Identify fraudulent transaction by machine learning

Qixuan Zhang(qz367) & Sujie Fang(sf568)

Course: ORIE 5741

Instructor: Dr. Haiyun He

1. Abstract

This project aims to utilize machine learning techniques for the detection of fraudulent transactions in financial data. The rise of financial fraud in recent years has prompted the need for effective prevention methods. By leveraging a comprehensive dataset and employing various machine learning models, this study seeks to identify fraudulent activities and contribute to the prevention of financial losses. The proposed process involves data analysis, feature selection, correlation analysis, data visualization, data cleaning, feature engineering, and a comparative analysis of different machine learning algorithms. By identifying the most relevant features and evaluating the performance of multiple models, the project aims to construct an accurate predictive model that can effectively identify and prevent fraudulent transactions.

2. Background

Financial fraud has emerged as a significant challenge for financial institutions worldwide, leading to substantial economic losses. Fraudulent practices such as account takeovers, phishing attacks, and identity theft have wreaked havoc on individuals and organizations alike. As a data analyst, this project seeks to address the pressing issue of financial fraud prevention using machine learning. By leveraging advanced algorithms and utilizing the vast potential of machine learning models, the project aims to develop a robust solution to detect and prevent fraudulent activities.

3. Introduction

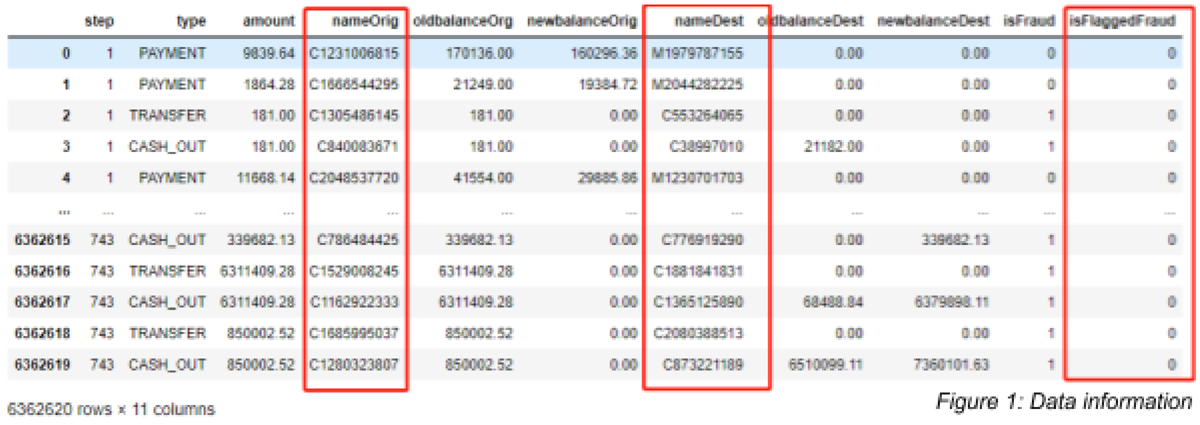
The selected dataset, the "Synthetic Financial Dataset For Fraud Detection" obtained from Kaggle, forms the foundation of this project. This dataset encompasses a vast array of transactional data, featuring 11 distinct attributes, including step, type, amount, nameOrig, oldbalanceOrg, newbalanceOrig, nameDest, oldbalanceDest, newbalanceDest, isFraud, and isFlaggedFraud. The inclusion of such a diverse range of variables will facilitate the training and testing of machine learning models to address the problem at hand.

The significance of this project lies in its potential to curb financial fraud, offering protection to financial institutions and their clientele. Machine learning algorithms present an opportunity to detect fraudulent transactions promptly, thereby minimizing financial losses and preventing further harm. By identifying patterns and markers of fraudulent behavior, financial institutions can implement proactive measures to safeguard their customers' accounts and maintain the integrity of their systems.

Through a comprehensive data analysis process, involving feature selection, correlation analysis, data visualization, data cleaning, feature engineering, and a comparative study of different machine learning models, this project seeks to identify the most pertinent features for predicting fraud and determine the most effective machine learning model for the given dataset. By leveraging these insights, the project aims to develop a highly accurate predictive model that can be employed in real-world scenarios to prevent fraudulent activities within the financial system.

