MGT3013 – Risk and Fraud Analytics

J Component – Project Report

TRANSACTION FRAUD DETECTION

By

19MIA1028	Nithya Sharma
19MIA1053	Yuvashree R
19MIA1063	Aishwarya S
19MIA1073	Keerthana Madhavan
19MIA1083	K Niharika Samyuktha
19MIA1084	Podalakuru Sahithya

M.Tech Intg CSE with Business Analytics

Submitted to

Dr. Bhuvanesh C,

Assistant Professor,

VITBS, VIT, Chennai

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

November 2022

ABSTRACT

Financial fraud is a serious problem that is only getting worse and has far-reaching effects. Data mining has been crucial in identifying fraudulent internet transactions. To enhance client experience and reduce financial loss, it is essential for financial companies to aggressively detect transactional hazards. In this work, we compare various machine learning algorithms for predicting the authenticity of financial transactions with efficiency and effectiveness. The algorithms used in this study were: SVM, KNN, Logistic Regression, and XGBoost. The dataset was collected from Kaggle. The Balanced Accuracy of SVM is 50%, Precession is 0%, Recall is 0%, the F1-score is 0%, and Kappa is 0%. The Balanced Accuracy of KNN is 54.3%, Precession is 83.3%, Recall is 8.6%, the F1-score is 15.6% and Kappa is 15.6%. The Balanced Accuracy of Logistic Regression is 55.2%, Precession is 100%, Recall is 10.3%, the F1-score is 0%, and Kappa is 18.7%. The Balanced Accuracy of XGBoost is 77.6%, Precession is 94.1%, Recall is 55.2%, the F1-score is 0%, and Kappa is 69.5%.

TABLE OF CONTENTS

1. Introduction

For a long time, fraudulent transactions and detectors have had a composite role. Fraudulent transactions are happening more frequently than ever before, principally in today's era of the Internet, and it is the cause of foremost financial losses. Detectors and fraudulent transactions have played complementary roles for a very long time. The biggest source of financial losses is fraud, which is occurring more frequently than ever before, especially in this era of the Internet. Around $28 billion was lost to the economy as a result of transaction fraud in 2019, $30 billion in 2020, and more than $32 billion in 2021. The amount of fraudulent transactions worldwide is anticipated to increase yearly, reaching $34 billion in 2022. In order to identify and screen financial transactions, banks and financial service providers might need an automatic fraud detection tool. Systems for detecting fraud are designed to separate out unusual action patterns from a vast number of transactional records, and then utilize those patterns to find or follow fraud. Machine learning has been revealed to be very rewarding at detecting and classification of fraud transactions. In another way, a great number of transaction reports may be used to train and validate fraud classifiers. Although supervised learning has been quite effective at identifying fraudulent transactions, transactional fraud analysis technology will continue to advance.

2. Literature Survey

Sl no	Title	Author / Journal name / Year	Technique	Result
1	Online Transaction Fraud Detection System Based on Machine Learning	Bocheng Liu, Xiang Chen1 and Kaizhi Yu Journal of Physics: Conference (ICCTPE 2021)	Based on Fully Connected Neural Network & XGBoost model, which can automatically analyze the transaction data uploaded and return the fraud detection results to users.	AUC values can achieve 0.912 and 0.969 respectively.
2	Detecting Credit Card Fraud Using Selected Machine Learning Algorithms	Puh, Maja & Brkic, Ljiljana. (2019). MIPRO	In this paper, three major ML algorithms are used: Random Forest (RF), Support Vector Machine (SVM) and Logistic Regression (LR).	The results suggested that SVM shows the poorest performance in both static and incremental setup and the difference between RF and LR is slight. LR shows marginally better results in incremental setup of accuracy 84%.

