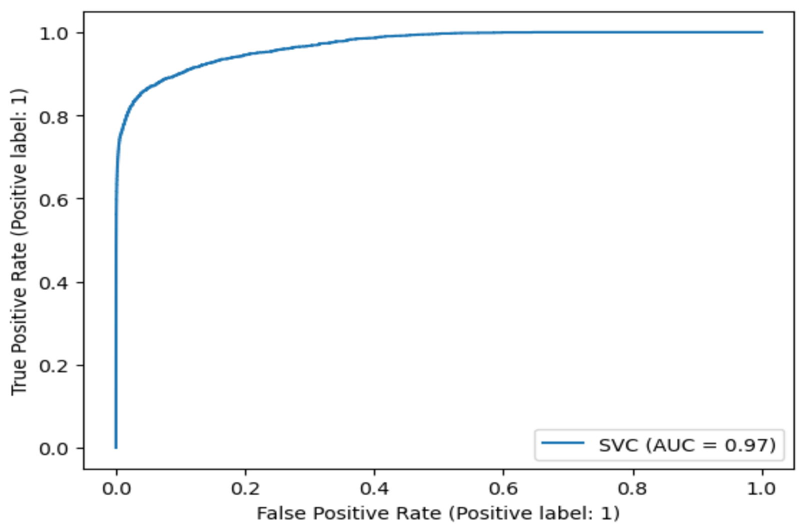




7) SVC

```
SVC
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    386716
           1       0.96      0.41      0.57      2287

    accuracy                           1.00    389003
   macro avg       0.98      0.70      0.79    389003
weighted avg       1.00      1.00      1.00    389003


Sensitivity:  0.40752076956711847
Specificity:  0.9998965649210273
predict_proba is not available when probability=False
```





VI. Discussion and Recommendation

According to the presented assessment measures, it appears that the Random Forest Classifier model outperforms the other models. An AUC score of 0.98 for the model shows that it has a strong ability to differentiate between positive and negative classifications. The model has the highest scores for sensitivity and specificity, properly identifying the majority of positive samples with a low proportion of false positives. The model offers decent prediction accuracy for the positive class, as evidenced by the precision, recall, and F1-score values for the positive class being quite high when compared to the other models. Overall, it appears that the Random Forest Classifier model balances high accuracy, precision, recall, and AUC, which are crucial metrics for a binary classification problem, the best. It's crucial to remember that selecting the optimal model ultimately depends on the particular requirements and limitations of the current challenge.

VII. Summary

Machine learning algorithms and rule-based systems are only two of the methods that fraud detection systems utilize to spot possibly fraudulent transactions. Real-time monitoring, transaction pattern analysis, and anomaly identification are some typical characteristics of fraud detection systems. In order to increase the precision and effectiveness of fraud detection systems, cutting-edge methodologies like deep learning and artificial intelligence are also being deployed. A multi-layered approach that incorporates both technology and human experience is necessary for effective fraud detection. To find new and emerging fraud types, transaction data must be continuously monitored and analyzed. Implementing a strong fraud detection approach can greatly lower the risk of financial losses due to credit card fraud, even though no fraud detection system is 100% reliable.