3. Data clean and EDA

By reading the data initially, we obtained the following data information in *Figure 1*. After inspection, we decided that variables such as `nameOrig` and `nameDest` are useless variables since they are all specific customer's information, and the feature `isFlaggedFraud` is a feature containing all 0 values, which doesn't help us build the model. So we decided to remove these features. At the same time, we created a dummy variable that identifies whether a fraudulent transaction or non-fraudulent through yes or no.



	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	0	0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	1	0
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	0	0
...	...	...	...	...	...	...	...	...	...	...	...
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	1	0
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	1	0
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	1	0
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	1	0
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	1	0

6362620 rows × 11 columns

Figure 1: Data information

We imaged two of the features we are more interested in, they are `step` (a unit of time in the real world. In this case 1 step is 1 hour of time.) and `type` (CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER). In our opinion, these two features might be the most influential to determine whether or not the transaction would be a fraudulent transaction or non-fraudulent transaction. We are hoping to observe their distribution to see if we can see something. Through these two graphs in *Figure 2*, we realize one thing, the number of fraudulent transactions is much more than that of non-fraudulent transactions, in total over 6 million observation, there are only no more than 10 thousands observation are fraudulent transaction, which will greatly affect our subsequent models, so we decided to do downward sampling on the data.

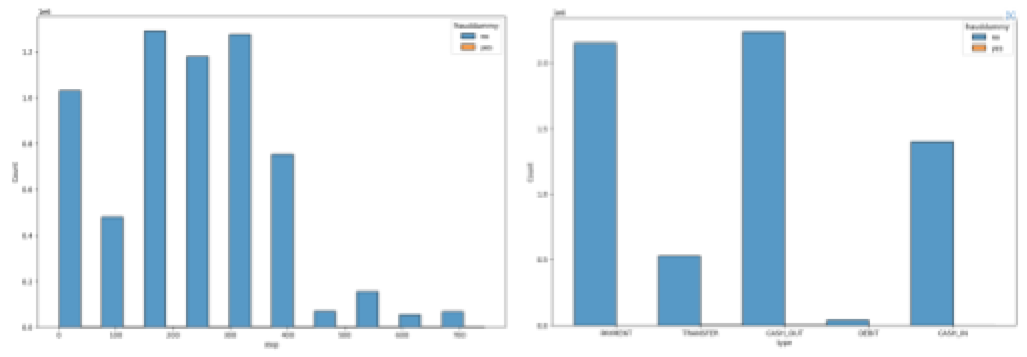


Figure 2: Data distribution of step and type

Downward sampling is a technique used to balance an imbalanced dataset by randomly removing samples from the majority class until the dataset is balanced. After we do the downward sampling, we fit the two plots again, we can see the plots in *Figure 3* become more satisfying, however, there is not too much information we can get from these plots.

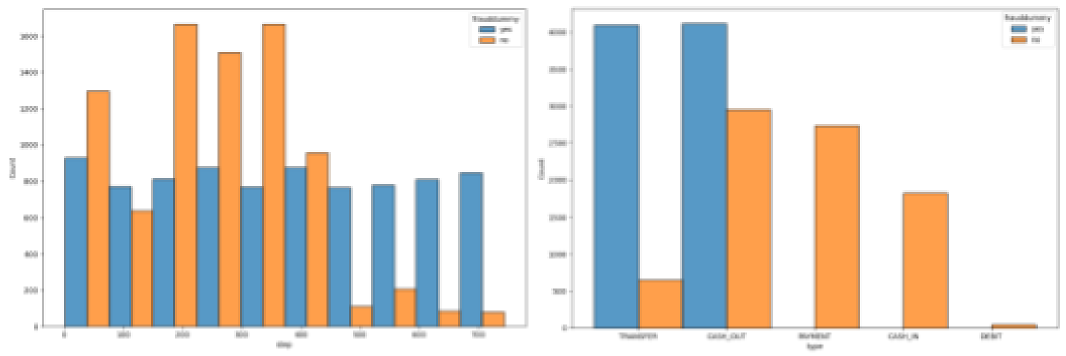
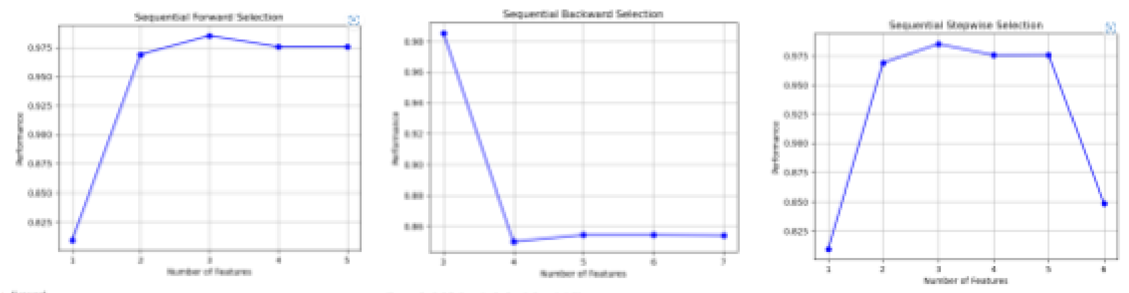