3	Online Transaction Fraud Detection System Using Machine Learning & E-Commerce	International Research Journal of Engineering and Technology (IRJET) Volume: 09 Issue: 04 | Apr 2022 Shayan Wangde, Raj Kheratkar, Zoheb Waghu, Prof. Suhas Lawand	We attempted to detect fraud using bank payment data (Behavior and Location Analysis) The SMOTE oversampling technique to produce additional minority class cases	Random forest: Accuracy of 0.98 KNN: Accuracy of 0.99
4	Credit Card Fraud Detection: A Hybrid Approach Using Fuzzy Clustering & Neural Network	T. K. Behera and S. Panigrahi, 2015 Second International Conference on Advances in Computing and Communication Engineering.	In this paper, a fuzzy c-means clustering algorithm is applied to find out the normal usage patterns of credit card users based on their past activity. Neural network based learning mechanism is applied to determine whether it was actually a fraudulent activity or an occasional deviation by a genuine user.	Extensive experimentation with stochastic models shows that the combined use of clustering techniques along with learning helps in detecting fraudulent activities effectively while minimizing the generation of false alarms.

5	Online Transaction Fraud Detection using Python & Backlogging on ECommerce	Prof. Radhika Mundhada, Oz Ibrahim, Maneri Sohel3,Sunny Prajapati Momin Azhar5, IJSRD - International Journal for Scientific Research & Development| Vol. 9, Issue 3, 2021 | ISSN (online): 2321-0613	An extensive survey disclosed the techniques used in this domain include data mining applications, automated fraud detection, Naive Bayes System, Hidden Markov Model, Artificial Immune System	

3. Dataset

The dataset is taken from Kaggle.

https://www.kaggle.com/datasets/ealaxi/paysim1

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
283	CASH_IN	210329.84	C1159819632	3778062.79	3988392.64	C1218876138	1519266.6	1308936.76	0	0
132	CASH_OUT	215489.19	C1372369468	21518	0	C467105520	6345756.55	6794954.89	0	0
355	DEBIT	4431.05	C1059822709	20674	16242.95	C76588246	80876.56	85307.61	0	0
135	CASH_OUT	214026.2	C1464960643	46909.73	0	C1059379810	13467450.36	13681476.56	0	0
381	CASH_OUT	8858.45	C831134427	0	0	C579876929	1667180.58	1676039.03	0	0
208	CASH_IN	256440.86	C1001269496	554	256994.86	C1503528288	0	0	0	0
347	CASH_OUT	120989.98	C803141068	0	0	C1951883397	6963396.37	7084386.35	0	0
183	CASH_OUT	62655.01	C309960888	18997	0	C1715810305	130706.17	193361.18	0	0
184	CASH_OUT	256745.11	C1447987365	0	0	C1629212528	1475890.41	1732635.52	0	0
12	PAYMENT	13693.22	C243045039	9040	0	M55115072	0	0	0	0
15	PAYMENT	8688.19	C506470255	0	0	M1821067206	0	0	0	0
186	PAYMENT	1028.37	C410510449	0	0	M1322208123	0	0	0	0
321	CASH_IN	147708.56	C2624555	348	148056.56	C832913110	0	0	0	0
691	CASH_IN	41882.88	C2047867326	71044	112926.88	C1089286229	0	0	0	0
239	CASH_OUT	749.39	C1347941621	0	0	C787242927	1585956.31	1586705.7	0	0
163	CASH_IN	126511.11	C863490494	147480.22	273991.33	C57994636	239264.38	112753.27	0	0
350	CASH_OUT	230581.53	C979441035	0	0	C1185333233	27634106.04	27864687.56	0	0
137	CASH_OUT	317575.58	C1490538273	20047.77	0	C1396438141	1192790	1759040.77	0	0
188	CASH_IN	87633.89	C2086756631	2812144.57	2899778.46	C1058045286	2507636.17	2420002.28	0	0
159	CASH_OUT	254252.74	C1022177576	32778	0	C1164758175	0	254252.74	0	0
283	PAYMENT	6169.23	C1110969574	226	0	M1793441748	0	0	0	0
184	PAYMENT	41582.12	C211097674	20042	0	M1684313434	0	0	0	0

step - maps a unit of time in the real world. In this case 1 step is 1 hour of time. Total steps 744 (30 days simulation).

type - CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER.

amount - amount of the transaction in local currency.

nameOrig - customer who started the transaction

oldbalanceOrg - initial balance before the transaction

newbalanceOrig - new balance after the transaction

nameDest - customer who is the recipient of the transaction

oldbalanceDest - initial balance recipient before the transaction. Note that there is not information for customers that start with M (Merchants).

newbalanceDest - new balance recipient after the transaction. Note that there is not information for customers that start with M (Merchants).

isFraud - This is the transactions made by the fraudulent agents inside the simulation. In this specific dataset the fraudulent behavior of the agents aims to profit by taking control or

customers accounts and try to empty the funds by transferring to another account and then cashing out of the system.

isFlaggedFraud - The business model aims to control massive transfers from one account to another and flags illegal attempts. An illegal attempt in this dataset is an attempt to transfer more than 200.000 in a single transaction.

Data types and structure

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284483 entries, 0 to 284482
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   step              284483 non-null  int64
 1   type              284483 non-null  object
 2   amount            284483 non-null  float64
 3   name_orig         284483 non-null  object
 4   oldbalance_org    284483 non-null  float64
 5   newbalance_orig   284483 non-null  float64
 6   name_dest         284483 non-null  object
 7   oldbalance_dest   284483 non-null  float64
 8   newbalance_dest   284483 non-null  float64
 9   is_fraud          284483 non-null  int64
 10  is_flagged_fraud  284483 non-null  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 23.9+ MB
```

4. Proposed Methodology

- **Data Description:** Data collection and analysis will be our initial phase. Threats or removals will be made against the lacking values. Calculations for descriptive statistics such as kurtosis, skewness, media, fashion, median, and standard deviation will be performed.
- **Feature Engineering:** To help with the formulation of the hypothesis and the development of additional features, a mind map will be made in this part. These presumptions will be useful for exploratory data analysis and could raise model performance.
- **Data Filtering:** To eliminate columns or rows that are not relevant to the company, employ data filtering. For instance, rows with age do not include human age or columns with customer ID or hash codes.
- **Exploratory Data Analysis:** To aid in comprehending the database, the exploratory data analysis section includes univariate analysis, bivariate analysis, and multivariate analysis. The bivariate analysis will test the hypothesis that was developed in step 2.
- **Data Preparation:** The data will be ready for machine learning modeling in this fifth section. They will be altered accordingly to enhance the machine learning model's capacity to learn; they may be encoded, oversampled, subsampled, or rescaled.
- **Feature Selection:** Following this section's data preparation, algorithms like Boruta will decide which columns to employ for the machine learning model's training. This lessens the database's dimensionality and lowers the likelihood of overfitting.
- **Machine Learning Modeling:** The goal of step 7 is to teach the machine learning algorithms how to forecast the data. The model is trained, validated, and used for cross-validation for validation in order to determine the model's capability for learning.
- **Hyper parameter Fine Tuning:** After choosing the appropriate model to use for the project, it's crucial to fine-tune the parameters to raise the model's results. The same model performance techniques that were utilized in step 7 are employed.
- **Conclusions:** The generation capacity model is evaluated at this point using hypothetical data. In order to demonstrate the model's application in the business setting, various questions about business are also answered.

5. Algorithms / Techniques description:

Support Vector Machine

A supervised learning approach called Support Vector Machine, or SVM, is used to solve Classification and Regression issues. It is primarily employed in Machine Learning Classification issues. SVM categorizes data points even when they are not otherwise linearly separable by mapping the data to a high-dimensional feature space. Once a separator between the categories is identified, the data are converted to enable the hyperplane representation of the separator.

K Nearest Neighbors

A machine learning algorithm called K-Nearest Neighbor is based on the technique of supervised learning. The K-NN algorithm makes the assumption that the new case and the existing cases are comparable, and it places the new instance in the category that is most like the existing categories. The K-nearest neighbors (KNN) algorithm predicts the values of new data points based on their "feature similarity," which further indicates that the value of the new data point will depend on how closely it resembles the values of the points in the training set.

Logistic Regression

Logistic regression is a supervised machine learning model. Logistic regression predicts the likelihood that an event will occur; since the outcome is a probability, the dependent variable is restricted to the range of 0 and 1, making it a discriminative model that seeks to distinguish between classes (or categories). Logistic regression is utilised when your y variable can only take two values and it is more effective to divide the data into two distinct classes if it is linearly separable.

XGBoost

XGBoost is a popular and efficient open-source implementation of the gradient-boosted trees algorithm. Gradient boosting is a supervised learning process that combines the predictions of a number of weaker, simpler models to attempt to properly predict a target

variable. Extreme Gradient Boosting (XGBoost) is a distributed, scalable gradient-boosted decision tree (GBDT) machine learning framework. The top machine learning library for regression, classification, and ranking issues, it offers parallel tree boosting.

6. Results

Logistic Regression ¶

In [54]:
```
lg = LogisticRegression()
lg.fit(X_train_cs, y_train)