Figure 3: Data information of step and type after downward sampling

At the same time, we also create a correlation heatmap, we can see there is a strong correlation between the old balance and the new balance, although our intuition tells us that the old balance and new balances of customer's balance and recipient's balance are all affected by the amount. The difference between them is not very big, but the correlation between them and the amount is not always very strong. but we decided to not remove them.

4. Feature selection

We decided to do a feature selection to decide which of the features we are going to use to fit the models. The way we choose to use is Sequential feature selection. Sequential feature selection algorithms are a family of greedy search algorithms that are used to reduce an initial d-dimensional feature space to a k-dimensional feature subspace where k < d. The sequential forward selection provides five features, and the sequential backward selection provides three features. We decided to do one more sequential Forward Floating Selection since Stepwise Regression prevents multicollinearity problems to a great extent. And the final three features we got are \`amount\`, \`oldblanceOrig\`, and \`newblanceOrig\` which you can see in *Figure 5*. After that, we apply those features to our new data, we use train size = 0.8 to split the dataset into training dataset and testing dataset. We also do the normalizing data.



```
['amount',
 'oldbalanceOrg',
 'newbalanceOrig',
 'oldbalanceDest',
 'newbalanceDest']
```

```
['amount', 'oldbalanceOrg', 'newbalanceOrig']
```

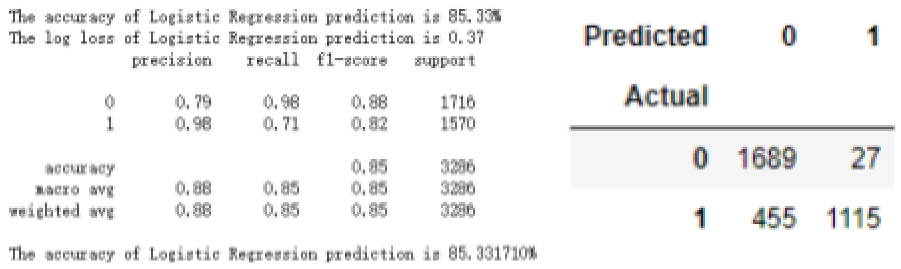
```
['amount', 'oldbalanceOrg', 'newbalanceOrig']
```

Figure 5: Sequential feature selection

5. Model details

5.1 Logistic Regression

We evaluate the logistic regression model as part of our machine learning pipeline. Logistic regression is a widely-used classification algorithm that models the relationship between input variables and binary outcomes. By utilizing a linear function and a sigmoid activation, logistic regression estimates the probability of an instance belonging to a specific class.For our experiment, we employ a logistic regression model with default hyperparameters. The model is trained on the provided dataset, and the class_weight parameter is not adjusted as the dataset exhibits a relatively balanced class distribution. Upon evaluating the logistic regression model, we assess its performance based on key metrics such as accuracy, precision, recall, and F1 score. Additionally, we present a detailed analysis of the model's performance in a visual representation, providing insights into its ability to classify fraudulent transactions accurately. In our result of Logistic Regression in Figure 6, we can see that the accuracy of Logistic Regression prediction is 85.33% and the accuracy of Logistic Regression prediction is 85.33%.



```
The accuracy of Logistic Regression prediction is 85.33%
The log loss of Logistic Regression prediction is 0.37
              precision    recall  f1-score   support

           0       0.79      0.98      0.88      1716
           1       0.98      0.71      0.82      1570

    accuracy                           0.85      3286
   macro avg       0.88      0.85      0.85      3286
weighted avg       0.88      0.85      0.85      3286

The accuracy of Logistic Regression prediction is 85.331710%
```

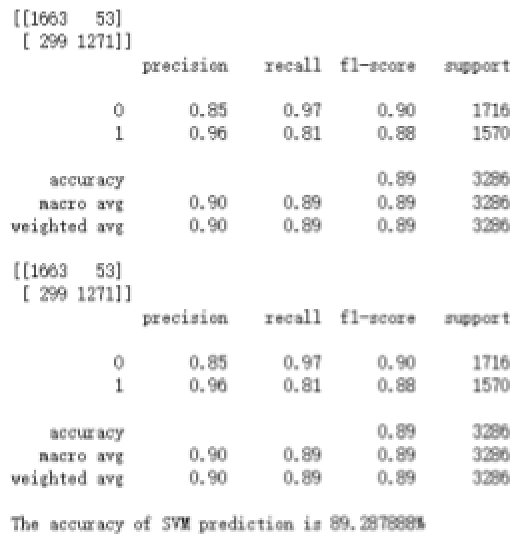
Predicted	0	1
Actual		
0	1689	27
1	455	1115

Figure 6: Result of Logistic Regression

5.2 Support Vector Machines (SVM)

In our project, we incorporate Support Vector Machines (SVM) as one of the machine learning models for fraud detection. SVM is a powerful algorithm known for its effectiveness in both linear and non-linear classification tasks. It aims to find an optimal hyperplane that maximally separates the different classes in the dataset. To fine-tune the SVM model and optimize its performance, we employ a technique called grid search. Grid search involves systematically searching through a predefined set of hyperparameter combinations to identify the optimal configuration for the model. By exhaustively evaluating various combinations of hyperparameters, such as the choice of kernel (e.g., linear, polynomial, or radial basis function) and the regularization parameter (C), grid search helps us identify the best settings for the SVM model.

We fit 5 folds for each of 90 candidates, totalling 450 fits, the best fit provided by the grid search is regularization parameter = 5, degree = 1, gamma value = scale, and the kernel = rbf. Applying the best fit detail into our Support Vector Machines model, we got the accuracy of SVM prediction is 89.29% which can be seen in *Figure 7*.



```
[[1663   53]
 [ 299 1271]]
              precision    recall  f1-score   support

           0       0.85      0.97      0.90      1716
           1       0.96      0.81      0.88      1570

    accuracy                           0.89      3286
   macro avg       0.90      0.89      0.89      3286
weighted avg       0.90      0.89      0.89      3286

[[1663   53]
 [ 299 1271]]
              precision    recall  f1-score   support

           0       0.85      0.97      0.90      1716
           1       0.96      0.81      0.88      1570

    accuracy                           0.89      3286
   macro avg       0.90      0.89      0.89      3286
weighted avg       0.90      0.89      0.89      3286

The accuracy of SVM prediction is 89.287888%
```