y_pred = lg.predict(X_valid_cs)
```

In [55]:
```
lg_results = ml_scores('Logistic Regression', y_valid, y_pred)
lg_results
```

Out[55]:

	Balanced Accuracy	Precision	Recall	F1	Kappa
Logistic Regression	0.552	1.0	0.103	0.188	0.187

In [56]:
```
print(classification_report(y_valid, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     45460
           1       1.00      0.10      0.19        58

    accuracy                           1.00     45518
   macro avg       1.00      0.55      0.59     45518
weighted avg       1.00      1.00      1.00     45518
```

In [57]:
```
lg_cv = ml_cv_results('Logistic Regression',
                      LogisticRegression(),
                      X_temp_cs, y_temp)
lg_cv
```

```
Fold K=1
Fold K=2
Fold K=3
Fold K=4
Fold K=5
```

Out[57]:

	Balanced Accuracy	Precision	Recall	F1	Kappa
Logistic Regression	0.545 +/- 0.021	1.0 +/- 0.0	0.089 +/- 0.043	0.161 +/- 0.074	0.161 +/- 0.074

KNN

In [58]:
```
knn = KNeighborsClassifier()
knn.fit(X_train_cs, y_train)

y_pred = knn.predict(X_valid_cs)
```

In [59]:
```
knn_results = ml_scores('K Nearest Neighbors', y_valid, y_pred)
knn_results
```

Out[59]:

	Balanced Accuracy	Precision	Recall	F1	Kappa
K Nearest Neighbors	0.543	0.833	0.086	0.156	0.156

In [60]:
```
print(classification_report(y_valid, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     45460
           1       0.83      0.09      0.16        58

    accuracy                           1.00     45518
   macro avg       0.92      0.54      0.58     45518
weighted avg       1.00      1.00      1.00     45518
```

In [61]:
```
knn_cv = ml_cv_results('K Nearest Neighbors', KNeighborsClassifier(),
                       X_temp_cs, y_temp)
knn_cv
```

```
Fold K=1
Fold K=2
Fold K=3
Fold K=4
Fold K=5
```

Out[61]:

	Balanced Accuracy	Precision	Recall	F1	Kappa
K Nearest Neighbors	0.675 +/- 0.02	0.971 +/- 0.024	0.351 +/- 0.04	0.514 +/- 0.045	0.514 +/- 0.045

SVM

In [62]:
```
svm = SVC()
svm.fit(X_train_cs, y_train)

y_pred = svm.predict(X_valid_cs)
```

In [63]:
```
svm_results = ml_scores('SVM', y_valid, y_pred)
svm_results
```

Out[63]:

	Balanced Accuracy	Precision	Recall	F1	Kappa
SVM	0.5	0.0	0.0	0.0	0.0

In [64]:
```
print(classification_report(y_valid, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     45460
           1       0.00      0.00      0.00        58

    accuracy                           1.00     45518
   macro avg       0.50      0.50      0.50     45518
weighted avg       1.00      1.00      1.00     45518
```

In [65]:
```
svm_cv = ml_cv_results('SVM', SVC(), X_temp_cs, y_temp)
svm_cv
```