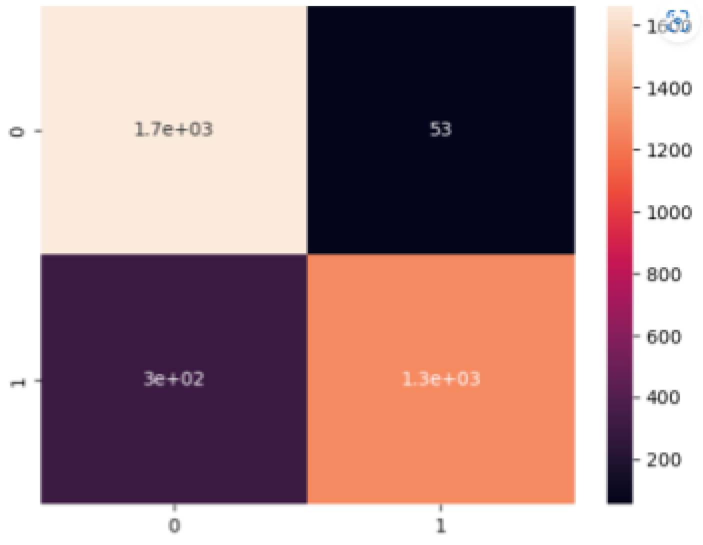
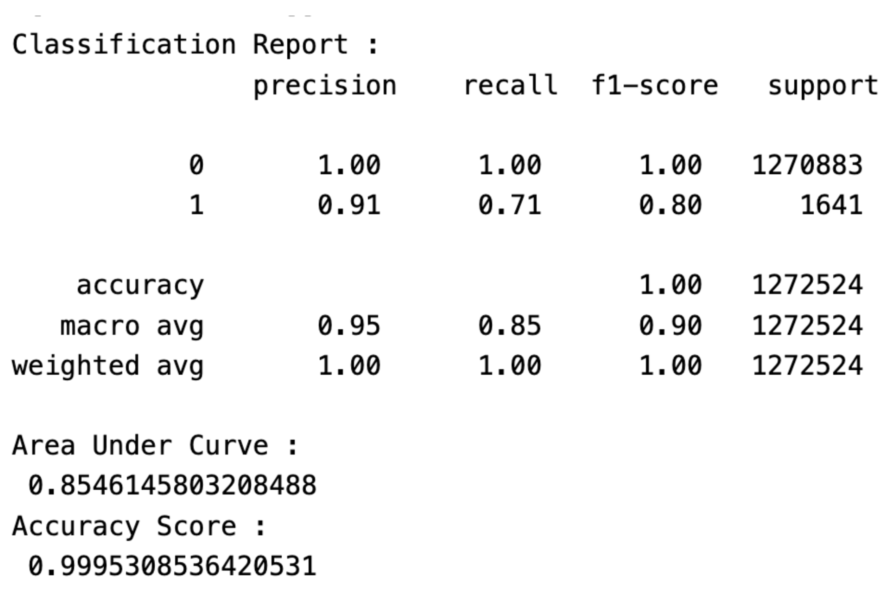


Figure 7: The result of Support Vector Model

5.3 Random Forest

We experiment with a vanilla random forest algorithm. Random Forest is a popular ensemble learning algorithm for classification tasks. It works by building a large number of decision trees, each trained on a random subset of the training data and using a random subset of the features.The number of decision trees is set to 25. The class_weight argument is set to true, which automatically adjusts the weights of the classes to be inversely proportional to their frequency in the training data. This can help to improve the performance of the algorithm on imbalanced datasets. The performance of the model is summarized in the figure below.



```
Classification Report :
               precision    recall  f1-score   support

           0       1.00      1.00      1.00   1270883
           1       0.91      0.71      0.80      1641

    accuracy                           1.00   1272524
   macro avg       0.95      0.85      0.90   1272524
weighted avg       1.00      1.00      1.00   1272524

Area Under Curve :
 0.8546145803208488
Accuracy Score :
 0.9995308536420531
```

Figure 8. Result of Random Forest

The model archives 0.99 accuracy on the test set. The use of random subsets of the training data and features helps to reduce overfitting and improve the generalization performance of the model. In addition, by aggregating the predictions of multiple decision trees, the random forest algorithm achieves high accuracy and robustness to noise and outliers in the data.

5.4 Extreme Gradient Boosting

We experiment with Extreme Gradient Boosting (XGBoost). XGBoost is different from vanilla boosting algorithms in the following ways: First, it introduce L1 and L2 regularization on the weights of the trees to prevent overfitting; second, it uses a technique called "approximate greedy algorithm" that can quickly find the best split points for each tree, without having to search the entire feature space. The performance of the model is summarized in the figure below.

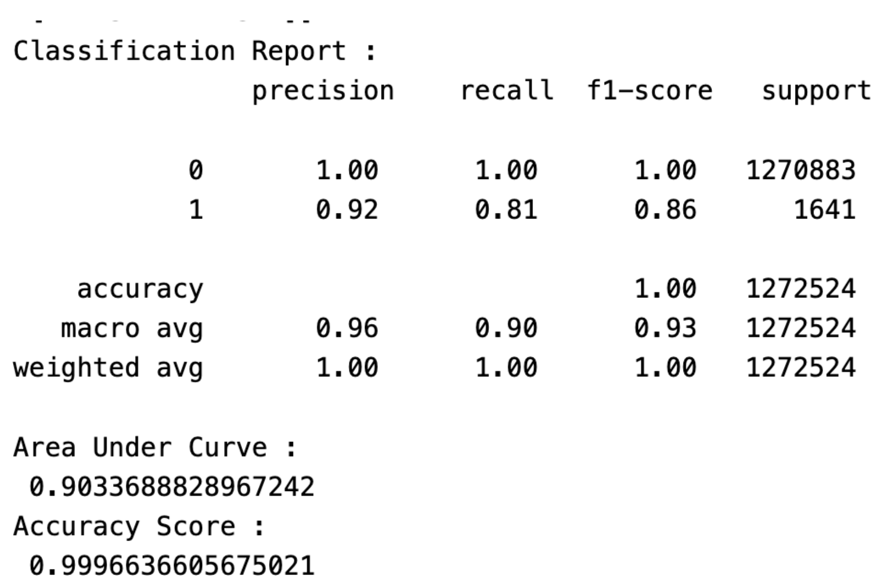

```
Classification Report :
               precision    recall  f1-score   support

           0       1.00      1.00      1.00   1270883
           1       0.92      0.81      0.86      1641

    accuracy                           1.00   1272524
   macro avg       0.96      0.90      0.93   1272524
weighted avg       1.00      1.00      1.00   1272524

Area Under Curve :
 0.9033688828967242
Accuracy Score :
 0.9996636605675021
```

Figure 9. Result of Extreme Gradient Boosting

This model achieves 0.99 accuracy on the test set. Notably, it takes less than 1 minute to train the model. It is impressive that XGBoost can achieve higher accuracy and faster training times at the same time.

6. Conclusion

Fraud detection helps prevent financial losses and protects individuals and businesses from the negative consequences of fraudulent activity. In this project, we performed extensive EDA to explore the nature of the dataset; utilized multiple techniques to preprocess the data; and experimented with four different classes of machine learning models to perform the classification task. The best two algorithms turn out to be Random Forest and Extreme Gradient Boosting, which both achieve 99.99% accuracy on the previously unseen test data.

However, for this model to be more informative and valuable for real world deployment, further work needs to be performed. For instance, real data would be more desirable because the synthetic dataset we use may not capture may not capture the nuances of human behavior or the intrinsics of the complex financial system. Additionally, synthetic datasets may suffer from biases and limitations inherent in the algorithms used to generate them.

7. Discussion and Recommendations

We consider three aspects of whether our project would become a weapon of math destruction (WMD):

(1) Is the model's outcome hard to measure? Yes. Since fraud detection is inherently a challenging task, it is hard to accurately measure the outcome (i.e., tell if a transaction is INDEED fraudulent).

(2) Will the predictions have negative consequences? Yes. High false negative (i.e., flagging a non-fraud transaction as fraud) will hinder normal transactions; high false positive (i.e., flagging frauds as normal transactions) can lead to financial losses.

(3) Will the predictions create self-fulfilling (or defeating) feedback loops? Unlikely. The fraud detection process is independent of the transactions so it's unlikely to create a feedback loop.

So there's still a chance that our project becomes a WMD if we don't carefully consider the design and implementation of the model, including the data used to train it, the algorithms and methods used, and the metrics used to evaluate its performance.

Ensuring fairness in our fraud detection model is crucial to prevent innocent individuals from being incorrectly flagged as fraudulent and to prevent fraudulent activities from going undetected. When selecting a model for fraud detection, it is important to consider whether the model is biased against certain groups of people or influenced by irrelevant factors. For example, a model that relies heavily on demographic or geographic data may be more likely to discriminate against certain groups of people. Similarly, a model that is trained on historical data that reflects past discrimination or systemic bias may perpetuate these biases in its predictions. Thus we performed extensive preprocessing of the dataset to ensure the fairness of our models.