```
Fold K=1
Fold K=2
Fold K=3
Fold K=4
Fold K=5
```

Out[65]:

	Balanced Accuracy	Precision	Recall	F1	Kappa
SVM	0.576 +/- 0.019	1.0 +/- 0.0	0.151 +/- 0.037	0.261 +/- 0.054	0.261 +/- 0.054

XGBoost

In [67]:
```
xgb = XGBClassifier()
xgb.fit(X_train_cs, y_train)

y_pred = xgb.predict(X_valid_cs)
```

In [68]:
```
xgb_results = ml_scores('XGBoost', y_valid, y_pred)
xgb_results
```

Out[68]:

	Balanced Accuracy	Precision	Recall	F1	Kappa
XGBoost	0.776	0.941	0.552	0.696	0.695

In [69]:
```
print(classification_report(y_valid, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     45460
           1       0.94      0.55      0.70        58

    accuracy                           1.00     45518
   macro avg       0.97      0.78      0.85     45518
weighted avg       1.00      1.00      1.00     45518
```

In [70]:
```
xgb_cv = ml_cv_results('XGBoost', XGBClassifier(),
                       X_temp_cs, y_temp)
xgb_cv
```

```
Fold K=1
Fold K=2
Fold K=3
Fold K=4
Fold K=5
```

Out[70]:

	Balanced Accuracy	Precision	Recall	F1	Kappa
XGBoost	0.818 +/- 0.031	0.959 +/- 0.018	0.635 +/- 0.062	0.762 +/- 0.045	0.762 +/- 0.045

Comparing Performances ¶

In [71]:
```
modeling_performance = pd.concat([dummy_results, lg_results, knn_results,
                                  xgb_results,svm_results])
modeling_performance.sort_values(by="F1", ascending=True)
```

Out[71]:

	Balanced Accuracy	Precision	Recall	F1	Kappa
dummy	0.500	0.000	0.000	0.000	0.000
SVM	0.500	0.000	0.000	0.000	0.000
K Nearest Neighbors	0.543	0.833	0.086	0.156	0.156
Logistic Regression	0.552	1.000	0.103	0.188	0.187
XGBoost	0.776	0.941	0.552	0.696	0.695

7. Findings and Conclusion

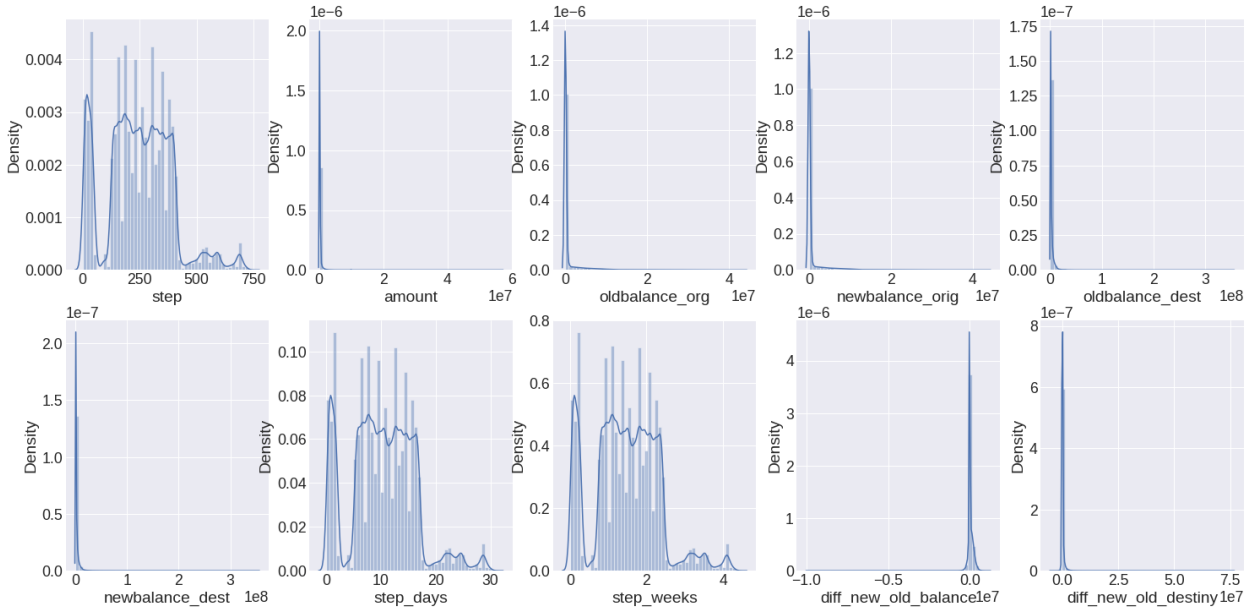