8. References

[1] Pudil, P., Novovičová, J., & Kittler, J. (1994). "Floating search methods in feature selection." Pattern recognition letters 15.11 (1994): 1119-1125.

[2] Ferri, F. J., Pudil P., Hatef, M., Kittler, J. (1994). "Comparative study of techniques for large-scale feature selection." Pattern Recognition in Practice IV : 403-413.

[3] EDGAR LOPEZ-ROJAS (2017) Synthetic Financial Datasets For Fraud Detection, Version 2. Retrieved May 12, 2023 from https://www.kaggle.com/datasets/ealaxi/paysim1

9. Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
!pip install mlxtend
import joblib
import sys
sys.modules['sklearn.externals.joblib'] = joblib
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from sklearn.linear_model import LinearRegression
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs
df = pd.read_csv("Synthetic Financial Datasets.csv")
df
df.info()
df.describe()
del df['nameOrig']
del df['nameDest']
del df['isFlaggedFraud']
df['frauddummy']=np.where(df['isFraud']==0,"no","yes")
df.head()
plt.figure(figsize=(12,8))
sns.histplot(data=df, x="step", hue="frauddummy", multiple="dodge", bins=10)
plt.figure(figsize=(12,8))
sns.histplot(data=df, x="type", hue="frauddummy", multiple="dodge", bins=5)
fraud_cases = df[df["isFraud"] == 1]
non_fraud_cases = df[df["isFraud"] == 0]
n_samples = min(len(fraud_cases), len(non_fraud_cases))
non_fraud_cases_downsampled = non_fraud_cases.sample(n=n_samples, random_state=42)
df_downsampled = pd.concat([fraud_cases, non_fraud_cases_downsampled])
plt.figure(figsize=(12,8))
sns.countplot(data=df_downsampled, x="frauddummy")
plt.figure(figsize=(12,8))
sns.histplot(data=df_downsampled, x="step", hue="frauddummy", multiple="dodge", bins=10)
plt.figure(figsize=(12,8))
sns.histplot(data=df_downsampled, x="type", hue="frauddummy", multiple="dodge", bins=5)
df_downsampled['type'], _ = pd.factorize(df_downsampled['type'], sort=True)
df_downsampled['frauddummy'], _ = pd.factorize(df_downsampled['frauddummy'], sort=True)
del df_downsampled["isFraud"]
df_downsampled.info()
plt.figure(figsize=(24, 12))
heatmap = sns.heatmap(df_downsampled.corr(), vmin=-1, vmax=1, annot=True, cmap='BrBG')
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':18}, pad=12)
X = df_downsampled.drop('frauddummy',axis = 1)
y = df_downsampled.loc[:,'frauddummy']
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8)
print(f"Length of the Dataset:, {len(X)}")
print(f"Length of the Training Dataset:, {len(X_train)}")
print(f"Length of the Training Dataset responese:, {len(y_train)}")
print(f"Length of the Test Dataset:, {len(X_test)}")
sfs = SFS(LinearRegression(),
          k_features=5,
          forward=True,
          floating=False,
          scoring = 'roc_auc',
          cv = 0)
sfs.fit(X_train, y_train)
plot_sfs(sfs.get_metric_dict(), kind='std_dev')
plt.title('Sequential Forward Selection')
plt.grid()
plt.show()
sfs.k_feature_names_
sbs = SFS(LinearRegression(),
          k_features=3,
          forward=False,
          floating=False,
          scoring = 'roc_auc',
          cv = 0)
sbs.fit(X_train, y_train)
plot_sfs(sbs.get_metric_dict(), kind='std_dev')
plt.title('Sequential Backward Selection')
plt.grid()
plt.show()
sbs.k_feature_names_
sffs = SFS(LinearRegression(),
         k_features=(2,6),
         forward=True,
         floating=True,
         scoring = 'roc_auc',
         cv=0)
sffs.fit(X_train, y_train)
fig1 = plot_sfs(sffs.get_metric_dict(), kind='std_dev')
plt.title('Sequential Stepwise Selection')
plt.grid()
plt.show()
sffs.k_feature_names_
new_X = X[['amount', 'oldbalanceOrg', 'newbalanceOrig']]
X_train, X_test, y_train, y_test = train_test_split(new_X, y, train_size = 0.8)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
scaler.fit(X_test)
X_train = pd.DataFrame(scaler.transform(X_train), index=X_train.index, columns=X_train.columns)
X_test = pd.DataFrame(scaler.transform(X_test), index=X_test.index, columns=X_test.columns)
#getting libraries for different ml methods
from sklearn import svm
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.metrics import accuracy_score, classification_report, log_loss
lr = LogisticRegression(C = 1, penalty = 'l2', tol = 1e-4, solver = 'saga')
lr = lr.fit(X_train, y_train)
lr_predictions = lr.predict(X_test)
print("The accuracy of Logistic Regression prediction is", '{:.2%}'.format(accuracy_score(y_test, lr_predictions)))
print("The log loss of Logistic Regression prediction is",
      '{:.2}'.format(log_loss(y_test, lr.predict_proba(X_test))))
print(classification_report(y_test, lr_predictions))
pd.crosstab(index = y_test,
           columns = lr_predictions,
           rownames = ['Actual'],
           colnames = ['Predicted'],
           margins = False)
print("The accuracy of Logistic Regression prediction is", '{:.6%}'.format(accuracy_score(y_test, lr_predictions)))
pd.crosstab(index = y_test,
           columns = lr_predictions,
           rownames = ['Actual'],
           colnames = ['Predicted'],
           margins = False)
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
param_grid = {'C': [1,2,3,4,5], #regularizaiton
             'gamma': ['scale','auto'],
             'kernel': ['rbf','sigmoid','poly'],
             'degree': [1,2,3]}
grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)
```