Good prediction results can be achieved with imbalanced datasets as well as with balanced ones. Logistic Regression, SVM, KNN & XGBClassifier gave us the best results being able to detect more than 77.6% of fraud transactions with a precision of 94.1% and at the same time not classifying some of the non-fraud transactions as fraud. There is no perfect model, and precision and recall will always have to be traded off. The choice of strategy for each unique circumstance is up to the business and its goals.

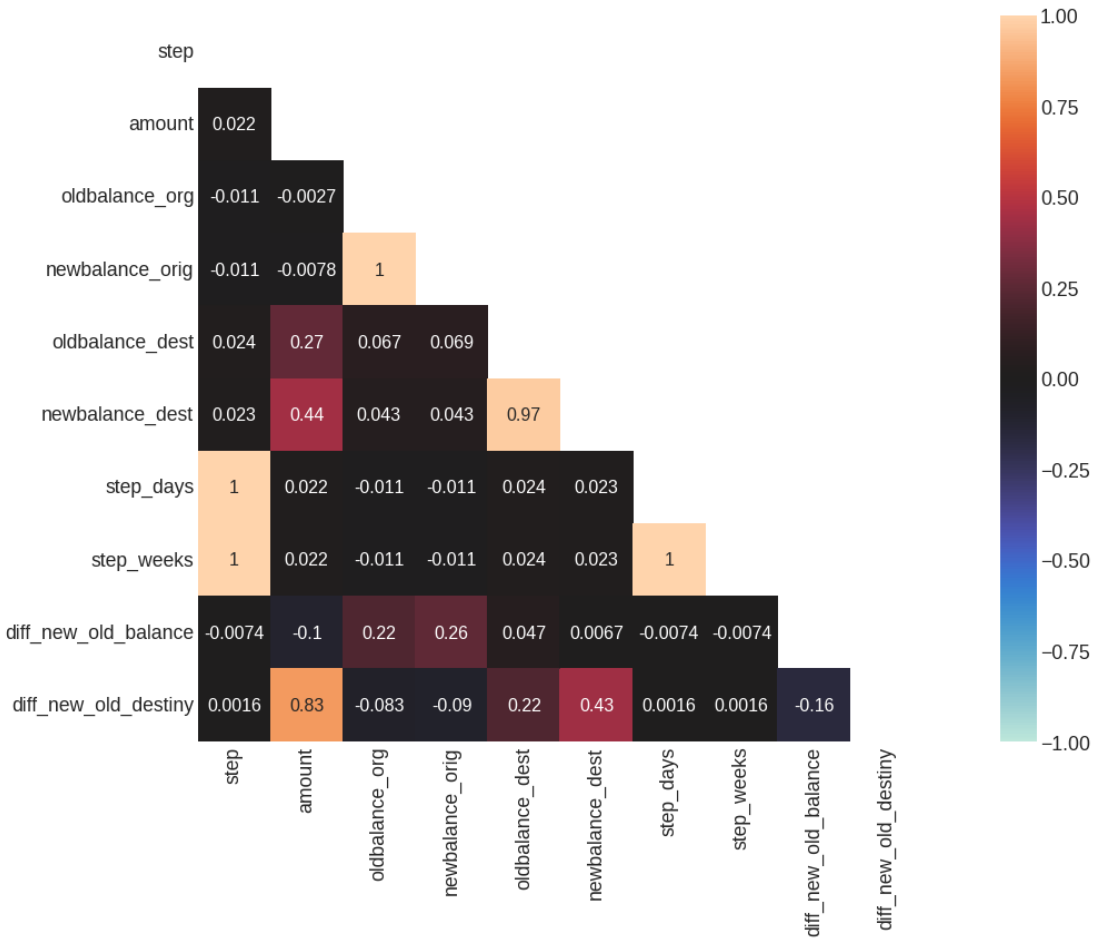
	Balanced Accuracy	Precision	Recall	F1	Kappa
dummy	0.500	0.000	0.000	0.000	0.000
SVM	0.500	0.000	0.000	0.000	0.000
K Nearest Neighbors	0.543	0.833	0.086	0.156	0.156
Logistic Regression	0.552	1.000	0.103	0.188	0.187
XGBoost	0.776	0.941	0.552	0.696	0.695

On further Hyper fine-tuning, XGBoost classifier we got an Accuracy of 81.6% and a Precision of 94.7%

	Balanced Accuracy	Precision	Recall	F1	Kappa
XGBoost GS	0.816 +/- 0.037	0.947 +/- 0.026	0.633 +/- 0.073	0.757 +/- 0.059	0.757 +/- 0.059

	Balanced Accuracy	Precision	Recall	F1	Kappa
unseen	0.808	0.978	0.616	0.756	0.756





REFERENCES

1. H., Mehbodniya, A., Alam, I., Pande, S., Neware, R., Rane, K. P., Shabaz, M., & Madhavan, M. V. (2021, September 11). *Financial Fraud Detection in Healthcare Using Machine Learning and Deep Learning Techniques*. Financial Fraud Detection in Healthcare Using Machine Learning and Deep Learning Techniques. Retrieved November 4, 2022, from https://www.hindawi.com/journals/scn/2021/9293877/