```
grid.fit(X_train, y_train)
print(grid.best_params_)
print(grid.best_estimator_)
from sklearn.metrics import confusion_matrix
rbf_svc = SVC(C= 5, degree= 1, gamma='scale', kernel = 'rbf')
rbf_svc.fit(X_train, y_train)
rbf_pred = rbf_svc.predict(X_test)
print(confusion_matrix(y_test, rbf_pred))
print(classification_report(y_test, rbf_pred))
rbf_cm = confusion_matrix(y_test, rbf_pred)
sns.heatmap(rbf_cm,annot=True)
plt.savefig('confusion_Matrix.png')
print(rbf_cm)
accuracy_score(y_test,rbf_pred)
print(classification_report(y_test, rbf_pred))
print("The accuracy of SVM prediction is", '{:.6%}'.format(accuracy_score(y_test, rbf_pred)))
# Initiate model
RFC = RandomForestClassifier(n_estimators = 25, class_weight = 'balanced', random_state=0)
# Fit the model
RFC_fit = RFC.fit(x_train, y_train)
# Predict the test set
RFC_pred = RFC.predict(x_test)
# Evaluating model
CM_RFC = confusion_matrix(y_test,RFC_pred)
CR_RFC = classification_report(y_test,RFC_pred)
fprRFC, recallRFC, thresholdsRFC = roc_curve(y_test, RFC_pred)
AUC_RFC = auc(fprRFC, recallRFC)

resultsRFC = {"Confusion Matrix":CM_RFC,"Classification Report":CR_RFC,"Area Under Curve":AUC_RFC}

for measure in resultsRFC:
  print(measure,": \n",resultsRFC[measure])
# Initiate model
XGB = XGBClassifier(random_state=0)
# Fit the model
XGB_fit = XGB.fit(x_train, y_train)
# Predict the test set
XGB_pred = XGB.predict(x_test)
# Evaluating model
CM_XGB = confusion_matrix(y_test,XGB_pred)
CR_XGB = classification_report(y_test,XGB_pred)
fprXGB, recallXGB, thresholdsXGB = roc_curve(y_test, XGB_pred)
AUC_XGB = auc(fprXGB, recallXGB)
resultsXGB = {"Confusion Matrix":CM_XGB,"Classification Report":CR_XGB,"Area Under Curve":AUC_XGB}
for measure in resultsXGB:
  print(measure,": \n",resultsXGB[measure])
```