2. J. O. Awoyemi, A. O. Adetunmbi and S. A. Oluwadare, "Credit card fraud detection using machine learning techniques: A comparative analysis," *2017 International Conference on Computing Networking and Informatics (ICCNI)*, 2017, pp. 1-9, doi: 10.1109/ICCNI.2017.8123782. https://ieeexplore.ieee.org/abstract/document/8123782
3. https://seon.io/resources/fraud-detection-and-prevention/
4. Nath Dornadula, & S . (2020, February 27). *Credit Card Fraud Detection using Machine Learning Algorithms*. Credit Card Fraud Detection Using Machine Learning Algorithms -

ScienceDirect. Retrieved November 4, 2022, from https://www.sciencedirect.com/science/article/pii/S187705092030065X

5. https://www.heavy.ai/technical-glossary/fraud-detection-and-prevention
6. Mittal, & Tyagi. (n.d.). *Performance Evaluation of Machine Learning Algorithms for Credit Card Fraud Detection*. Performance Evaluation of Machine Learning Algorithms for Credit Card Fraud Detection | IEEE Conference Publication | IEEE Xplore. Retrieved November 4, 2022, from https://ieeexplore.ieee.org/abstract/document/8776925
7. Varun Kumar K S , Vijaya Kumar V G , Vijayshankar A , Pratibha K, 2020, Credit Card Fraud Detection using Machine Learning Algorithms, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 09, Issue 07 (July 2020), https://www.ijert.org/credit-card-fraud-detection-using-machine-learning-algorithms
8. https://binariks.com/blog/financial-fraud-detection-machine-learning/
9. Mosa M. M. Megdad, Bassem S. Abu-Nasser and Samy S. Abu-Naser,Department of Information Technology,Faculty of Engineering and Information Technology,Al-Azhar University, Gaza, Palestine, *Fraudulent Financial Transactions Detection Using Machine Learning* ,International Journal of Academic Information Systems Research (IJAISR) Vol. 6 Issue 3, March - 2022, Pages:30-39 www.ijeais.org/ijaisr